

HTTP
Internet-Draft
Intended status: Standards Track
Expires: May 8, 2020

K. Oku
Fastly
L. Pardue
Cloudflare
November 05, 2019

Extensible Prioritization Scheme for HTTP
draft-kazuho-httpbis-priority-03

Abstract

This document describes a scheme for prioritizing HTTP responses. This scheme expresses the priority of each HTTP response using absolute values, rather than as a relative relationship between a group of HTTP responses.

This document defines the Priority header field for communicating the initial priority in an HTTP version-independent manner, as well as HTTP/2 and HTTP/3 frames for reprioritizing the responses. These share a common format structure that is designed to provide future extensibility.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 8, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

Internet-Draft Extensible Prioritization Scheme for HTTP November 2019

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Notational Conventions	3
2.	Motivation for Replacing HTTP/2 Priorities	4
3.	Negotiating Priorities	5
3.1.	The SETTINGS_PRIORITIES SETTINGS Parameter	5
3.2.	Defined Prioritization Scheme Values	6
3.2.1.	H2_TREE	6
3.2.2.	URGENCY	7
4.	The Priority HTTP Header Field	7
4.1.	urgency	7
4.1.1.	prerequisite	8
4.1.2.	default	8
4.1.3.	supplementary	8
4.1.4.	background	9
4.2.	progressive	9
5.	Reprioritization	10
5.1.	HTTP/2 PRIORITY_UPDATE Frame	11
5.2.	HTTP/3 PRIORITY_UPDATE Frame	11
6.	Merging Client- and Server-Driven Parameters	12
7.	Security Considerations	13
7.1.	Fairness and Coalescing Intermediaries	13
8.	Considerations	14
8.1.	Why use an End-to-End Header Field?	14
8.2.	Why do Urgencies Have Meanings?	14
8.3.	Can an Intermediary Send its own Signal?	15
9.	IANA Considerations	15
9.1.	HTTP Prioritization Scheme Registry	16
10.	References	17
10.1.	Normative References	17
10.2.	Informative References	18
Appendix A.	Acknowledgements	18
Appendix B.	Change Log	19
B.1.	Since draft-kazuho-httpbis-priority-02	19
B.2.	Since draft-kazuho-httpbis-priority-01	19

B.3. Since draft-kazuho-httpbis-priority-00	19
Authors' Addresses	19

Internet-Draft Extensible Prioritization Scheme for HTTP November 2019

[1.](#) Introduction

It is common for an HTTP ([\[RFC7230\]](#)) resource representation to have relationships to one or more other resources. Clients will often discover these relationships while processing a retrieved representation, leading to further retrieval requests. Meanwhile, the nature of the relationship determines whether the client is blocked from continuing to process locally available resources. For example, visual rendering of an HTML document could be blocked by the retrieval of a CSS file that the document refers to. In contrast, inline images do not block rendering and get drawn progressively as the chunks of the images arrive.

To provide meaningful representation of a document at the earliest moment, it is important for an HTTP server to prioritize the HTTP responses, or the chunks of those HTTP responses, that it sends.

HTTP/2 ([\[RFC7540\]](#)) provides such a prioritization scheme. A client sends a series of PRIORITY frames to communicate to the server a "priority tree"; this represents the client's preferred ordering and weighted distribution of the bandwidth among the HTTP responses. However, the design and implementation of this scheme has been observed to have shortcomings, explained in [Section 2](#).

This document defines the Priority HTTP header field that can be used by both client and server to specify the precedence of HTTP responses in a standardized, extensible, protocol-version-independent, end-to-end format. Along with the protocol-version-specific frame for reprioritization, this prioritization scheme acts as a substitute for the original prioritization scheme of HTTP/2.

[1.1.](#) Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

The terms `sh-token` and `sh-boolean` are imported from [\[STRUCTURED-HEADERS\]](#).

Example HTTP requests and responses use the HTTP/2-style formatting from [\[RFC7540\]](#).

This document uses the variable-length integer encoding from [\[I-D.ietf-quick-transport\]](#).

[2.](#) Motivation for Replacing HTTP/2 Priorities

An important feature of any implementation of a protocol that provides multiplexing is the ability to prioritize the sending of information. This was an important realization in the design of HTTP/2. Prioritization is a difficult problem, so it will always be suboptimal, particularly if one endpoint operates in ignorance of the needs of its peer.

HTTP/2 introduced a complex prioritization signaling scheme that used a combination of dependencies and weights, formed into an unbalanced tree. This scheme has suffered from poor deployment and interoperability.

The rich flexibility of client-driven HTTP/2 prioritization tree building is rarely exercised; experience shows that clients either choose a single model optimized for a web use case (and don't vary it) or do nothing at all. But every client builds their prioritization tree in a different way, which makes it difficult for servers to understand their intent and act or intervene accordingly.

Many HTTP/2 server implementations do not include support for the priority scheme, some favoring instead bespoke server-driven schemes based on heuristics and other hints, like the content type of resources and the order in which requests arrive. For example, a server, with knowledge of the document structure, might want to prioritize the delivery of images that are critical to user experience above other images, but below the CSS files. Since client trees vary, it is impossible for the server to determine how such

images should be prioritized against other responses.

The HTTP/2 scheme allows intermediaries to coalesce multiple client trees into a single tree that is used for a single upstream HTTP/2 connection. However, most intermediaries do not support this. The scheme does not define a method that can be used by a server to express the priority of a response. Without such a method, intermediaries cannot coordinate client-driven and server-driven priorities.

HTTP/2 describes denial-of-service considerations for implementations. On 2019-08-13 Netflix issued an advisory notice about the discovery of several resource exhaustion vectors affecting multiple HTTP/2 implementations. One attack, CVE-2019-9513 aka "Resource Loop", is based on manipulation of the priority tree.

The HTTP/2 scheme depends on in-order delivery of signals, leading to challenges in porting the scheme to protocols that do not provide

global ordering. For example, the scheme cannot be used in HTTP/3 [[I-D.ietf-quic-http](#)] without changing the signal and its processing.

Considering the problems with deployment and adaptability to HTTP/3, retaining the HTTP/2 priority scheme increases the complexity of the entire system without any evidence that the value it provides offsets that complexity. In fact, multiple experiments from independent research have shown that simpler schemes can reach at least equivalent performance characteristics compared to the more complex HTTP/2 setups seen in practice, at least for the web use case.

The problems and insights laid out above are motivation for the alternative and more straightforward prioritization scheme presented in this document. In order to support deployment of new schemes, a general-purpose negotiation mechanism is specified in [Section 3](#).

[3](#). Negotiating Priorities

The document specifies a negotiation mechanism that allows each peer to communicate which, if any, priority schemes are supported, as well as the server's ranked preference.

For both HTTP/2 and HTTP/3, either peer's SETTINGS may arrive first, so any negotiation must be unilateral and not rely upon receiving the peer's SETTINGS value.

Servers are likely to only use one prioritization scheme at once per each connection, and may be unable to change the scheme once established, so the setting **MUST** be sent prior to the first request if it is ever sent. In HTTP/3, SETTINGS might arrive after the first request even if they are sent first. Therefore, future specifications that define alternative prioritization schemes for HTTP/3 **SHOULD** define how the server would act when it receives a stream-level priority signal prior to receiving the SETTINGS frame.

[3.1.](#) The SETTINGS_PRIORITIES SETTINGS Parameter

This document defines a new SETTINGS_PRIORITIES parameter (0x9) for HTTP/2 and HTTP/3, which allows both peers to indicate which prioritization schemes they support. The value of this parameter is interpreted in two ways depending on if it is zero or non-zero.

If the setting has a value of zero it indicates no support for priorities. If either side sends the parameter with a value of zero, clients **SHOULD NOT** send hop-by-hop priority signals (e.g., HTTP/2 PRIORITY frame) and servers **SHOULD NOT** make any assumptions based on the presence or lack thereof of such signals.

If the value is non-zero, then it is interpreted as an ordered preference list of prioritization schemes represented by 8-bit values. The least significant 8 bits indicate the sender's most preferred priority scheme, the second least significant 8 bits indicate the sender's second choice, and so on. This allows expressing support for 4 schemes in HTTP/2 and 7 in HTTP/3.

A sender **MUST** comply with the following restrictions when constructing a preference list: duplicate 8-bit values (excluding the value 0) **MUST NOT** be used, and if any byte is 0 then all more significant bytes **MUST** also be 0. An endpoint that receives a setting in violation of these requirements **MUST** treat it as a connection error of type `PROTOCOL_ERROR` for HTTP/2 [[RFC7540](#)], or of type `H3_SETTINGS_ERROR` for HTTP/3 [[I-D.ietf-quic-http](#)].

In HTTP/2, the setting SHOULD appear in the first SETTINGS frame and peers MUST NOT process the setting if it is received multiple times in order to avoid changing the agreed upon prioritization scheme.

If there is a prioritization scheme supported by both the client and server, then the server's preference order prevails and both peers SHOULD only use the agreed upon priority scheme for the remainder of the session. The server chooses because it is in the best position to know what information from the client is of the most value.

Once the negotiation is complete, endpoints MAY stop sending hop-by-hop prioritization signals that were not negotiated in order to conserve bandwidth. However, endpoints SHOULD continue sending end-to-end signals (e.g., the Priority header field), as that might have meaningful effect to other nodes that handle the HTTP message.

[3.2.](#) Defined Prioritization Scheme Values

This document defines two prioritization scheme values for use with the SETTINGS_PRIORITIES setting.

[3.2.1.](#) H2_TREE

This document defines the priority scheme identifier H2_TREE (8-bit value of 1) that indicates support for HTTP/2-style priorities ([\[RFC7540\]](#), [Section 5.3](#)).

The H2_TREE priority scheme identifier MUST NOT be sent in an HTTP/3 settings because there is no defined mapping of this scheme. Endpoints MUST treat receipt of H2_TREE as a connection error of type H3_SETTINGS_ERROR.

[3.2.2.](#) URGENCY

This document defines the priority scheme identifier URGENCY (8-bit value of 2) that indicates support for the extensible priority scheme defined in the present document.

An intermediary connecting to a backend server SHOULD declare support for the extensible priority scheme when and only when all the

requests that are to be sent on that backend connection originates from one client-side connection that has negotiated the use of the extensible priority scheme (see [Section 7.1](#)).

4. The Priority HTTP Header Field

The Priority HTTP header field can appear in requests and responses. A client uses it to specify the priority of the response. A server uses it to inform the client that the priority was overwritten. An intermediary can use the Priority information from client requests and server responses to correct or amend the precedence to suit it (see [Section 6](#)).

The value of the Priority header field is a Structured Headers Dictionary ([[STRUCTURED-HEADERS](#)]). Each dictionary member represents a parameter of the Priority header field. This document defines the "urgency" and "progressive" parameters. Values of these parameters MUST always be present. When any of the defined parameters are omitted, or if the Priority header field is not used, their default values SHOULD be applied.

Unknown parameters MUST be ignored.

4.1. urgency

The "urgency" parameter takes an integer between -1 and 6 as shown below:

Urgency		Definition
-1		prerequisite (Section 4.1.1)
0		default (Section 4.1.2)
between 1 and 5		supplementary (Section 4.1.3)
6		background (Section 4.1.4)

Table 1: Urgencies

The value is encoded as an sh-integer. The default value is zero.

A server SHOULD transmit HTTP responses in the order of their urgency

values. The lower the value, the higher the precedence.

The following example shows a request for a CSS file with the urgency set to "-1":

```
:method = GET
:scheme = https
:authority = example.net
:path = /style.css
priority = urgency=-1
```

The definition of the urgencies and their expected use-case are described below. Endpoints SHOULD respect the definition of the values when assigning urgencies.

[4.1.1.](#) prerequisite

The prerequisite urgency (value -1) indicates that the response prevents other responses with an urgency of prerequisite or default from being used.

For example, use of an external stylesheet can block a web browser from rendering the HTML. In such case, the stylesheet is given the prerequisite urgency.

[4.1.2.](#) default

The default urgency (value 0) indicates a response that is to be used as it is delivered to the client, but one that does not block other responses from being used.

For example, when a user using a web browser navigates to a new HTML document, the request for that HTML is given the default urgency. When that HTML document uses a custom font, the request for that custom font SHOULD also be given the default urgency. This is because the availability of the custom font is likely a precondition for the user to use that portion of the HTML document, which is to be rendered by that font.

[4.1.3.](#) supplementary

The supplementary urgency indicates a response that is helpful to the client using a composition of responses, even though the response itself is not mandatory for using those responses.

For example, inline images (i.e., images being fetched and displayed as part of the document) are visually important elements of an HTML

document. As such, users will typically not be prevented from using the document, at least to some degree, before any or all of these images are loaded. Display of those images are thus considered to be an improvement for visual clients rather than a prerequisite for all user agents. Therefore, such images will be given the supplementary urgency.

Values between 1 and 5 are used to represent this urgency, to provide flexibility to the endpoints for giving some responses more or less precedence than others that belong to the supplementary group. [Section 6](#) explains how these values might be used.

Clients SHOULD NOT use values 1 and 5. Servers MAY use these values to prioritize a response above or below other supplementary responses.

Clients MAY use values 2 to indicate that a request is given relatively high priority, or 4 to indicate relatively low priority, within the supplementary urgency group.

For example, an image certain to be visible at the top of the page, might be assigned a value of 2 instead of 3, as it will have a high visual impact for the user. Conversely, an asynchronously loaded JavaScript file might be assigned an urgency value of 4, as it is less likely to have a visual impact.

When none of the considerations above is applicable, the value of 3 SHOULD be used.

[4.1.4.](#) background

The background urgency (value 6) is used for responses of which the delivery can be postponed without having an impact on using other responses.

As an example, the download of a large file in a web browser would be assigned the background urgency so it would not impact further page loads on the same connection.

[4.2.](#) progressive

The "progressive" parameter takes an sh-boolean as the value that indicates if a response can be processed progressively, i.e. provide some meaningful output as chunks of the response arrive.

The default value of the "progressive" parameter is "0".

Internet-Draft Extensible Prioritization Scheme for HTTP November 2019

A server SHOULD distribute the bandwidth of a connection between progressive responses that share the same urgency.

A server SHOULD transmit non-progressive responses one by one, preferably in the order the requests were generated. Doing so maximizes the chance of the client making progress in using the composition of the HTTP responses at the earliest moment.

The following example shows a request for a JPEG file with the urgency parameter set to "3" and the progressive parameter set to "1".

```
:method = GET
:scheme = https
:authority = example.net
:path = /image.jpg
priority = urgency=3, progressive=?1
```

5. Reprioritization

Once a client sends a request, circumstances might change and mean that it is beneficial to change the priority of the response. As an example, a web browser might issue a prefetch request for a JavaScript file with the urgency parameter of the Priority request header field set to "urgency=6" (background). Then, when the user navigates to a page which references the new JavaScript file, while the prefetch is in progress, the browser would send a reprioritization frame with the priority field value set to "urgency=-1" (prerequisite).

However, a client cannot reprioritize a response by using the Priority header field. This is because an HTTP header field can only be sent as part of an HTTP message. Therefore, to support reprioritization, it is necessary to define a HTTP-version-dependent mechanism for transmitting the priority parameters.

This document specifies a new PRIORITY_UPDATE frame type for HTTP/2 ([RFC7540]) and HTTP/3 ([I-D.ietf-quic-http]) that is specialized for reprioritization. It carries updated priority parameters and

references the target of the reprioritization based on a version-specific identifier; in HTTP/2 this is the Stream ID, in HTTP/3 this is either the Stream ID or Push ID.

In HTTP/2 and HTTP/3 a request message sent on a stream transitions it into a state that prevents the client from sending additional frames on the stream. Modifying this behavior requires a semantic change to the protocol, this is avoided by restricting the stream on which a `PRIORITY_UPDATE` frame can be sent. In HTTP/2 the frame is on

stream zero and in HTTP/3 it is sent on the control stream ([[I-D.ietf-quic-http](#)], Section 6.2.1).

5.1. HTTP/2 `PRIORITY_UPDATE` Frame

The HTTP/2 `PRIORITY_UPDATE` frame (type=0xF) carries the stream ID of the response that is being reprioritized, and the updated priority in ASCII text, using the same representation as that of the Priority header field value.

The Stream Identifier field ([[RFC7540](#)], [Section 4.1](#)) in the `PRIORITY_UPDATE` frame header MUST be zero (0x0).

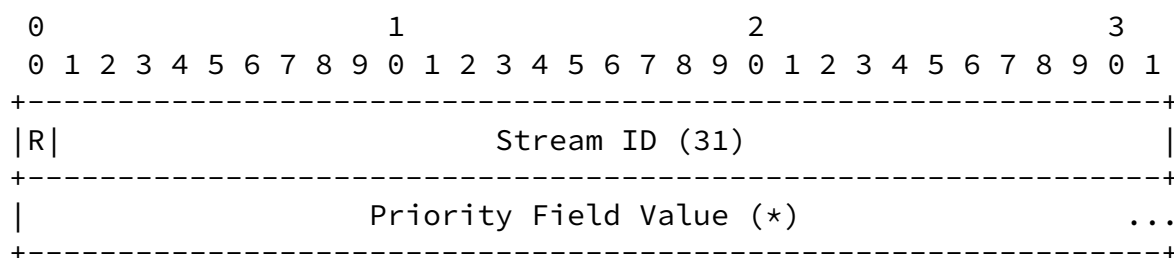


Figure 1: HTTP/2 `PRIORITY_UPDATE` Frame Payload

TODO: add more description of how to handle things like receiving `PRIORITY_UPDATE` on wrong stream, a `PRIORITY_UPDATE` with an invalid ID, etc.

5.2. HTTP/3 `PRIORITY_UPDATE` Frame

The HTTP/3 `PRIORITY_UPDATE` frame (type=0xF) carries the identifier of the element that is being reprioritized, and the updated priority in ASCII text, using the same representation as that of the Priority

header field value.

The PRIORITY_UPDATE frame MUST be sent on the control stream ([I-D.ietf-guic-http], Section 6.2.1).

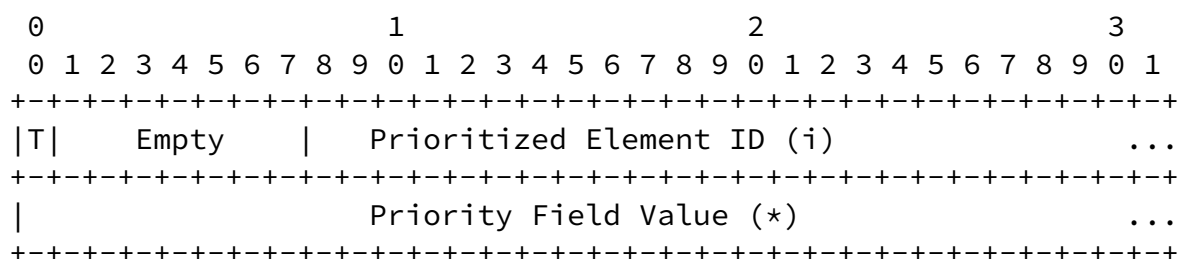


Figure 2: HTTP/3 PRIORITY_UPDATE Frame Payload

The `PRIORITY_UPDATE` frame payload has the following fields:

T (Prioritized Element Type): A one-bit field indicating the type of element being prioritized. A value of 0 indicates a reprioritization for a Request Stream, so the Prioritized Element ID is interpreted as a Stream ID. A value of 1 indicates a reprioritization for a Push stream, so the Prioritized Element ID is interpreted as a Push ID.

Empty: A seven-bit field that has no semantic value.

TODO: add more description of how to handle things like receiving PRIORITY_UPDATE on wrong stream, a PRIORITY_UPDATE with an invalid ID, etc.

6. Merging Client- and Server-Driven Parameters

It is not always the case that the client has the best understanding of how the HTTP responses deserve to be prioritized. For example, use of an HTML document might depend heavily on one of the inline images. Existence of such dependencies is typically best known to the server.

By using the "Priority" response header, a server can override the prioritization hints provided by the client. When used, the parameters found in the response header field overrides those specified by the client.

For example, when the client sends an HTTP request with

```
:method = GET
:scheme = https
:authority = example.net
:path = /menu.png
priority = urgency=3, progressive=?1
```

and the origin responds with

```
:status = 200
content-type = image/png
priority = urgency=1
```

the intermediary's understanding of the urgency is promoted from "3" to "1", because the server-provided value overrides the value provided by the client. The progressiveness continues to be "1", the value specified by the client, as the server did not specify the "progressive" parameter.

[7.](#) Security Considerations

[7.1.](#) Fairness and Coalescing Intermediaries

When an intermediary coalesces HTTP requests coming from multiple clients into one HTTP/2 or HTTP/3 connection going to the backend server, requests that originate from one client might have higher precedence than those coming from others.

It is sometimes beneficial for the server running behind an intermediary to obey to the value of the Priority header field. As an example, a resource-constrained server might defer the transmission of software update files that would have the background urgency being associated. However, in the worst case, the asymmetry between the precedence declared by multiple clients might cause responses going to one end client to be delayed totally after those going to another.

In order to mitigate this fairness problem, when a server responds to a request that is known to have come through an intermediary, the server SHOULD prioritize the response as if it was assigned the priority of "urgency=0, progressive=?1" (i.e. round-robin) regardless of the value of the Priority header field being transmitted, unless the server has the knowledge that no intermediaries are coalescing requests from multiple clients. That can be determined by the settings when the intermediaries support this specification (see [Section 3.2.2](#)), or else through configuration.

A server can determine if a request came from an intermediary through configuration, or by consulting if that request contains one of the following header fields:

- o CDN-Loop ([\[RFC8586\]](#))
- o Forwarded, X-Forwarded-For ([\[RFC7239\]](#))
- o Via ([\[RFC7230\]](#), [Section 5.7.1](#))

Responding to requests coming through an intermediary in a round-robin manner works well when the network bottleneck exists between the intermediary and the end client, as the intermediary would be buffering the responses and then be forwarding the chunks of those buffered responses based on the prioritization scheme it implements. A sophisticated server MAY use a weighted round-robin reflecting the urgencies expressed in the requests, so that less urgent responses would receive less bandwidth in case the bottleneck exists between the server and the intermediary.

[8.](#) Considerations

[8.1.](#) Why use an End-to-End Header Field?

Contrary to the prioritization scheme of HTTP/2 that uses a hop-by-hop frame, the Priority header field is defined as end-to-end.

The rationale is that the Priority header field transmits how each response affects the client's processing of those responses, rather than how relatively urgent each response is to others. The way a client processes a response is a property associated to that client

generating that request. Not that of an intermediary. Therefore, it is an end-to-end property. How these end-to-end properties carried by the Priority header field affect the prioritization between the responses that share a connection is a hop-by-hop issue.

Having the Priority header field defined as end-to-end is important for caching intermediaries. Such intermediaries can cache the value of the Priority header field along with the response, and utilize the value of the cached header field when serving the cached response, only because the header field is defined as end-to-end rather than hop-by-hop.

It should also be noted that the use of a header field carrying a textual value makes the prioritization scheme extensible; see the discussion below.

[8.2.](#) Why do Urgencies Have Meanings?

One of the aims of this specification is to define a mechanism for merging client- and server-provided hints for prioritizing the responses. For that to work, each urgency level needs to have a well-defined meaning. As an example, a server can assign the highest precedence among the supplementary responses to an HTTP response carrying an icon, because the meaning of "urgency=1" is shared among the endpoints.

This specification restricts itself to defining a minimum set of urgency levels in order to provide sufficient granularity for prioritizing responses for ordinary web browsing, at minimal complexity.

However, that does not mean that the prioritization scheme would forever be stuck to the eight levels. The design provides extensibility. If deemed necessary, it would be possible to subdivide any of the eight urgency levels that are currently defined. Or, a graphical user-agent could send a "visible" parameter to indicate if the resource being requested is within the viewport.

A server can combine the hints provided in the Priority header field with other information in order to improve the prioritization of responses. For example, a server that receives requests for a font [[RFC8081](#)] and images with the same urgency might give higher

precedence to the font, so that a visual client can render textual information at an early moment.

[8.3.](#) Can an Intermediary Send its own Signal?

There might be a benefit in recommending a coalescing intermediary to embed its own prioritization hints into the HTTP request that it forwards to the backend server, as otherwise the Priority header field would not be as helpful to the backend (see [Section 7.1](#)).

One way of achieving that, without dropping the original signal, would be to let the intermediary express its own signal using the Priority header field, at the same time transplanting the original value to a different header field.

As an example, when a client sends an HTTP request carrying a priority of "urgency=-1" and the intermediary wants to instead associate "urgency=0; progressive=?1", the intermediary would send a HTTP request that contains to the following two header fields to the backend server:

```
priority = urgency=0; progressive=?1
original-priority = urgency=-1
```

[9.](#) IANA Considerations

This specification registers the following entry in the Permanent Message Header Field Names registry established by [\[RFC3864\]](#):

Header field name: Priority

Applicable protocol: http

Status: standard

Author/change controller: IETF

Specification document(s): This document

Related information: n/a

This specification registers the following entry in the HTTP/2 Settings registry established by [\[RFC7540\]](#):

Name: SETTINGS_PRIORITIES

Code: 0x9

Initial value: 0

Specification: This document

This specification registers the following entry in the HTTP/2 Settings registry established by [[I-D.ietf-quic-http](#)]:

Name: SETTINGS_PRIORITIES

Code: 0x9

Initial value: 0

Specification: This document

This specification registers the following entry in the HTTP/2 Frame Type registry established by [[RFC7540](#)]:

Frame Type: PRIORITY_UPDATE

Code: 0xF

Specification: This document

This specification registers the following entries in the HTTP/3 Frame Type registry established by [[I-D.ietf-quic-http](#)]:

Frame Type: PRIORITY_UPDATE

Code: 0xF

Specification: This document

[9.1.](#) HTTP Prioritization Scheme Registry

This document establishes a registry for HTTP prioritization scheme codes to be used in conjunction with the SETTINGS_PRIORITIES parameter. The "HTTP Prioritization Scheme" registry manages an 8-bit space. The "HTTP Prioritization Scheme" registry operates under either of the "IETF Review" or "IESG Approval" policies [[RFC5226](#)] for values between 0x00 and 0xef, with values between 0xf0 and 0xff being reserved for Experimental Use.

New entries in this registry require the following information:

Internet-Draft Extensible Prioritization Scheme for HTTP November 2019

Prioritization Scheme: A name or label for the prioritization scheme.

Code: The 8-bit code assigned to the prioritization scheme.

Specification: A reference to a specification that includes a description of the prioritization scheme.

The entries in the following table are registered by this document.

Prioritization Scheme	Code	Specification
H2_TREE	1	Section 3.2.1
URGENCY	2	Section 3.2.2

[10.](#) References

[10.1.](#) Normative References

[I-D.ietf-quic-http]

Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", [draft-ietf-quic-http-23](#) (work in progress), September 2019.

[I-D.ietf-quic-transport]

Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", [draft-ietf-quic-transport-23](#) (work in progress), September 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [RFC 5226](#), DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.

- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", [RFC 7230](#), DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.

Internet-Draft Extensible Prioritization Scheme for HTTP November 2019

- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", [RFC 7540](#), DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.

[STRUCTURED-HEADERS]

Nottingham, M. and P. Kamp, "Structured Headers for HTTP", [draft-ietf-httpbis-header-structure-14](#) (work in progress), October 2019.

[10.2.](#) Informative References

- [I-D.lassey-priority-setting]
Lassey, B. and L. Pardue, "Declaring Support for HTTP/2 Priorities", [draft-lassey-priority-setting-00](#) (work in progress), July 2019.
- [RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", [BCP 90](#), [RFC 3864](#), DOI 10.17487/RFC3864, September 2004, <<https://www.rfc-editor.org/info/rfc3864>>.
- [RFC7239] Petersson, A. and M. Nilsson, "Forwarded HTTP Extension", [RFC 7239](#), DOI 10.17487/RFC7239, June 2014, <<https://www.rfc-editor.org/info/rfc7239>>.
- [RFC8081] Lilley, C., "The "font" Top-Level Media Type", [RFC 8081](#), DOI 10.17487/RFC8081, February 2017, <<https://www.rfc-editor.org/info/rfc8081>>.
- [RFC8586] Ludin, S., Nottingham, M., and N. Sullivan, "Loop Detection in Content Delivery Networks (CDNs)", [RFC 8586](#), DOI 10.17487/RFC8586, April 2019,

<<https://www.rfc-editor.org/info/rfc8586>>.

10.3. URIs

- [1] <http://tools.ietf.org/agenda/83/slides/slides-83-httpbis-5.pdf>
- [2] <https://github.com/pmeenan/http3-prioritization-proposal>

Appendix A. Acknowledgements

Roy Fielding presented the idea of using a header field for representing priorities in <http://tools.ietf.org/agenda/83/slides/slides-83-httpbis-5.pdf> [1]. In <https://github.com/pmeenan/http3-prioritization-proposal> [2], Patrick Meenan advocates for representing the priorities using a tuple of urgency and concurrency.

Oku & Pardue

Expires May 8, 2020

[Page 18]

Internet-Draft Extensible Prioritization Scheme for HTTP November 2019

The negotiation scheme described in this document is based on [\[I-D.lassey-priority-setting\]](#), authored by Brad Lassey and Lucas Pardue.

The motivation for defining an alternative to HTTP/2 priorities is drawn from discussion within the broad HTTP community. Special thanks to Roberto Peon, Martin Thomson and Netflix for text that was incorporated explicitly in this document.

In addition to the people above, this document owes a lot to the extensive discussion in the HTTP priority design team, consisting of Alan Frindell, Andrew Galloni, Craig Taylor, Ian Swett, Kazuho Oku, Lucas Pardue, Matthew Cox, Mike Bishop, Roberto Peon, Robin Marx, Roy Fielding.

Appendix B. Change Log

B.1. Since [draft-kazuho-httpbis-priority-02](#)

- o Consolidation of the problem statement (#61, #73)
- o Define SETTINGS_PRIORITIES for negotiation (#58, #69)
- o Define PRIORITY_UPDATE frame for HTTP/2 and HTTP/3 (#51)
- o Explain fairness issue and mitigations (#56)

[B.2.](#) Since [draft-kazuho-httpbis-priority-01](#)

- o Explain how reprioritization might be supported.

[B.3.](#) Since [draft-kazuho-httpbis-priority-00](#)

- o Expand urgency levels from 3 to 8.

Authors' Addresses

Kazuho Oku
Fastly

Email: kazuhooku@gmail.com

Lucas Pardue
Cloudflare

Email: lucaspardue.24.7@gmail.com