HTTP/2

Im Vergleich mit HTTP/1.1, wo liegen die Unterschiede

Nach dem Studium unzähliger Beschreibungen/Artikel über die Vorzüge von HTTP/2 gegenüber HTTP/1.1 kann ich mich nun nicht mehr zurückhalten, denn es wurden unzählige falsche Behauptungen aufgestellt und die Autoren ließen mich ihre völlige Unkenntnis des Materie erkennen.

Ich möchte hier nicht alle Behauptungen aufzählen wie parallele/schnellere Datenübertragung, natives HTTPS, abwärtskompatibel … und auch die Nachteile von HTTP/1.1, offenbar haben die Autoren sich noch nie richtig mit der Materie befasst oder ihnen fehlt einfach das Verständnis dafür.

HTTPS - Hypertext Transfer Protocol Secure

HTTPS ist eigentlich kein separates Protokoll, auch wenn es häufig so beschrieben wird, es bedeutet lediglich das über eine verschlüsselte Verbindung das HTTP Protokoll genutzt wird.

Der Client (Browser) erkennt über das Schema (Scheme) also http:// (unverschlüsselt) oder https:// (verschlüsselt) eines Links, ob er eine Verbindung zur Domain (Host) über eine unverschlüsselte Verbindung (Port 80) oder eine verschlüsselte Verbindung (Port 443) aufbauen soll.
Nach der Namensauflösung (Ermittlung der IP zur aufzurufenden Domain, über den DNSServer (Domain Name System) wird zur ermittelten IP und dem vorgegebenen Port eine Verbindung aufgebaut. Wird eine unverschlüsselte Verbindung (socket connection) angefragt, kann die Verarbeitung der HTTP Informationen erfolgen, oft jedoch führt die Konfiguration auf dem Server allerdings zu einer Weiterleitung (redirect) auf die verschlüsselte Verbindung.
Wird eine verschlüsselte Verbindung angefragt kommt es zum Austausch von Informationen zwischen Client und Server, es wird unter anderem geklärt welche TLS Version (Transport Layer Security) verwendet werden soll, der Vorgänger SSL (Secure Sockets Layer) sollte nicht mehr genutzt werden, bei richtiger Server-Konfiguration sollten auch eine vom Server bevorzugte Sammlung kryptographischer Verfahren (Cipher Suite) genutzt werden. Der Server sendet sein Zertifikat an den Client und dieser sollte es überprüfen, zumindest ob das Zertifikat für die angefragte Domain signiert wurde und ob es noch gültig ist.

Werden mehrere Zertifikate auf einer IP unterstützt ist Server Name Indication (SNI) erforderlich, was bedeutet das während des Verbindungsaufbaus auch noch überprüft werden muss welches Zertifikat ausgeliefert bzw. genutzt werden muss.

Bis zu diesem Punkt existiert beim Verbindungsaufbau kein Unterschied zwischen HTTP/1.1 und HTTP/2 !

Neu kommt nun das Application Layer Protocol Negotiation (ALPN) hinzu, also eine Überprüfung der unterstützten Protokolle, unterstützen beide Enden der Verbindung HTTP/2 kann eine Umschaltung des Protokolls vorgenommen werden, es ist somit allerdings auch möglich einen Server nur für HTTP/2 zu realisieren.
Dieses passiert noch während des Verbindungsaufbaus, bevor auch nur eine Zeile über ein HTTP Protokoll gelaufen ist!

Ist der Verbindungsaufbau abgeschlossen besteht eine Verbindung (Socket Connection) in kommende und gehende Richtung.

HTTP/1.1 - Hypertext Transfer Protocol Version 1.1

Nach geglückten Verbindungsaufbau schickt der Client eine Anfrage (Request) an den Server und wartet auf seine Antwort, diese Verbindung ist somit belegt. Möchte der Client weitere Anfragen stellen muss er darauf warten das seine vorherige Anfrage beantwortet wurde (Head-of-line blocking) oder eine weitere Verbindung aufbauen, beim erneuten Verbindungsaufbau entfällt die benötigte Zeit für die Namensauflösung der restliche Aufwand ist identisch.

Je nach Browser kann eine unterschiedliche Anzahl von parallelen Verbindungen zum Server aufgebaut werden, die Werte sind in den vergangenen Jahren immer weiter gestiegen.

Parallele Verbindung bedeutet aber auch gleichzeitige Übermittlung von Daten!

Die Methode (GET, POST, HEAD), die relative Adresse, das Protokoll und die Header Fields werden im Klartext übertragen, die Antwort enthält den Status inklusive der Header Fields ebenfalls im Klartext, gefolgt von den Daten. Dies ist sicherlich eine ziemlich vereinfachte Darstellung sie soll an dieser Stelle allerdings reichen.

HTTP/2 - Hypertext Transfer Protocol Version 2

Nach geglückten Verbindungsaufbau schickt der Client eine Begrüßung womit die eigentliche HTTP/2 Verbindung steht, es folgt ein Austausch der Konfigurationen, bezüglich Übertragungsblockgröße (frame size) und ein paar andere Werte, gefolgt von ein paar weiteren Frames die für die Priorisierung erforderlich sind, wurde vom Server die Konfiguration bestätigt kann die erste Anfrage erfolgen.

Das Protokoll ist inhaltlich gleich zum älteren Standard also gleiche Methoden und Antwort-Codes sowie überwiegend identische Header Felder, allerdings werden diese Informationen (HPack) komprimiert und somit ist die Datenmenge gerade bei den folgenden Anfragen geringer. Allerdings müssen hierfür Listen (dictionaries) für beide Richtungen gepflegt werden um ein Entpacken zu ermöglichen.

Im Regelfall hat der Client erst einmal nur die eigentlich aufzurufende Seite und muss auf die Antwort warten, beim Verarbeiten der Antwort ist es jedoch möglich quasi beim auftreten einer neuen Anforderung sofort eine neue Anfrage (request) abzusetzen.

Um die Wartezeiten in der Verbindung zu füllen besteht durch das neue Leistungsmerkmal Push nun auch die Möglichkeit für den Server die Initiative zu übernehmen und Daten zum Client zu senden ohne das sie angefordert wurden, das macht natürlich nur dann Sinn wenn er weiß was der Client anschließend benötigt.

HTTP/2 unterstützt nur noch eine Verbindung, parallele Datenübertragung ist somit nicht mehr möglich, damit mehrere Anfragen in einer Verbindung beantwortet werden können braucht es nun ein Frame-Konzept in dessen Header wiederum die Zugehörigkeit zur Anfrage (request) zugeordnet ist. Diese Frames können auch gemischt (multiplext) übertragen werden, durch eine Priorisierung durch den Client ist es auch möglich, nach Versendung einer Anfrage, noch einmal dem Server mitzuteilen welche Daten bevorzugt benötigt werden.

Ein Server als auch ein Client kann das HTTP/2 Protokoll anwenden ohne es vollständig zu unterstützen, er muss z.B. die Priorisierung und Push nicht umsetzen!

Zu klären bzw. zu untersuchen wäre:

- die Umsetzung / Nutzung / Einsatz der Priorisierung in den Browsern (Clients) sowohl als auch in den Severn
- die Umsetzung / Nutzung / Einsatz von Push auf Servern inCMS
- welchen Einfluss hat die Bandbreite bei HTTP/1.1 bzw. HTTP/2 Verbindungen
- Ladeergebnisse in unterschiedlichen Browsern / Geräten und Websites

Einfluss auf die Nutzung der Bandbreite

Da bei HTTP/2 bereits während der Auswertung des HTML-Dokumentes sofort über die bestehende Verbindung alle fehlende Inhalte nachgefordert werden können und nicht nur eine Anfrage und dann für weitere Anfragen neue Verbindungen zur Verfügung stehen müssen, nutzt HTTP/2 die Bandbreite der Connection früher aus als HTTP/1.1. Denn die Angeforderten Inhalte können dann ohne Wartezeiten aneinandergereiht vom Server ausgeliefert werden.

HTTP/2 Priority

Wünschenswert wäre das die Browser/Clients eine Priorisierung der Daten die für den aktuell sichtbaren Bereich erforderlich sind vornehmen, diese scheint allerdings nicht so zu sein, denn bei meinen Tests war die folgend Reihenfolge von Dateitypen ausschlaggebend 1. HTML, 2. CSS, 3. JS und anschließend Bild/Video Daten, die Wichtigkeit für das aktuelle Rendern im View schien nicht von Bedeutung zu sein.

Ein Beispiel: setzt man am Fuß der Seite ein Video Tag mit dem Attribut preload Wert auto wird es geladen, ohne Rücksicht auf den Slider mit nachzuladenden Bildern im Kopf der Seite der sich im Browser gerade im aktuellen Blickfeld befindet!

Da ein Browser auch nicht ermitteln kann ob ein JavaScript erforderlich ist und ob es sofort Auswirkungen aufs Rendering hat ist diese Reihenfolge allerdings auch nachvollziehbar.

HTTP/2 Preload Header

Für den HTTP/2 Preload Header existieren bereits Implementierungen, bei dieser Variante werden dem Client über den Header benötigte Dateien mitgeteilt und ihm nahegelegt diese zu laden, dadurch entsteht der zeitlicher Vorsprung bis die benötigte Datei im Inhalt beim Rendern der Seite gefunden wird. Falls die beim Client schon existiert kann dieser an Hand der Caching Informationen immer noch selbstständig entscheiden ob er sie benötigt. Es empfiehlt sich allerdings nicht alle möglichen Dateien über diesen Weg zur Verfügung zu stellen da es hierdurch auch zu einem erhöhten Analyseaufwand kommen kann und dadurch die Verarbeitungsgeschwindigkeit beeinträchtigt wird.

```
<FilesMatch "index.html">
  Header add Link "</css/styles.css>; rel=preload; as=style"
  Header add Link "</js/scripts.js>; rel=preload; as=script"
</FilesMatch>
```

```
<?php
header("Link: </css/styles.css>; rel=preload; as=style, </js/scripts.js>;
rel=preload; as=script");
```

Cache Digests for HTTP/2

Diese Methode ermöglicht dem Client den Server über den Inhalt seines Caches zu informieren. Diese

Informationen kann der Server dann verwenden um zu Entscheiden was er über Push oder Preload zur Verfügung stellen möchte. Weitere Informationen hier. (https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-cache-digest-05)

Resümee

- Time to first byte (TTFB): Die Antwort auf die erste Anfrage trifft in der Regel bei HTTP/2 später ein, das liegt an der zusätzlichen Kommunikation zwischen Client und Server (ALPN, Begrüßungstext, Konfigurationsaustausch und ein paar leerer Frames vor der ersten Anfrage).
- HTTP/1.1 kann unter Umständen durch parallele Datenübertragung gleichzeitig ein höheres Datenvolumen transferieren, wird allerdings nach dem Erreichen der maximalen Anzahl an parallelen Verbindungen ausgebremst (Head-of-line blocking).
- HTTP/2 kann durch gleichzeitiges Senden und Empfangen weitere Anfragen senden ohne warten zu müssen, bekommt dann allerdings nur nacheinander die Antworten.
- Leider muss die Priorisierung unter HTTP/2 für die Anzeige auch nicht unbedingt von Vorteil sein.

Autor: Udo Schmal, veröffentlicht: 30.04.2019, letzte Änderung: 29.08.2024

© Copyright 2024 Udo Schmal