Fundamentals of HTTP

Fundamentals of HTTP

White Paper

by Patrick Chang

Understanding HTTP and how these headers control behavior of web-based applications can lead to better end-user performance, as well as making it easier to choose an application acceleration solution that addresses the shortcomings of HTTP and browser-based solutions.

HTTP (Hypertext Transfer Protocol) is one of the most ubiquitous protocols on the Internet. It is also one of the few protocols that bridges the gap between networking and application development groups, containing information that is used by both in the delivery and development of web-based applications.

The inner workings of HTTP, particularly the headers used by the client and the server to exchange information regarding state and capabilities, often have an impact on the performance of web-based applications. Understanding HTTP and how these headers control behavior of web-based applications can lead to better end-user performance, as well as making it easier to choose an application acceleration solution that addresses the shortcomings of HTTP and browser-based solutions.

GETting a Web Page

When you open up a browser and request a web page (either by setting a default page or by entering a Uniform Resource Locater or URL), the first thing that happens is that the browser relies upon the operating system to resolve the host name in the URL to an IP address. Normally this is done via a DNS (Domain Name System) query over UDP (User Datagram Protocol) on port 53. However, if the host is listed in the local hosts file, the operating system will not make a DNS query.

When the IP address is obtained, the browser will attempt to open a TCP (Transmission Control Protocol) connection to the web server, usually on port 80. Once the TCP connection is made, the browser will issue an HTTP request to the server using the connection. The request comprises a header section, and possibly a body section (this is where things like POST data go). Once the request is sent, the browser will wait for the response. When the web server has assembled the response, it is sent back to the browser for rendering.

The base request comprises a method, the URI (Uniform Resource Indicator) of the web page or resource being requested, and the HTTP version desired (1.0 or 1.1). The method may be one of:

- Get
- Post
- Put
- Delete
- Head

GET and POST are almost universally supported by web servers, with the difference between them being the way in which query parameters are represented. With the GET method, all query parameters are part of the URI. This restricts the length of the parameters because a URI is generally limited to a set number of characters. Conversely, all parameters are included within the body of the request when using the POST method and there is usually no limit on the length of the body. PUT and DELETE, though considered important for emerging technology architectures such as REST (Representational State Transfer), are considered potentially dangerous as they enable the user to modify resources on the web server. These methods are generally disabled on web servers and not supported by modern web browsers.

The HTTP response consists of a header section and a body. The header section tells the browser how to treat the body content and the browser renders the content for viewing. Each HTTP response includes a status code, which indicates the status of the request. The most common status codes are:

- 200 OK. This indicates success
- 304 Not Modified. This shows that the resource in question has not changed and the browser should load it from its cache instead. This is only used when the browser performs a conditional GET request.
- 404 Not Found. This suggests that the resource requested cannot be found on the server.
- 401 Authorization Required. This indicates that the resource is protected and requires valid credentials before the server can grant access.
- 500 Internal Error. This signifies that the server had a problem processing the request.

While most developers do not need to know these status codes as they are not used within D/HTML, AJAX (Asynchronous Javascript and XML) developers may need to recognize these codes as part of their development efforts.

Most HTTP responses will also contain references to other objects within the body that will cause the browser to automatically request these objects as well. Web pages often contain more than 30 other object references required to complete the page.

TCP controls many performance-related aspects of web applications and is often not manageable by developers or network administrators. Affecting performance by modifying TCP parameters may require the assistance of application delivery controllers or web acceleration solutions, or changing settings in the operating system itself.

Modern browsers render content as it is retrieved, known as progressive rendering, except in the case of Internet Explorer (IE) and table objects. IE will wait for the entire table object to be retrieved before rendering it to the page, which can cause IE to appear to be "slow" when opening a web page. This can often be remedied by adding table-layout:fixed to the style applied to the table in question.

When retrieving these referenced objects, the default browser behavior is to open two TCP connections per host seen in the references. With Internet Explorer there is a Windows registry setting that limits this to a total of eight TCP connections. There is a similar setting in Firefox, but its maximum is 24 TCP connections.

HTTP Headers

HTTP headers carry information about behavior and application state between the browser and the server. These headers can be modified and examined by the browser and the server, as well as intermediary devices such as web acceleration solutions and application delivery controllers. The headers sent by the browser notify the web server of the browser's capabilities. The headers sent by the web server tell the browser how to treat the content.

The most important browser headers, in terms of end-user performance, are:

1. HTTP version (HTTP/1.0 or HTTP/1.1)
2. Accept-Encoding: gzip, deflate
3. Connection: Keep-Alive
4. If-* headers
5. Cache-Control or Pragma no-cache

The first three items are interrelated. HTTP 1.0 does not include compression–indicated by the Accept-Encoding: gzip, deflate header, or connection keep-alives. Compression can reduce the byte count of text by 6:1 to 8:1. This often translates into a 40-50 percent reduction in size for a page. Connection: Keep-Alive will reuse TCP connections for subsequent requests and will save on the latency incurred by the 3-way hand-shake, and 4-way tear-down required for TCP connections on every request. Keeping connections open is important in emerging web-based applications that utilize Web 2.0 technology such as AJAX (Asynchronous JavaScript and XML) to perform real-time updates of content because it reduces the overhead associated with opening and closing TCP connections.

The various If-* headers, such as If-Modified-Since, will enable the web server to send a response that indicates the content has not been modified if this is true. This can potentially turn a 200KB download into a 1KB download, as the browser will respond to the 304 Not Modified response by loading the referenced content from the browser's cache. However, a lot of If-* requests for static content can result in unnecessary round trips. This can really slow end-user performance. The no-cache header and its relatives—no-store, private, must-revalidate, and proxy-revalidate—request that proxies and, sometimes, web servers not cache the response to the request. Honoring those requests can cause the servers to do a lot more work because they must always return the full content rather than enable the browser to use a cached version.

The most important web server headers, in terms of end-user performance, are:

1. The HTTP version (either HTTP/1.0 or HTTP/1.1) at the beginning of the status line
2. Connection: Keep-Alive/Close
3. Encoding: gzip, deflate
4. The various cache-control headers, especially max-age
5. Content-Type:
6. Date:
7. Accept-Ranges: bytes

Again, the first three items are inter-related and are meant to impart the same information as when sent by the browser. The cache-control headers are very important because they can be used to store items in the browser cache and avoid future HTTP requests altogether. However, using cached data runs the risk of using out-dated data if the content changes before the cached object expires. Content-type is important for telling the browser how to handle the object. This is most important for content that the browser hands off to plug-ins (Flash, Microsoft Office documents, etc.). It is also the biggest clue to the true function of that object in the web application. Improper content types will often result in slower, but not broken web applications. The Date header is very important because it affects how the browser interprets the cache-control headers. It is important to make sure the date on the server is set correctly so that this field is accurate. The Accept-Ranges header is only important when downloading PDF documents. It enables the browser to know that it can request the PDF document one page at a time.

F5's BIG-IP WebAccelerator employs a set of technologies collectively called Intelligent Browser Referencing (IBR) that make the use of the browser's cache and TCP connections more efficient, often dramatically improving end-user performance.

Cookies

Cookies are sent by the web server to the browser as an HTTP header and used to store all sorts of information about a user's interaction with the site. Generally speaking the use of cookies will not affect the performance of an application, unless they are encrypted for security purposes. The reason encrypted cookies can affect performance is because the web server needs to decrypt them before use, and the encryption/decryption process is resource intensive. The more encrypted cookies that are used by a site, the longer it takes for the web server to process them into a readable format.

Meta Tags

The HTML standard allows the inclusion of meta tags within the HEAD element of an HTML page. There are two types of meta tags: HTTP-EQUIV and NAME. HTTP-EQUIV meta tags are equivalent to HTTP headers. These meta tags can conflict with–and even contradict—the HTTP headers sent by the browser or web server. This is problematic because meta tags will take precedence. In many cases, HTML coders will use meta tags to provide web page functionality without realizing what the meta tags do to the inner workings of the browser such as cache behavior. The two meta tags that cause the most problems with web application performance are the no-cache and refresh tags. The no-cache tag instructs the browser to not cache the object that contains the meta tag. This forces the browser to always get a full download of that object, even if it has not changed. The refresh tag is often used to mimic an HTTP 302 redirect response. The problem is that the refresh tag tells the browser to override the browser's cache settings and revalidate every object referenced by the refresh tag.

F5 BIG-IP Local Traffic Manager can act as a cookie gateway and perform cookie encryption/decryption. It can also improve the performance of encryption/decryption for cookies as well as secure traffic (HTTPS) due to acceleration technology.

Conclusion

There are many more headers and settings involved in HTTP, but these are the ones that can affect the performance of HTTP the most. Being aware of how HTTP and its headers interact between the browser and the server can not only help developers and network professionals improve the end-user experience, it can also provide invaluable information when troubleshooting particularly slow sites and applications.

Web application acceleration solutions can also act to improve the end-user experience by using the many HTTP headers and browser options available to ensure optimal performance. These solutions are often preferred over making changes to the application itself because they are less invasive and include additional protocol layer (TCP) enhancements and optimizations that improve the overall delivery of applications.

F5 Networks, Inc.
401 Elliott Avenue West, Seattle, WA 98119
888-882-4447 www.f5.com

Americas
info@f5.com

Asia-Pacific
apacinfo@f5.com

Europe/Middle-East/Africa
emeainfo@f5.com

Japan
f5j-info@f5.com