

HTTP(超文字傳輸協議, HyperText Transfer Protocol)是網際網路上應用最為廣泛的一種網路協議。所有的WWW檔案都必須遵守這個標準。設計HTTP最初的目的的是為了提供一種釋出和接收HTML頁面的方法。是用於從WWW伺服器傳輸超文字到本地瀏覽器的傳輸協議。預設使用80埠, HTTP客戶端發起一個請求, 建立一個到伺服器指定埠(預設是80埠)的TCP連線。HTTP協議和TCP協議是不衝突的, HTTP定義在七層協議中的應用層, TCP解決的是傳輸層的邏輯。HTTP使用TCP而不是UDP的原因在於(開啟)一個網頁必須傳送很多資料, 而TCP協議提供傳輸控制, 按順序組織資料, 和錯誤糾正。HTTP協議的瓶頸及其優化技巧都是基於TCP協議本身的特性。如TCP建立連線時三次握手有1.5個RTT(round-trip time)的延遲, 為了避免每次請求的都經歷握手帶來的延遲, 應用層會選擇不同策略的http長連結方案。又如TCP在建立連線的初期有慢啟動(slow start)的特性, 所以連線的重用總是比新建連線效能要好。

HTTP連線使用的是“請求—響應”的方式, 不僅在請求時需要先建立連線, 而且需要客戶端向伺服器發出請求後, 伺服器端才能回覆資料。HTTP/1.0是第一個在通訊中指定版本號的HTTP協議版本, 至今仍被廣泛採用, 特別是在代理伺服器中。HTTP/1.1是當前版本, 持久連線被預設採用, 並能很好地配合代理伺服器工作, 還支援以管道方式同時傳送多個請求, 以便降低線路負載, 提高傳輸速度。HTTP/2.0在HTTP 1.x的基礎上, 大幅度的提高了web效能, 減少了網路延遲。HTTP1.0和1.1在之後很長的一段時間內會一直並存, 這是由於網路基礎設施更新緩慢所決定的。

HTTP1.0

HTTP協議老的標準是HTTP/1.0, 為了提高系統的效率, HTTP 1.0規定瀏覽器與伺服器只保持短暫的連線, 瀏覽器的每次請求都需要與伺服器建立一個TCP連線, 伺服器完成請求處理後立即斷開TCP連線, 伺服器不跟蹤每個客戶也不記錄過去的請求。但是, 這也造成了一些效能上的缺陷, 例如, 一個包含有許多影象的網頁檔案中並沒有包含真正的影象資料內容, 而只是指明瞭這些影象的URL地址, 當WEB瀏覽器訪問這個網頁檔案時, 瀏覽器首先要發出針對該網頁檔案的請求, 當瀏覽器解析WEB伺服器返回的該網頁文件中的HTML內容時, 發現其中的影象標籤後, 瀏覽器將根據標籤中的src屬性所指定的URL地址再次向伺服器發出下載影象資料的請求。顯然, 訪問一個包含有許多影象的網頁檔案的整個過程包含了多次請求和響應, 每次請求和響應都需要建立一個單獨的連線, 每次連線只是傳輸一個文件和影象, 上一次和下一次請求完全分離。即使影象檔案都很小, 但是客戶端和伺服器端每次建立和關閉連線卻是一個相對比較費時的過程, 並且會嚴重影響客戶機和伺服器的效能。當一個網頁檔案中包含JavaScript檔案, CSS檔案等內容時, 也會出現類似上述的情況。

同時, 頻寬和延遲也是影響一個網路請求的重要因素。在網路基礎建設已經使得頻寬得到極大的提升的當下, 大部分時候都是延遲在於響應速度。基於此會發現, http1.0被抱怨最多的就是連線無法複用, 和head of line blocking這兩個問題。理解這兩個問題有一個十分重要的前提:客戶端是依據域名來向伺服器建立連線, 一般PC端瀏覽器會針對單個域名的server同時建立6~8個連線, 手機端的連線數則一般控制在4~6個。顯然連線

數並不是越多越好，資源開銷和整體延遲都會隨之增大。連線無法複用會導致每次請求都經歷三次握手和慢啟動。三次握手在高延遲的場景下影響較明顯，慢啟動則對檔案類大請求影響較大。head of line blocking會導致頻寬無法被充分利用，以及後續健康請求被阻塞。

head of line blocking(holb)會導致健康的請求會被不健康的請求影響，而且這種體驗的損耗受網路環境影響，出現隨機且難以監控。為了解決holb帶來的延遲，協議設計者設計了一種新的pipelining機制。pipelining只能適用於http1.1，而且由於使用苛刻，很多瀏覽器廠商並不支援。

HTTP1.1

為了克服HTTP 1.0的這個缺陷，HTTP 1.1支援持久連線(HTTP/1.1的預設模式使用帶流水線的持久連線)，在一個TCP連線上可以傳送多個HTTP請求和響應，減少了建立和關閉連線的消耗和延遲。一個包含有許多影象的網頁檔案的多個請求和應答可以在一個連線中傳輸，但每個單獨的網頁檔案的請求和應答仍然需要使用各自的連線。HTTP 1.1還允許客戶端不用等待上一次請求結果返回，就可以發出下一次請求，但伺服器端必須按照接收到客戶端請求的先後順序依次回送響應結果，以保證客戶端能夠區分出每次請求的響應內容，這樣也顯著地減少了整個下載過程所需要的時間。

在http1.1, request和response頭中都有可能出現一個connection的頭，此header的含義是當client和server通訊時對於長連結如何進行處理。

在http1.1中，client和server都是預設對方支援長連結的，如果client使用http1.1協議，但又不希望使用長連結，則需要在header中指明connection的值為close；如果server方也不想支援長連結，則在response中也需要明確說明connection的值為close。不論request還是response的header中包含了值為close的connection，都表明當前正在使用的tcp連結在當天請求處理完畢後會被斷掉。以後client再進行新的請求時就必須建立新的tcp連結了。

HTTP 1.1在繼承了HTTP 1.0優點的基礎上，也克服了HTTP 1.0的效能問題。HTTP 1.1通過增加更多的請求頭和響應頭來改進和擴充HTTP 1.0的功能。如，HTTP 1.0不支援Host請求頭欄位，WEB瀏覽器無法使用主機頭名來明確表示要訪問伺服器上的哪個WEB站點，這樣就無法使用WEB伺服器在同一個IP地址和埠號上配置多個虛擬WEB站點。在HTTP 1.1中增加Host請求頭欄位後，WEB瀏覽器可以使用主機頭名來明確表示要訪問伺服器上的哪個WEB站點，這才實現了在一臺WEB伺服器上可以在同一個IP地址和埠號上使用不同的主機名來建立多個虛擬WEB站點。HTTP 1.1的持續連線，也需要增加新的請求頭來幫助實現，例如，Connection請求頭的值為Keep-Alive時，客戶端通知伺服器返回本次請求結果後保持連線；Connection請求頭的值為close時，客戶端通知伺服器返回本次請求結果後關閉連線。HTTP 1.1還提供了與身份認證、狀態管理和Cache快取等機制相關的請求頭和響應頭。HTTP/1.0不支援檔案斷點續傳，`RANGE:bytes`是HTTP/1.1新增內容，HTTP/1.0每次傳送檔案都是從檔

案頭開始，即0位元組處開始。`RANGE:bytes=XXXX`表示要求伺服器從檔案XXXX位元組處開始傳送，這就是我們平時所說的斷點續傳！

由上，HTTP/1.1相較於HTTP/1.0協議的區別主要體現在：

- 1 快取處理
- 2 頻寬優化及網路連線的使用
- 3 錯誤通知的管理
- 4 訊息在網路中的傳送
- 5 網際網路地址的維護
- 6 安全性及完整性

常用的請求方式

GET 請求獲取Request-URI所標識的資源

POST 在Request-URI所標識的資源後附加新的資料

HEAD 請求獲取由Request-URI所標識的資源的響應訊息報頭

PUT 請求伺服器儲存一個資源，並用Request-URI作為其標識

DELETE 請求伺服器刪除Request-URI所標識的資源

TRACE 請求伺服器回送收到的請求資訊，主要用於測試或診斷

CONNECT 保留將來使用

OPTIONS 請求查詢伺服器的效能，或者查詢與資源相關的選項和需求

GET方法：在瀏覽器的位址列中輸入網址的方式訪問網頁時，瀏覽器採用GET方法向伺服器獲取資源，POST方法要求被請求伺服器接受附在請求後面的資料，常用於提交表單。GET是用於獲取資料的，POST一般用於將資料發給伺服器之用。

HTTP 1.1狀態程式碼及其含義

狀態程式碼有三位數字組成，第一個數字定義了響應的類別，且有五種可能取值：

1xx：指示資訊—表示請求已接收，繼續處理

2xx: 成功—表示請求已被成功接收、理解、接受

3xx: 重定向—要完成請求必須進行更進一步的操作

4xx: 客戶端錯誤—請求有語法錯誤或請求無法實現

5xx: 伺服器端錯誤—伺服器未能實現合法的請求

使用HTTP2.0測試便可看出HTTP2.0比之前的協議在效能上有很大的提升。下面總結了HTTP2.0協議的幾個特性。

多路複用 (Multiplexing)

多路複用允許同時通過單一的 HTTP/2 連線發起多重的請求-響應訊息。在 HTTP/1.1 協議中瀏覽器客戶端在同一時間，針對同一域名下的請求有一定數量限制。超過限制數目的請求會被阻塞。這也是為何一些站點會有多個靜態資源 CDN 域名的原因之一，拿 Twitter 為例，<http://twimg.com>，目的就是變相的解決瀏覽器針對同一域名的請求限制阻塞問題。而 HTTP/2 的多路複用(Multiplexing) 則允許同時通過單一的 HTTP/2 連線發起多重的請求-響應訊息。因此 HTTP/2 可以很容易的去實現多流並行而不用依賴建立多個 TCP 連線，HTTP/2 把 HTTP 協議通訊的基本單位縮小為一個一個的幀，這些幀對應著邏輯流中的訊息。並行地在同一個 TCP 連線上雙向交換訊息。

二進位制分幀

HTTP/2 在 應用層(HTTP/2)和傳輸層(TCP or UDP)之間增加一個二進位制分幀層。在不改動 HTTP/1.x 的語義、方法、狀態碼、URI 以及首部欄位的情況下，解決了HTTP1.1 的效能限制，改進傳輸效能，實現低延遲和高吞吐量。在二進位制分幀層中，HTTP/2 會將所有傳輸的資訊分割為更小的訊息和幀(frame)，並對它們採用二進位制格式的編碼，其中 HTTP1.x 的首部資訊會被封裝到 HEADER frame，而相應的 Request Body 則封裝到 DATA frame 裡面。

HTTP/2 通訊都在一個連線上完成，這個連線可以承載任意數量的雙向資料流。在過去，HTTP 效能優化的關鍵並不在於高頻寬，而是低延遲。TCP 連線會隨著時間進行自我調諧，起初會限制連線的最大速度，如果資料成功傳輸，會隨著時間的推移提高傳輸的速度。這種調諧則被稱為 TCP 慢啟動。由於這種原因，讓原本就具有突發性和短時性的 HTTP 連線變的十分低效。HTTP/2 通過讓所有資料流共用同一個連線，可以更有效地使用 TCP 連線，讓高頻寬也能真正的服務於 HTTP 的效能提升。

這種單連線多資源的方式，減少服務端的連結壓力，記憶體佔用更少，連線吞吐量更大；而且由於 TCP 連線的減少而使網路擁塞狀況得以改善，同時慢啟動時間的減少，使擁塞和丟包恢復速度更快。

首部壓縮(Header Compression)

HTTP/1.1並不支援 HTTP 首部壓縮，為此 SPDY 和 HTTP/2 應運而生，SPDY 使用的是通用的DEFLATE 演算法，而 HTTP/2 則使用了專門為首部壓縮而設計的 HPACK 演算法。

服務端推送(Server Push)

服務端推送是一種在客戶端請求之前傳送資料的機制。在 HTTP/2 中，伺服器可以對客戶端的一個請求傳送多個響應。Server Push 讓 HTTP1.x 時代使用內嵌資源的優化手段變得沒有意義；如果一個請求是由你的主頁發起的，伺服器很可能會響應主頁內容、logo 以及樣式表，因為它知道客戶端會用到這些東西。這相當於在一個 HTML 文件內集合了所有的資源，不過與之相比，伺服器推送還有一個很大的優勢：可以快取！也讓在遵循同源的情況下，不同頁面之間可以共享快取資源成為可能。