一、HTTP 協議

HTTP[1]是網際網路上應用最為廣泛的網路協議。所有的 www 檔案都必須遵守這個標準。

設計 HTTP 的最初目的是為了提供一種釋出和接收 HTML 頁面的方法用於從 www 伺服器傳輸超文字到本地瀏覽器的傳輸協議，預設為 80 埠，HTTP 客戶端發起一個請求，建立一個到伺服器指定埠的 TCP 連線。HTTP 和 TCP[2]這兩個協議互不衝突，HTTP 定義在七層協議中的**應用層**，TCP 則是**傳輸層**的邏輯。

HTTP 使用 TCP 而非 UDP[3]的原因在於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 協議的瓶頸跟優化技巧都是基於 TCP 本身的特性。HTTP 連線是用「**請求——響應**」的方式，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；**HTTP/1.1** 的持久連線被預設採用，並能很好的配合代理伺服器工作，支援以管道的方式同時傳送多個請求，降低了線路負載，提高傳輸速度；**HTTP/2.0** 大幅度提高 web 的效能，減少網路延緩。

二、HTTP1.0

HTTP 協議中最老的標準是 HTTP/1.0，為了提高系統的效率，它規定了**瀏覽器與伺服器只保持短暫的連線**，瀏覽器的每次請求都要跟伺服器建立一個 TCP 連線，完成請求處理後伺服器就立即斷開 TCP 連線，**不跟蹤客戶也不記錄過去的請求**。

因為只保持了短暫的連線，這造成了**效能上的缺陷**，例如：一個包含有許多影像的網頁檔案中並沒有包含真正的影像資料內容，而只是指明瞭這些影像的 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，發現其中的影像標籤後，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像資料的請求。顯然，訪問一個包含有許多影像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像，上一次和下一次請求完全分離。即使影像檔案都很小，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。

頻寬和延遲也是影響網路請求的重要因素。在網路基礎建設已經使得頻

[1] 超文字傳輸協議，HyperText Transfer Protocol
[2] 傳輸控制協定，Transmission Control Protocol
[3] 用戶數據報協議，User Datagram Protocol

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**。基於此發現，http1.0 被抱怨最多的就是**連線無法重複使用**和 **head of line blocking**[4]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。

為了解決 holb 帶來的延遲，協議設計者又設了一種新的 **pipelining 機制**，但只適用於 http1.1 且使用苛刻，很多瀏覽器不支援。

三、HTTP1.1

跟 HTTP1.0 不同，**HTTP1.1 支援持久連線(**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 請求與響應**，減少了建立和關閉連線的消耗跟延遲。一個包含許多影像的網頁檔案的多個請求與應答可以在一個連線中傳輸，但是**每個單獨的網頁檔案請求和應答仍然需要使用各自的連線**。

HTTP1.1 允許客戶端**不用等待上一次的請求結果返回就可以發出下一個請求**，但是伺服器端必須要按照接收到客戶端請求的**先後順序依次回覆響應結果**，讓客戶端能分出每次的響應內容，減少了下載過程的所需時間。

在 HTTP1.1 中，**request 和 response 的 header 中都有可能出現一個 connection 的 header**，這個的含意是當 client 跟 server 通訊時對於長連結如何進行處理。

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 heade 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 HTTP1.0 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。例如，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

[4] Head of line blocking(holb)會導致健康的請求被不健康的請求影響，而且這種體驗的損耗受到網路環境的影響，出現是隨機的且難以監控。

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，<code>RANGE:bytes</code> 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。<code>RANGE:bytes=XXXX</code> 表示要求伺服器從檔案 XXXX 位元組處開始傳送，這就是斷點續傳。

四、HTTP/1.1 v.s. HTTP/1.0

1. 快取處理
2. 頻寬優化以及網路連線的使用
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. 安全性以及完整性
7. 常用的請求方式：
    a. GET 請求獲取 Request-URI 所標示的資源
    b. POST 在 Request-URI 所標識的資源後附加新的資料
    c. HEAD 請求獲取由 Request-URI 所標識的資源的響應訊息報頭
    d. PUT 請求伺服器儲存一個資源，並用 Request-URI 作為其標識
    e. DELETE 請求伺服器刪除 Request-URI 所標識的資源
    f. TRACE 請求伺服器回送收到的請求資訊，主要用於測試或診斷
    g. CONNECT 保留將來使用
    h. 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

五、HTTP1.1 狀態程式碼及含意

1. 1xx：指示資訊-表示請求已接收，繼續處理
2. 2xx：成功-表示請求已經被成功接收、理解、接受
3. 3xx：重定向-要完成請求必須進行更進一步的操作
4. 4xx：客戶端錯誤-請求有語法錯誤或無法實現
5. 5xx：伺服器端錯誤-伺服器未能實現合法的請求

(詳細請看附錄：狀態程式碼狀態資訊含意)

六、HTTP2.0

使用 HTTP2.o 測試便可看出 HTTP2.0 比之前的協議在效能上有很大的提

升。下面為 HTTP2.0 協議的幾個特性。

1. 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[5]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。

2. 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**。

3. 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

[5] 內容分發網路(Content Delivery Network) 是指一種透過網際網路互相連接的電腦網路系統，利用最靠近每位使用者的伺服器，更快、更可靠地將音樂、圖片、影片、應用程式及其他檔案傳送給使用者，來提供高效能、可擴展性及低成本的網路內容傳遞給使用者。

壓縮而設計的 HPACK 演算法。

4. 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。

七、HTTPS

HTTP 協議傳輸的資料都是未加密的，也就是明文的，因此使用 HTTP 協議傳輸隱私資訊非常不安全。為了保證這些隱私資料能加密傳輸，於是網景公司設計了 SSL（Secure Sockets Layer）協議[6]，從而就誕生 HTTPS。現在的 HTTPS 都是用的 TLS 協議，但是由於 SSL 出現的時間比較早，並且依舊被現在瀏覽器所支援，因此 SSL 依然是 HTTPS 的代名詞。

HTTPS 在傳輸資料之前需要客戶端（瀏覽器）與服務端（網站）之間進行一次握手，在握手過程中將確立雙方加密傳輸資料的密碼資訊。TLS/SSL 協議不僅僅是一套加密傳輸的協議，TLS/SSL 中使用了非對稱加密，對稱加密以及 HASH 演算法。

握手過程的簡單描述如下：

1. 瀏覽器將自己支援的一套加密規則傳送給網站。
2. 網站從中選出一組加密演算法與 HASH 演算法，並將自己的身份資訊以證書的形式發回給瀏覽器。證書裡面包含了網站地址，加密公鑰，以及證書的頒發機構等資訊。
3. 獲得網站證書之後瀏覽器要做以下工作：
   a. 驗證證書的合法性（頒發證書的機構是否合法，證書中包含的網站地址是否與正在訪問的地址一致等），如果證書受信任，則瀏覽器欄裡面會顯示一個小鎖頭，否則會給出證書不受信的提示。
   b. 如果證書受信任，或者是使用者接受了不受信的證書，瀏覽器會生成一串隨機數的密碼，並用證書中提供的公鑰加密。
   c. 使用約定好的 HASH 計算握手訊息，並使用生成的隨機數對訊息進行加密，最後將之前生成的所有資訊傳送給網站。
4. 網站接收瀏覽器發來的資料之後要做以下的操作：
   a. 使用自己的私鑰將資訊解密取出密碼，使用密碼解密瀏覽器發來的握手訊息，並驗證 HASH 是否與瀏覽器發來的一致。
   b. 使用密碼加密一段握手訊息，傳送給瀏覽器。

[6]用於對 HTTP 協議傳輸的資料進行加密。

5. 瀏覽器解密並計算握手訊息的 HASH，如果與服務端發來的 HASH 一致，此時握手過程結束，之後所有的通訊資料將由之前瀏覽器生成的隨機密碼並利用對稱加密演算法進行加密。

這裡瀏覽器與網站互相傳送加密的握手訊息並驗證，**目的是為了保證雙方都獲得了一致的密碼，並且可以正常的加密解密資料**。其中，非對稱加密演算法用於**在握手過程中加密生成的密碼**，對稱加密演算法用於**對真正傳輸的資料進行加密**，而 HASH 演算法用於**驗證資料的完整性**。由於瀏覽器生成的密碼是整個資料加密的關鍵，因此在傳輸的時候使用了非對稱加密演算法對其加密。非對稱加密演算法會生成**公鑰和私鑰**，公鑰只能用於**加密資料**，因此可以隨意傳輸，而網站的私鑰用於**對資料進行解密**，所以網站都會非常小心的保管自己的私鑰，防止洩漏。

TLS 握手過程中如果有任何錯誤，都會使加密連線斷開，從而阻止了隱私資訊的傳輸。正是由於 HTTPS 非常的安全，攻擊者無法從中找到下手的地方，於是更多的是採用了**假證書的手法**來欺騙客戶端，從而獲取明文的資訊。預設 HTTP 的埠號為 80，HTTPS 的埠號為 443。

附錄：狀態程式碼狀態資訊含意

100 Continue 初始的請求已經接受，客戶應當繼續傳送請求的其餘部分。（HTTP 1.1新）
101 Switching Protocols 伺服器將遵從客戶的請求轉換到另外一種協議（HTTP 1.1新）
200 OK 一切正常，對GET和POST請求的應答文件跟在後面。
201 Created 伺服器已經建立了文件，Location頭給出了它的URL。
202 Accepted 已經接受請求，但處理尚未完成。
203 Non-Authoritative Information 文件已經正常地返回，但一些應答頭可能不正確，因為使用的是文件的拷貝（HTTP 1.1新）。

204 No Content 沒有新文件，瀏覽器應該繼續顯示原來的文件。如果使用者定期地重新整理頁面，而Servlet可以確定使用者文件足夠新，這個狀態程式碼是很有用的。
205 Reset Content 沒有新的內容，但瀏覽器應該重置它所顯示的內容。用來強制瀏覽器清除表單輸入內容（HTTP 1.1新）。
206 Partial Content 客戶傳送了一個帶有Range頭的GET請求，伺服器完成了它（HTTP 1.1新）。
300 Multiple Choices 客戶請求的文件可以在多個位置找到，這些位置已經在返回的文件內列出。如果伺服器要提出優先選擇，則應該在Location應答頭指明。

301 Moved Permanently 客戶請求的文件在其他地方，新的URL在Location頭中給出，瀏覽器應該自動地訪問新的URL。
302 Found 類似於301，但新的URL應該被視為臨時性的替代，而不是永久性的。注意，在HTTP1.0中對應的狀態資訊是“Moved Temporatily”。
出現該狀態程式碼時，瀏覽器能夠自動訪問新的URL，因此它是一個很有用的狀態程式碼。

注意這個狀態程式碼有時候可以和301替換使用。例如，如果瀏覽器錯誤地請求http://host/~user（缺少了後面的斜槓），有的伺服器返回301，有的則返回302。
嚴格地說，我們只能假定只有當原來的請求是GET時瀏覽器才會自動重定向。請參見307。

303 See Other 類似於301/302，不同之處在於，如果原來的請求是POST，Location頭指定的重定向目標文件應該通過GET提取（HTTP 1.1新）。
304 Not Modified 客戶端有緩衝的文件併發出了一個條件性的請求（一般是提供If-Modified-Since頭表示客戶只想比指定日期更新的文件）。伺服器告訴客戶，原來緩衝的文件還可以繼續使用。
305 Use Proxy 客戶請求的文件應該通過Location頭所指明的代理伺服器提取（HTTP 1.1新）。

307 Temporary Redirect 和302（Found）相同。許多瀏覽器會錯誤地響應302應答進行重定向，即使原來的請求是POST，即使它實際上只能在POST請求的應答是303時 才能重定向。由於這個原因，HTTP 1.1新增了307，以便更加清除地區分幾個狀態程式碼：當出現303應答時，瀏覽器可以跟隨重定向的GET和POST請求；如果是307應答，則瀏覽器只能跟隨對GET請求的重定向。（HTTP 1.1新）

400 Bad Request 請求出現語法錯誤。

401 Unauthorized 客戶試圖未經授權訪問受密碼保護的頁面。應答中會包含一個WWW-Authenticate頭，瀏覽器據此顯示使用者名稱字/密碼對話方塊，然後在填寫合適的Authorization頭後再次發出請求。

403 Forbidden 資源不可用。伺服器理解客戶的請求，但拒絕處理它。通常由於伺服器上檔案或目錄的許可權設定導致。

404 Not Found 無法找到指定位置的資源。這也是一個常用的應答。

405 Method Not Allowed 請求方法（GET、POST、HEAD、DELETE、PUT、TRACE等）對指定的資源不適用。（HTTP 1.1新）

406 Not Acceptable 指定的資源已經找到，但它的MIME型別和客戶在Accpet頭中所指定的不相容（HTTP 1.1新）。
407 Proxy Authentication Required 類似於401，表示客戶必須先經過代理伺服器的授權。（HTTP 1.1新）
408 Request Timeout 在伺服器許可的等待時間內，客戶一直沒有發出任何請求。客戶可以在以後重複同一請求。（HTTP 1.1新）
409 Conflict 通常和PUT請求有關。由於請求和資源的當前狀態相沖突，因此請求不能成功。（HTTP 1.1新）

410 Gone 所請求的文件已經不再可用，而且伺服器不知道應該重定向到哪一個地址。它和404的不同在於，返回407表示文件永久地離開了指定的位置，而404表示由於未知的原因文件不可用。（HTTP 1.1新）
411 Length Required 伺服器不能處理請求，除非客戶傳送一個Content-Length頭。（HTTP 1.1新）
412 Precondition Failed 請求頭中指定的一些前提條件失敗（HTTP 1.1新）。
413 Request Entity Too Large 目標文件的大小超過伺服器當前願意處理的大小。如果伺服器認為自己能夠稍後再處理該請求，則應該提供一個Retry-After頭（HTTP 1.1新）。

414 Request URI Too Long URI太長（HTTP 1.1新）。

416 Requested Range Not Satisfiable 伺服器不能滿足客戶在請求中指定的Range頭。（HTTP 1.1新）

500 Internal Server Error 伺服器遇到了意料不到的情況，不能完成客戶的請求。

501 Not Implemented 伺服器不支援實現請求所需要的功能。例如，客戶發出了一個伺服器不支援的PUT請求。

502 Bad Gateway 伺服器作為閘道器或者代理時，為了完成請求訪問下一個伺服器，但該伺服器返回了非法的應答。