HTTP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 21, 2018

C. Pratt

D. Thakore
CableLabs
B. Stark
AT&T

March 20, 2018

HTTP Random Access and Live Content
draft-ietf-httpbis-rand-access-live-03

Abstract

To accommodate byte range requests for content that has data appended over time, this document defines semantics that allow a HTTP client and server to perform byte-range GET and HEAD requests that start at an arbitrary byte offset within the representation and ends at an indeterminate offset.

Editorial Note (To be removed by RFC Editor before publication)

Discussion of this draft takes place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org), which is archived at <https://lists.w3.org/Archives/Public/ietf-http-wg/>.

Working Group information can be found at <http://httpwg.github.io/>; source code and issues list for this draft can be found at <https://github.com/httpwg/http-extensions/labels/rand-access-live>.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Some Hypertext Transfer Protocol (HTTP) clients use byte-range requests (Range requests using the "bytes" Range Unit) to transfer select portions of large representations ([RFC7233]). And in some cases large representations require content to be continuously or periodically appended - such as representations consisting of live audio or video sources, blockchain databases, and log files. Clients cannot access the appended/live content using a Range request with the bytes range unit using the currently defined byte-range semantics without accepting performance or behavior sacrifices which are not acceptable for many applications.

For instance, HTTP clients have the ability to access appended content on an indeterminate-length resource by transferring the entire representation from the beginning and continuing to read the appended content as it's made available. Obviously, this is highly inefficient for cases where the representation is large and only the most recently appended content is needed by the client.

Alternatively, clients can also access appended content by sending periodic open-ended bytes Range requests using the last-known end byte position as the range start. Performing low-frequency periodic bytes Range requests in this fashion (polling) introduces latency since the client will necessarily be somewhat behind the aggregated content - mimicking the behavior (and latency) of segmented content representations such as "HTTP Live Streaming" (HLS, [RFC8216]) or "Dynamic Adaptive Streaming over HTTP" (MPEG-DASH, [DASH]). And while performing these Range requests at higher frequency can reduce this latency, it also incurs more processing overhead and HTTP exchanges as many of the requests will return no content - since content is usually aggregated in groups of bytes (e.g. a video frame, audio sample, block, or log entry).

This document describes a usage model for range requests which enables efficient retrieval of representations that are appended to over time by using large values and associated semantics for communicating range end positions. This model allows representations to be progressively delivered by servers as new content is added. It also ensures compatibility with servers and intermediaries that don't support this technique.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

1.2. Notational Conventions

This document cites productions in Augmented Backus-Naur Form (ABNF) productions from [RFC7233], using the notation defined in [RFC5234].

2. Performing Range requests on Random-Access Aggregating ("live") Content

This document recommends a two-step process for accessing resources that have indeterminate length representations.

Two steps are necessary because of limitations with the Range request header fields and the Content-Range response header fields. A server

cannot know from a range request that a client wishes to receive a response that does not have a definite end. More critically, the header fields do not allow the server to signal that a resource has indeterminate length without also providing a fixed portion of the resource.

A client first learns that the resource has a representation of indeterminate length by requesting a range of the resource. The server responds with the range that is available, but indicates that the length of the representation is unknown using the existing Content-Range syntax. See Section 2.1 for details and examples.

Once the client knows the resource has indeterminate length, it can request a range with a very large end position from the resource. The client chooses an explicit end value larger than can be transferred in the foreseeable term. A server which supports range requests of indeterminate length signals its understanding of the client's indeterminate range request by indicating that the range it is providing has a range end that exactly matches the client's requested range end rather than a range that is bounded by what is currently available. See Section 2.2 for details.

2.1. Establishing the Randomly Accessible Byte Range

Establishing if a representation is continuously aggregating ("live") and determining the randomly-accessible byte range can both be determined using the existing definition for an open-ended byte-range request. Specifically, Section 2.1 of [RFC7233] defines a byte-range request of the form:

```
byte-range-spec = first-byte-pos "-" [ last-byte-pos ]
```

which allows a client to send a HEAD request with a first-byte-pos and leave last-byte-pos absent. A server that receives a satisfiable byte-range request (with first-byte-pos smaller than the current representation length) may respond with a 206 status code (Partial Content) with a Content-Range header field indicating the currently satisfiable byte range. For example:

```
HEAD /resource HTTP/1.1
Host: example.com
Range: bytes=0-
```

returns a response of the form:

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 0-1234567/*
```

from the server indicating that (1) the complete representation length is unknown (via the "*" in place of the complete-length field) and (2) that only bytes 0-1234567 were accessable at the time the request was processed by the server. The client can infer from this response that bytes 0-1234567 of the representation can be requested and returned in a timely fashion (the bytes are immediately available).

2.2. Byte-Range Requests Beyond the Randomly Accessible Byte Range

Once a client has determined that a representation has an indeterminate length and established the byte range that can be accessed, it may want to perform a request with a start position within the randomly-accessible content range and an end position at an indefinite "live" point - a point where the byte-range GET request is fulfilled on-demand as the content is aggregated.

For example, for a large video asset, a client may wish to start a content transfer from the video "key" frame immediately before the point of aggregation and continue the content transfer indefinitely as content is aggregated - in order to support low-latency startup of a live video stream.

Unlike a byte-range Range request, a byte-range Content-Range response header field cannot be "open ended", per Section 4.2 of [RFC7233]:

```
byte-content-range  = bytes-unit SP
                       ( byte-range-resp / unsatisfied-range )

byte-range-resp     = byte-range "/" ( complete-length / "*" )
byte-range          = first-byte-pos "-" last-byte-pos
unsatisfied-range   = "*/" complete-length

complete-length     = 1*DIGIT
```

Specifically, last-byte-pos is required in byte-range. So in order to preserve interoperability with existing HTTP clients, servers, proxies, and caches, this document proposes a mechanism for a client to indicate support for handling an indeterminate-length byte-range response, and a mechanism for a server to indicate if/when it's providing a indeterminate-length response.

A client can indicate support for handling indeterminate-length byte-range responses by providing a Very Large Value for the last-byte-pos in the byte-range request. For example, a client can perform a byte-range GET request of the form:

```
GET /resource HTTP/1.1
Host: example.com
Range: bytes=1230000-999999999999
```

where the last-byte-pos in the Request is much larger than the last-byte-pos returned in response to an open-ended byte-range HEAD request, as described above.

In response, a server may indicate that it is supplying a continuously aggregating ("live") response by supplying the client request's last-byte-pos in the Content-Range response header field.

For example:

```
GET /resource HTTP/1.1
Host: example.com
Range: bytes=1230000-999999999999
```

returns

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 1230000-999999999999/*
```

from the server to indicate that the response will start at byte 1230000 and continues indefinitely to include all aggregated content, as it becomes available.

A server that doesn't support or supply a continuously aggregating ("live") response will supply the currently satisfiable byte range, as it would with an open-ended byte request.

For example:

```
GET /resource HTTP/1.1
Host: example.com
Range: bytes=1230000-999999999999
```

will return

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 1230000-1234567/*
```

from the server to indicate that the response will start at byte 1230000 and end at byte 1234567 and will not include any aggregated

content. This is the response expected from a typical HTTP server - one that doesn't support byte-range requests on aggregating content.

A client that doesn't receive a response indicating it is continuously aggregating must use other means to access aggregated content (e.g. periodic byte-range polling).

A server that does return a continuously aggregating ("live") response should return data using chunked transfer coding and not provide a Content-Length header field. A 0-length chunk indicates the end of the transfer, per Section 4.1 of [RFC7230].

3. Other Applications of Random-Access Aggregating Content

3.1. Requests Starting at the Aggregation ("Live") Point

A client that wishes to only receive newly-aggregated portions of a resource (i.e., start at the "live" point), can use a HEAD request to learn what range the server has currently available and initiate an indeterminate-length transfer. For example:

```
HEAD /resource HTTP/1.1
Host: example.com
Range: bytes=0-
```

With the Content-Range response header field indicating the range (or ranges) available. For example:

```
206 Partial Content
Content-Range: bytes 0-1234567/*
```

The client can then issue a request for a range starting at the end value (using a very large value for the end of a range) and receive only new content.

```
GET /resource HTTP/1.1
Host: example.com
Range: bytes=1234567-999999999999
```

with a server returning a Content-Range response indicating that an indeterminate-length response body will be provided

```
206 Partial Content
Content-Range: bytes 1234567-999999999999/*
```

3.2. Shift Buffer Representations

Some representations lend themselves to front-end content removal in addition to aggregation. While still supporting random access, representations of this type have a portion at the beginning (the "0" end) of the randomly-accessible region that become inaccessible over time. Examples of this kind of representation would be an audio-video time-shift buffer or a rolling log file.

For example a Range request containing:

```
HEAD /resource HTTP/1.1
Host: example.com
Range: bytes=0-
```

returns

```
206 Partial Content
Content-Range: bytes 1000000-1234567/*
```

indicating that the first 1000000 bytes were not accessible at the time the HEAD request was processed. Subsequent HEAD requests could return:

```
Content-Range: bytes 1000000-1234567/*

Content-Range: bytes 1010000-1244567/*

Content-Range: bytes 1020000-1254567/*
```

Note though that the difference between the first-byte-pos and last-byte-pos need not be constant.

The client could then follow-up with a GET Range request containing

```
GET /resource HTTP/1.1
Host: example.com
Range: bytes=1020000-999999999999
```

with the server returning

```
206 Partial Content
Content-Range: bytes 1020000-999999999999/*
```

with the response body returning bytes 1020000-1254567 immediately and aggregated ("live") data being returned as the content is aggregated.

A server that doesn't support or supply a continuously aggregating ("live") response will supply the currently satisfiable byte range, as it would with an open-ended byte request.

For example:

```
GET /resource HTTP/1.1
Host: example.com
Range: bytes=0-999999999999
```

will return

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 1020000-1254567/*
```

from the server to indicate that the response will start at byte 1020000, end at byte 1254567, and will not include any aggregated content. This is the response expected from a typical HTTP server - one that doesn't support byte-range requests on aggregating content.

Note that responses to GET requests against shift-buffer representations using Range can be cached by intermediaries, since the Content-Range response header indicates which portion of the representation is being returned in the response body. However GET requests without a Range header cannot be cached since the first byte of the response body can vary from request to request. To ensure Range-less GET requests against shift-buffer representations are not cached, servers hosting a shift-buffer representation should either not return a 200-level response (e.g. sending a 300-level redirect response with a URI that represents the current start of the shift-buffer) or indicate the response is non-cacheable. See HTTP Caching ([RFC7234]) for details on HTTP cache control.

4. IANA Considerations

This document has no actions for IANA.

5. Security Considerations

One potential issue with this recommendation is related to the use of very-large last-byte-pos values. Some client and server implementations may not be prepared to deal with byte position values of 2^^63 and beyond. So in applications where there's no expectation

that the representation will ever exceed 2^^63, a value smaller than this value should be used as the Very Large last-byte-pos in a byte-seek request or content-range response. Also, some implementations (e.g. JavaScript-based clients and servers) are not able to represent all values beyond 2^^53. So similarly, if there's no expectation that a representation will ever exceed 2^^53 bytes, values smaller than this limit should be used for the last-byte-pos in byte-range requests.

6. References

6.1. Normative References

[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>.

[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>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <https://www.rfc-editor.org/info/rfc7233>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

6.2. Informative References

[DASH] ISO, "Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats", ISO/IEC 23009-1:2014, May 2014, <http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC8216] Pantos, R., Ed. and W. May, "HTTP Live Streaming", RFC 8216, DOI 10.17487/RFC8216, August 2017, <https://www.rfc-editor.org/info/rfc8216>.

Acknowledgements

Mark Nottingham, Patrick McManus, Julian Reschke, Remy Lebeau, Rodger Combs, Thorsten Lohmar, Martin Thompson, Adrien de Croy, K. Morgan, Roy T. Fielding, Jeremy Poulter.

Authors' Addresses

Craig Pratt
Portland, OR 97229
US

Email: pratt@acm.org

Darshak Thakore
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
US

Email: d.thakore@cablelabs.com

Barbara Stark
AT&T
Atlanta, GA
US

Email: barbara.stark@att.com