Extensible Prioritization Scheme for HTTP

Lucas Pardue and Kazuho Oku

draft-ietf-httpbis-priority

HEADERS & FRAMES
HEADERS
FRAMES

New in -02: H3 PRIORITY_UPDATE breaking change

-01

- Type **0xF**
- **T** bit - Element ID is Request or Push
- Priority Field Value is ASCII encoded Structured Headers

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|T|    Empty    |   Prioritized Element ID (i)                ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Priority Field Value (*)                  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

-02

- Replace bitfield with frame type
- **0xF0700** applies to Request
- **0xF0701** applies to Push

```
HTTP/3 PRIORITY_UPDATE Frame {
  Type (i) = 0xF0700..0xF0701,
  Length (i),
  Prioritized Element ID (i),
  Priority Field Value (..),
}
```

New in -02

1. Add consideration for server scheduling (Section 9)
2. Remove specific instruction about intermediary fairness (Section 10.2)
3. Describe considerations when clients use PRIORITY_UPDATE (Section 6)

   Problem: Race condition where frame arrives before request

   Solution: Servers SHOULD buffer **most recent frame** and apply it once steam is opened.

   Need to add some further text on how servers can limit the state commitment.

Open issues

- Need a way to tell if server supports priority #1274
- "Within the stream limit" in HTTP/2? #1261
- The default priority of a pushed request #1056

#1274 - Need a way to tell if server supports priority (1)

We threw a lot of RFC 7540 prioritization away. But we maintained the principle in section 5.3:

Explicitly setting the priority for a stream is input to a prioritization process. It does not guarantee any particular processing or transmission order for the stream relative to any other stream. An endpoint cannot force a peer to process concurrent streams in a particular order using priority. Expressing priority is therefore only a suggestion.

#1274 - Need a way to tell if server supports priority (2)

Some applications that use HTTP* want to take advantage of stream multiplexing.

Example provided on the issue Apple LL-HLS. Strict scheduling, such as delivery order and bandwidth splitting allows for optimizations. If requirements are not met it could cause regressions.

Extensible Priorities could be a way to achieve that, if servers do exactly what they are asked. But that is at odds with what our principles have been.

Issue proponents say that client can react to if it knows the server **has not** scheduled streams as requested.

**draft-ietf-httpbis-bcp56bis*

#1274 - Need a way to tell if server supports priority (3)

One proposal from Roger Pantos on PR 1283

Define an “applied” sf-boolean parameter that a server sends to indicate that the requested priority was applied to the response.

There may be variants of this approach.

But there are many edge cases here. Editors believe it adds complexity to Extensible Priorities. It might also provide an unintended incentive for servers to lie.

#1274 - Need a way to tell if server supports priority (4)

Alternate proposal from Roger Pantos on the issue. To summarise:

Given weak guarantees of Extensible Priorities, allow LL-HLS to mandate things. Define prioritization levels specialized for LL-HLS resources, and how an LL-HLS server is expected to act on them. A client might find it difficult to validate but a HLS stream validator tool could.

This avoids needing changes to Extensible Priorities spec.

Might be good enough. Roger is seeking input from HLS community and Apple folks.

#1261 - "Within the stream limit" in HTTP/2? (1)

For HTTP/2 PRIORITY_UPDATE frame we say:

“The Prioritized Stream ID MUST be within the stream limit.”

Mike Bishop points out that

“HTTP/2 *specifies a maximum number of concurrently active streams, with streams which are either "open" or "half-open" counting toward the limit regardless of the ID”*

Are we attempting to redefine HTTP/2? No (if we can avoid it).

#1261 - "Within the stream limit" in HTTP/2? (2)

Kazuho suggests we rephrase the requirement to concurrency window size.

And while we’re fixing this we can probably add more text about buffering and limits in Section 6.

Implementation experience can help make this better.

Default priority of a push #1056 (1)

Server Push: the request and response headers are created by the server (or in an intermediary case, the origin). If either of the two contains a priority signal, resolve the merge as normal.

The question is what to do when there is **no priority signal**.

Default priority of a push #1056 (2)

Omitting priority signal in **normal** request/response: urgency=**3**, incremental=**false**

RFC 7540, Section 5.3.5 “*pushes initially depend on their associated stream … default weight of 16*”

Extensible priorities has no **dependency**, so what are a push’s default urgency and incremental?

Default priority of a push #1056 (3)

Possible push defaults:

1) Same as the associated stream (aka “request that triggered the push”).
2) “An urgency one less than the associated stream”.
3) No default.

Considerations for picking:

- No single best value.
- Server cannot use PRIORITY_UPDATE to signal push priority
- Performance benefits are hard to measure.
- Performance degradation is very possible if wrong value is picked
  - Especially if pushed resources priority cause bad scheduling of other resources

Questions?