Network Working Group
INTERNET-DRAFT
Intended Status: Informational
Expires: November 25, 2010

W. Zhu
Google, Inc.
M. Jennings

May 24, 2010

Implications of Full-Duplex HTTP
draft-zhu-http-fullduplex-01.txt

Abstract

Full-duplex HTTP follows the basic HTTP request-response semantics but also allows the server to send the response to the client at the same time that the client is transmitting the request to the server. Requirements for Full-duplex HTTP are under-specified in the existing HTTP specification, and this memo intends to clarify the requirements for implementing Full-duplex HTTP under the standard HTTP protocol.

Table of Contents

1 Introduction

HTTP [RFC2616] is a stateless, RPC style protocol which requires communication between client and server follows a strict request-response pattern.

HTTP may also be used to stream data from either client or server. When bi-directional streaming is required, two connections are often used to stream client and server data separately. Using two separate connections not only introduce overhead, but also make HTTP insufficient to be used as a standalone protocol, i.e. application-level protocols are required to handle the two connections.

However, if the server is allowed to send the response to the client at the same time that a request is being transmitted from the client to the server, then effectively full-duplex streaming becomes possible under the standard HTTP protocol [RFC2616].

Full-duplex streaming requires end-to-end support from both the client and server. More specifically, the client has to be explicitly designed to support such capability.

Given the unique properties of full-duplex HTTP, special requirements exist for both the client and server. And those requirements need be identified so that implementations will be able to follow standard behaviors in adopting full-duplex HTTP.

1.1 Terminology

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].

2 Streaming in HTTP

2.1 Request and Response Streaming

Request streaming requires the server to deliver to the application in real-time any streaming data that has been received by the server.

Response streaming requires the server to send to the client in real-time any streaming data that has been produced by the application.

Chunked transfer-encoding [RFC2616] is expected for both request and response streaming. However, client or server should not design the underlying streaming or messaging API based on the chunked transfer-encoding (which is generated hop-by-hop).

Most browsers don't support chunked transfer-encoding for requests.

2.2 Full-Duplex Streaming

When full-duplex streaming is enabled, request and response are transmitted between client and server simultaneously over the same HTTP connection.

Full-duplex streaming may be applied to any resource that is designed to concurrently stream request and response, for example a voice translator.

Full-duplex streaming should still follow the standard request-response semantics and maintain the basic temporal dependency between request and response:

1 A server SHOULD NOT generate any response until it has received a new request. That is, unsolicited server-initiated response is not allowed.

2 A server SHOULD NOT complete the response when the client has not completed the request. That is, a server can only complete the response after it has received the complete request from the client.

3 Protocol Considerations

3.1 Initialization

Full-duplex streaming may be started as soon as the server receives the first byte of the request body. This behavior is significantly different from the technique commonly-known as Hanging GET, in which a separate GET request is issued to initiate streaming to the client.

3.2 Termination

A client terminates request streaming by completing the request, i.e. sending out the last-chunk [RFC2616]. And the server may choose to continue streaming the response after it receives the complete request from the client as decided by the application. Eventually the server terminates response streaming by sending out the last-chunk of response.

Before a server receives the complete request from the client, the only way for the server to terminate response streaming is to close the connection. It is considered an illegal state for an HTTP connection to have a pending request when the response has already

been completed. A client should close the connection immediately if it receives a complete response when it is still streaming the request.

3.3 Persistent Connections

It's important that HTTP keep-alive and pipelining still work with full-duplex HTTP. To achieve that, the client and server should respect the original semantics of HTTP persistent connections [RFC2616].

Request streaming will make the connection unavailable for pipelined requests. Also, continuing response streaming after the request has been completed will prevent any pipelined requests from being processed.

3.4 Time-Out

HTTP server or client may time-out connections while waiting for request or response. Full-duplex HTTP should not override this behavior. A connection may be closed due to time-out when either request or response streaming becomes inactive.

Full-duplex HTTP introduces no special requirements on time-out of the underlying TPC connection. When time-out does happen both request and response streaming will be terminated.

3.5 Proxies

HTTP proxies may not support concurrent responses, and one of the purposes of this document is to increase awareness of full-duplex HTTP communication in proxies.

Proxies may also buffer streamed requests or responses, or have problems to handle chunked transfer-encoding, especially for requests.

3.6 Errors

According to RFC2616 [RFC2616], a client should close the connection if it receives an error response when it is still transmitting the request. Full-duplex HTTP must respect this requirement.

When a server is incapable of streaming response or decides to timeout, it should close the connection. This is also true for the client when it is streaming request.

A client or server should stop streaming any new data after it

notices that the underlying TCP connection has been closed by the other party. In-flight data will be discarded under such a half-close behavior.

4 Application Considerations

4.1 Compatibility

Full-duplex streaming is completely controlled by the server application, and should only be enabled for clients that have been explicitly identified by the server.

It is not sufficient to enable full-duplex HTTP solely based on User-Agent.

For non-controlled client applications, the client needs advise its capability of full-duplex streaming via URL parameters or headers (for example, "X-Accept-Streaming: full-duplex;timeout=30"). Otherwise, full-duplex streaming should be disabled, or the server should return 404 (Not Found) status if full-duplex streaming is mandatory for the requested resource.

4.2 Fall Back

Proxies may disallow early responses, or buffer requests or responses. In such a case, the application may have to switch to a different protocol that uses two connections and rely on polling techniques.

One efficient way to trigger a fall back will be for the client to wait for initial response for a short time-out period. Generally there is no reliable way for a server to distinguish between ill-behaved clients and non-compatible proxies.

4.3 Buffering

Full-duplex HTTP expects minimized buffering from both client and server. However, applications may choose to buffer a certain amount of streaming data for optimization or application specific purpose.

4.4 Messaging

Since chunked transfer-encoding isn't a reliable way to provide message framing, messaging support has to be provided by the application stack, along with any required delivery guarantees.

5 Security Considerations

Full-duplex HTTP introduces no new security concerns beyond those known with regular HTTP communication.

6 IANA Considerations

This document does not require any actions by the IANA.

7 Acknowledgments

Thanks to Henrik Nordstrom, Jamie Lokier, and Mark Nottingham for their feedback when the subject of this document was originally brought up on httpbis.

8 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

Author's Addresses

Wenbo Zhu
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Phone: +1 650 214 5894
Email: wenboz@google.com

Mike Jennings

Email: mike.c.jennings@gmail.com