Workgroup: MASQUE
Internet-Draft:
draft-pardue-masque-dgram-priority-02
Published: 25 July 2022
Intended Status: Experimental
Expires: 26 January 2023
Authors: L. Pardue
Cloudflare

HTTP Datagrams, UDP Proxying, and Extensible Prioritization

Abstract

Application protocols using the QUIC transport protocol rely on streams, and optionally the unreliable datagram extension, to carry application data. Streams and datagrams can be multiplexed in single connections but QUIC does not define an interoperable prioritization scheme or signaling mechanism. The HTTP Extensible Prioritization scheme describes an application-level scheme for the prioritization of streams in HTTP/2 and HTTP/3. This document defines how Extensible Priorities can be augmented to apply to the multiplexing of HTTP datagram flows with other flows or streams.

Note to Readers

RFC EDITOR: please remove this section before publication

Source code and issues list for this draft can be found at https://github.com/LPardue/draft-pardue-masque-dgram-priority.

Status of This Memo

Table of Contents

1. Introduction

Application protocols using the QUIC transport protocol [QUIC] rely on streams, and optionally the unreliable datagram extension [QUIC-DATAGRAM], to carry application data. Streams and datagrams can be multiplexed in single connections but QUIC does not define an interoperable prioritization scheme or signaling mechanism. The HTTP Extensible Prioritization scheme [PRIORITY] describes an application-level scheme for the prioritization of streams in HTTP/2 and HTTP/3. This document defines how Extensible Priorities can be applied to the multiplexing of HTTP datagram [HTTP-DATAGRAM] flows with other flows or streams.

The Extensible Priorities scheme for HTTP describes how clients can send priority signals related to requests in order to suggest how a server allocates resources to serving responses. When the protocol is HTTP/2, responses are carried on streams. When the protocol is HTTP/3, responses are carries on QUIC streams.

While QUIC streams support multiplexing natively via use of a stream identifier, the unreliable datagram extension does not provide any such multiplexing identifier.

HTTP datagrams ([HTTP-DATAGRAM]) defines how multiplexed, potentially unreliable datagrams can be sent inside an HTTP connection. All datagrams are always associated with a request stream. In HTTP/3, HTTP datagrams can map directly to QUIC datagrams, in which case they carry a Quarter Stream ID - an encoding of the request stream ID - that is used to demultiplex at the receiver; see Section 3.1 of [HTTP-DATAGRAM]. [HTTP-DATAGRAM] also defines the DATAGRAM capsule, which can be used for reliable delivery over all versions of HTTP; see Section 3.5 of [HTTP-DATAGRAM]. In all cases, the prioritization of datagrams is noted as unspecified and delegated to future extensions.

This document describes how the Extensible Priorities scheme can be augmented to also apply to HTTP datagrams that are multiplexed with other flows or streams. It enhances the Priority signals sent by clients, with a new datagram-urgency (du) parameter (Section 2.1) and explains how this input is to be considered in server scheduling decisions for HTTP datagrams mapped to QUIC datagrams; see Section 6.

When HTTP datagrams are used for proxying UDP, additional use cases extending beyond UDP data transfer are supported by the use of context IDs; see Section 4 of [HTTP-UDP-PROXY]. The CONTEXT_PRIORITY capsule type can be used to signal the datagram-priority of individual contexts; see Section 4.1.

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The term Integer is imported from [STRUCTURED-FIELDS].

2. Signalling Datagram Priority

The Extensible Prioritization scheme [PRIORITY] describes how clients can send priority signals related to requests. Signals are a set of parameters, encoded using [STRUCTURED-FIELDS].

[PRIORITY] defines the urgency and incremental parameters and provides guidance about how implementers can act on these parameters, in combination with other inputs, to make resource allocation and scheduling choices. Urgency communicates the client-view of request importance, and incremental communicates how the client intends to process response data as it arrives. Parameters are communicated in HTTP headers or version-specific frames. A client omitting the urgency or incremental parameters can be interpreted by the server as a signal to apply default priorities. The core scheme is extensible, new parameters can be defined to augment the base ones.

This specification defines the datagram-urgency (du) extension parameter that operates in addition to the base urgency. There is no extension to the base incremental behavior; individual datagrams, even if belonging to the same identifier, are messages that are expected to be processed individually as they arrive.

2.1. Datagram Urgency

The datagram-urgency (du) parameter is Integer (see Section 3.3.1 of [STRUCTURED-FIELDS]), between 0 and 7, in descending order of priority. This range matches the base urgency (u) parameter range; see Section 4.1 of [PRIORITY]. However, there is no default value.

This parameter indicates the sender's recommendation, based on the expectation that the server would transmit HTTP datagrams in the order of their datagram-urgency values if possible. The smaller the value, the higher the precedence. Omitting the datagram-urgency parameter is a signal to apply the value of the urgency parameter.

The following example shows a request for a CSS file with the urgency set to 0, any associated datagrams have the lower datagram-urgency of 2:

```
:method = GET
:scheme = https
:authority = example.net
:path = /style.css
priority = u=0, du=2
```

Note that when the urgency parameter is omitted, it's default value of 3 is applied. In the following example, the priority field is omitted entirely, invoking the default behaviour of urgency and datagram-urgency, causing them to both have the implicit value 3:

```
:method = GET
:scheme = https
:authority = example.net
:path = /style.css
```

Endpoints MUST NOT treat reception of the datagram-urgency parameter as an error, even if HTTP datagram support is not enabled.

The datagram-urgency parameter applies only to HTTP datagrams mapped to QUIC datagrams. Datagram capsules are sent on streams, so the base urgency parameter applies to them.

3. Reprioritization

Reprioritization behaves similarly to existing mechanisms defined in Section 6 of [PRIORITY]. CONTEXT_PRIORITY frames can be sent by clients to provide updated priority signals after the initial request has been sent.

4. Prioritization when Proxying UDP in HTTP

[HTTP-UDP-PROXY] describes how to proxy UDP using HTTP datagrams. Client make UDP proxying requests using Extended CONNECT, which initiates a UDP tunnel. HTTP datagrams related to this stream correspond to the UDP tunnel by default. In order support extension use cases, Section 4 of [HTTP-UDP-PROXY] defines context IDs, that are sent within datagrams, in addition to the Quarter Stream ID. UDP payloads use context ID 0, forms of data use other IDs.

Datagram priority applies to UDP proxying requests, as described in Section 2.1. By default the same datagram-urgency applies to all HTTP datagram contexts related to the request stream.

4.1. The CONTEXT_PRIORITY Capsule

There might be cases where it is beneficial to prioritize individual contexts differently from one another. This document defines the CONTEXT_PRIORITY (TBD) capsule type to carry a priority signal related to individual contexts.

Once a UDP proxy request converts to the capsule protocol (see Section 3 of [HTTP-UDP-PROXY], clients can send CONTEXT_PRIORITY capsules to signal the priority of the identified context.

A CONTEXT_PRIORITY capsule communicates a complete set of all priority parameters in the Priority Field Value field. Omitting a priority parameter is a signal to derive a value from defaults; see Section 2.1. Failure to parse the Priority Field Value MAY be treated as a connection error. In HTTP/2, the error is of type PROTOCOL_ERROR; in HTTP/3, the error is of type H3_GENERAL_PROTOCOL_ERROR.

TODO: describe what happens if capsules arrive before contexts exists. Buffer? Drop?

TODO: consider if servers could send this capsule type

```
Context Priority Capsule {
    Type (i) = CONTEXT_PRIORITY,
    Length (i),
    Context ID (i),
    Priority Field Value (..),
}
```

Figure 1: CONTEXT_PRIORITY Capsule Format

The CONTEXT_PRIORITY capsule has the following fields:

Context ID: The context ID that is the target of the priority update.

Priority Field Value: The priority update value in ASCII text, encoded using Structured Fields; see [PRIORITY].

5. Client Scheduling

A client MAY use datagram-urgency to make local processing or scheduling choices about HTTP datagrams related to the requests it initiates.

6. Server Scheduling

Priority signals are input to a prioritization process. Expressing priority is only a suggestion. The datagram-urgency parameter introduces new scheduling considerations on top of those presented in Section 10 of [PRIORITY].

It is RECOMMENDED that, when possible, servers respect the datagram-urgency parameter, sending higher-urgency HTTP datagrams before lower-urgency datagrams.

Where streams and datagrams have equal urgency and datagram-urgency respectively, a server needs to decide how to divide the available

sending capacity between stream and datagram data. Strict or static preference for one type of data over another (e.g., datagrams first, then streams) could lead to suboptimal results at the client, depending on the nature of the data. This is a form of starvation, as defined in Section 10 of [PRIORITY]. It applies whether the streams are incremental or not.

Similarly, if datagrams are used for HTTP proxying and there are multiple context IDs in use for different purposes, those purposes might interfere or starve each other if they have the equal datagram-urgency.

It is RECOMMENDED that servers avoid such starvation where possible. The method for doing so is an implementation decision. One approach is to divide the available bandwidth between stream and datagram data in some fixed or dynamic ratio. For instance, a server could choose to generate two classes of application data QUIC packets: STREAM-frame-only packets and DATAGRAM-only-frame packets. The server can control the capacity ratio split by managing the frequency of the packet classes. A simple alternating strategy would result in a roughly 50/50 split, while other frequencies would produce different ratios.

When HTTP datagrams are carried in DATAGRAM capsules. It is RECOMMENDED that servers schedule the capsules in the manner expected for response data; see Section 10 of [PRIORITY].

7. Retransmission Scheduling

Section 12 of [PRIORITY] provides guidance about scheduling of retransmission data vs. new data. Since QUIC datagrams are not retransmitted, endpoints that prioritize QUIC stream retransmission data could delay datagrams. Furthermore, since DATAGRAM capsules are sent as stream data, they **are** subject to retransmission and could also delay native QUIC datagrams.

8. Security Considerations

There are believed to be no additional considerations to those presented in [PRIORITY].

9. IANA Considerations

This specification registers the following entry in the HTTP Priority Parameters Registry

Name: du

Description: The urgency of HTTP datagrams associated with a response.

Reference:
This document

10. References

10.1. Normative References

[HTTP-DATAGRAM] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", Work in Progress, Internet-Draft, draft-ietf-masque-h3-datagram-11, 17 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-h3-datagram-11>.

[HTTP-UDP-PROXY] Schinazi, D., "Proxying UDP in HTTP", Work in Progress, Internet-Draft, draft-ietf-masque-connect-udp-15, 17 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-connect-udp-15>.

[PRIORITY] Oku, K. and L. Pardue, "Extensible Prioritization Scheme for HTTP", RFC 9218, DOI 10.17487/RFC9218, June 2022, <https://www.rfc-editor.org/rfc/rfc9218>.

[QUIC-DATAGRAM] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/rfc/rfc9221>.

[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/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[STRUCTURED-FIELDS] Nottingham, M. and P-H. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/rfc/rfc8941>.

10.2. Informative References

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

Appendix A. Change Log

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

A.1. Since draft-pardue-masque-dgram-priority-01

*Realign with Extensible Priorities RFC

*Realign with latest HTTP datagram and capsule protocol draft

*Add CONTEXT_PRIORITY capsule for prioritizing individual contexts

*Better explain that competing stream and datagram flows should share bandwidth but refrain from requiring any specific ratios.

Acknowledgements

This document is inspired by discussion by many people across HTTP, QUIC and MASQUE WGs.

Author's Address

Lucas Pardue
Cloudflare

Email: lucaspardue.24.7@gmail.com