WIKIPEDIA

HTTP persistent connection

HTTP persistent connection, also called **HTTP keep-alive**, or **HTTP connection reuse**, is the idea of using a single TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair. The newer HTTP/2 protocol uses the same idea and takes it further to allow multiple concurrent requests/responses to be multiplexed over a single connection.

Contents

Operation
- HTTP 1.0
- HTTP 1.1
  - Keepalive with chunked transfer encoding

Advantages

Disadvantages

Use in web browsers

See also

References

External links

Operation

HTTP 1.0

Under HTTP 1.0, connections are not considered persistent unless a keep-alive header is included,[1] although there is no official specification for how keepalive operates. It was, in essence, added to an existing protocol. If the client supports keep-alive, it adds an additional header to the request:

```
Connection: keep-alive
```

Then, when the server receives this request and generates a response, it also adds a header to the response:

```
Connection: keep-alive
```

Following this, the connection is not dropped, but is instead kept open. When the client sends another request, it uses the same connection. This will continue until either the client or the server decides that the conversation is over, and one of them drops the connection.

HTTP 1.1

In HTTP 1.1, all connections are considered persistent unless declared otherwise.[2] The HTTP persistent connections do not use separate keepalive messages, they just allow multiple requests to use a single connection. However, the default connection timeout of Apache httpd 1.3 and 2.0 is as little as 15 seconds[3][4] and just 5 seconds for Apache httpd 2.2 and above.[5][6] The advantage of a short timeout is the ability to deliver multiple components of a web page quickly while not consuming resources to run multiple server processes or threads for too long.[7]

Keepalive with chunked transfer encoding

Keepalive makes it difficult for the client to determine where one response ends and the next response begins, particularly during pipelined HTTP operation.[8] This is a serious problem when `Content-Length` cannot be used due to streaming.[9] To solve this problem, HTTP 1.1 introduced a chunked transfer coding that defines a `last-chunk` bit.[10] The `last-chunk` bit is set at the end of each response so that the client knows where the next response begins.

Advantages

- Reduced latency in subsequent requests (no handshaking).
- Reduced CPU usage and round-trips because of fewer new connections and TLS handshakes.
- Enables HTTP pipelining of requests and responses.
- Reduced network congestion (fewer TCP connections).
- Errors can be reported without the penalty of closing the TCP connection.

According to RFC 7230, section 6.4 (http://tools.ietf.org/html/rfc7230#section-6.4), "a client ought to limit the number of simultaneous open connections that it maintains to a given server". The previous version of the HTTP/1.1 specification stated specific maximum values (http://tools.ietf.org/html/rfc2616#section-8.1.4) but in the words of RFC 7230 "this was found to be impractical for many applications... instead... be conservative when opening multiple connections". These guidelines are intended to improve HTTP response times and avoid congestion. If HTTP pipelining is correctly implemented, there is no performance benefit to be gained from additional connections, while additional connections may cause issues with congestion.[11]

Disadvantages

If the client does not close the connection when all of the data it needs has been received, the resources needed to keep the connection open on the server will be unavailable for other clients. How much this affects the server's availability and how long the resources are unavailable depend on the server's architecture and configuration.

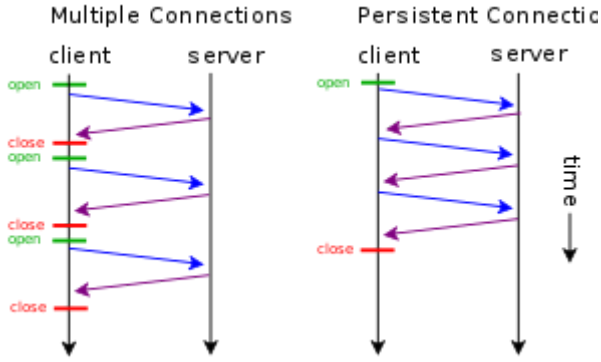
Also a race condition can occur where the client sends a request to the server at the same time that the server closes the TCP connection.[12] A server should send a 408 Request Timeout status code to the client immediately before closing the connection. When a client receives the 408 status code, after having sent the request, it may open a new connection to the server and re-send the request.[13] Not all clients will re-send the request, and many that do will only do so if the request has an idempotent HTTP method.

Use in web browsers

All modern web browsers including Google Chrome, Firefox, Internet Explorer (since 4.01), Opera (since 4.0)[14] and Safari use persistent connections.

By default, Internet Explorer versions 6 and 7 use two persistent connections while version 8 uses six.[15] Persistent connections time out after 60 seconds of inactivity which is changeable via the Windows Registry.[16]

In Firefox, the number of simultaneous connections can be customized (per-server, per-proxy, total). Persistent connections time out after 115 seconds (1.92 minutes) of inactivity which is changeable via the configuration.[17]



Schema of multiple vs. persistent connection.

See also

- HTTP pipelining, whereby multiple requests can be sent without waiting for a response
- HTTP/2, which allows out-of-order pipelining of requests and responses, and also predictive pushing of content before it has been requested

References

1. "The TCP/IP Guide - HTTP Persistent Connection Establishment, Management and Termination" (https://web.archive.org/web/20170521090116/http://www.tcpipguide.com/free/t_HTTPPersistentConnectionEstablishmentManagementand.htm). *www.tcpipguide.com*. Archived from the original (http://www.tcpipguide.com/free/t_HTTPPersistentConnectionEstablishmentManagementand.htm) on 2017-05-21. Retrieved 2017-12-31.
2. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, Persistence (http://tools.ietf.org/html/rfc7230#section-6.3)
3. Apache HTTP Server 1.3 – KeepAliveTimeout Directive (https://httpd.apache.org/docs/1.3/mod/core.html#keepalivetimeout)
4. Apache HTTP Server 2.0 – KeepAliveTimeout Directive (https://httpd.apache.org/docs/2.0/mod/core.html#keepalivetimeout)
5. Apache HTTP Server 2.2 – KeepAliveTimeout Directive (https://httpd.apache.org/docs/2.2/mod/core.html#keepalivetimeout)
6. Apache HTTP Server 2.4 – KeepAliveTimeout Directive (https://httpd.apache.org/docs/2.4/mod/core.html#keepalivetimeout)
7. Multiple (wiki). "Httpd/KeepAlive" (https://web.archive.org/web/20100106140535/http://docforge.com/wiki/Httpd/KeepAlive). *Docforge*. Archived from the original (http://docforge.com/wiki/Httpd/KeepAlive) on January 6, 2010. Retrieved 2010-01-30.
8. "HTTP: What are the relations between pipelining, keep alive and server sent events" (https://stackoverflow.com/questions/17295939/http-what-are-the-relations-between-pipelining-keep-alive-and-server-sent-events).
9. "HTTP Streaming (or Chunked vs Store & Forward)" (https://gist.github.com/CMCDragonkai/6bfade6431e9ffb7fe88).
10. "Chunked Transfer Coding" (http://greenbytes.de/tech/webdav/rfc2616.html#rfc.section.3.6.1).
11. Nielssen, Frystyk Henryk; Gettys, James; Baird-Smith, Anselm; Prud'hommeaux, Eric; Wium Lie, Håkon; Lilley, Chris (October 1997), "Network Performance Effects of HTTP/1.1, CSS1, and PNG" (http://conferences.sigcomm.org/sigcomm/1997/papers/p102.html), *Computer Communication Review*, **27** (4), ISSN 0146-4833 (https://www.worldcat.org/issn/0146-4833)
12. [1] (https://stackoverflow.com/questions/42631273/how-do-browsers-handle-http-keepalive-race-condition)
13. [2] (https://tools.ietf.org/html/rfc7231#section-6.5.7)
14. "Opera 4.0 Upgrades File Exchange: Includes HTTP 1.1" (http://www.opera.com/press/releases/2000/03/28/). Opera Software. 2000-03-28. Retrieved 2009-07-08.

15. "IE8 speeds things up" (http://www.stevesouders.com/blog/2008/03/10/ie8-speeds-things-up/). Stevesouders.com. 2008-03-10. Retrieved 2009-07-17.
16. "How to change the default keep-alive time-out value in Internet Explorer" (http://support.microsoft.com/kb/813827). Microsoft. 2007-10-27. Retrieved 2009-07-17.
17. "Network.http.keep-alive.timeout" (http://kb.mozillazine.org/Network.http.keep-alive.timeout). Mozillazine.org. Retrieved 2009-07-17.

External links

- Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, Connection Management, Persistence (http://tools.ietf.org/html/rfc7230#section-6.3)
- Persistent Connection Behavior of Popular Browsers (dated) (http://www.cs.wisc.edu/~cao/papers/persistent-connection.html)
- Apache HTTPD Keep-Alive Support (http://httpd.apache.org/docs/current/mod/core.html#keepalive)
- Network Performance Effects of HTTP/1.1, CSS1, and PNG (http://conferences.sigcomm.org/sigcomm/1997/papers/p102.html)