WiSe 2019/20
Big Data Management und Analytics in datenzentrischen Wissenschaften

Main improvements of http2 compared to http

by Peter Bahlmann

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

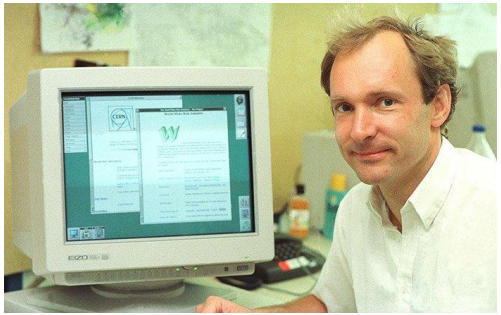
Programm

1. Definition HTTP
2. Funktionsweise HTTP/1.X
3. Verbesserungen von HTTP/2
4. Zusätzliche Informationen
5. Fragen

Definition

HTTP - was ist das ?

- Hypertext Transfer Protocol
- zustandloses Protokoll in OSI Layer 7
- zur Übertragung von Daten in einem Netzwerk
- vor allem zum Laden von Webseiten genutzt
- erste Version, HTTP 0.9, von u.a. Tim Berners-Lee am CERN 1991 entwickelt



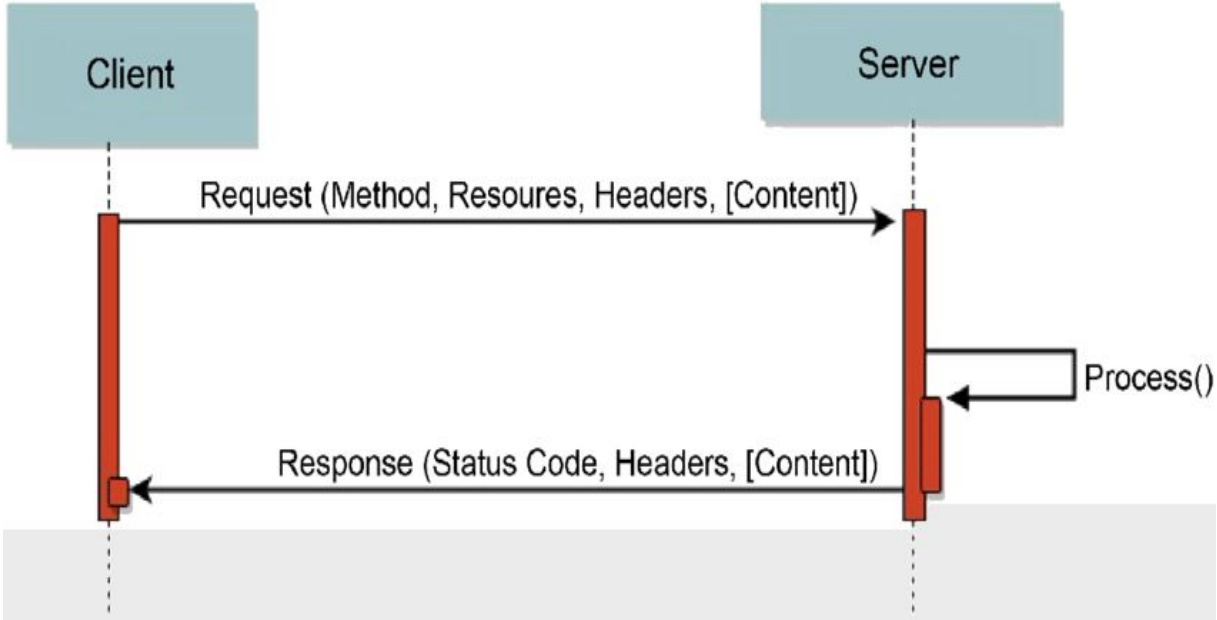
HTTP - Standards

- 1996: RFC 1945 -> HTTP/1.0
- 1999: RFC 2616 -> HTTP/1.1
- 2015: RFC 7540/1 -> HTTP/2

Funktionsweise HTTP/1.X

C/S

- Client (z.B. Browser)
  sendet Request
- Server (z.B. Webserver)
  antwortet mit Response
- TCP Connection
- Nachrichten bestehen aus Header und Body



Quelle: https://www.researchgate.net/figure/HTTP-request-response-pattern_fig1_308387942

Request Header

- Start Line
- Header
- Leerzeile
- Body (optional)

```
GET /request-und-response-18.html HTTP/1.1
Host: www.coder-welten.de
User-Agent: Mozilla/5.0 (weitere Angaben ...
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Response Header

- Status Line
- Header
- Leerzeile
- Body (z.B. bei Status Code 201, 204 generell kein Body)
- ansonsten ist im Body die gewünschte Ressource

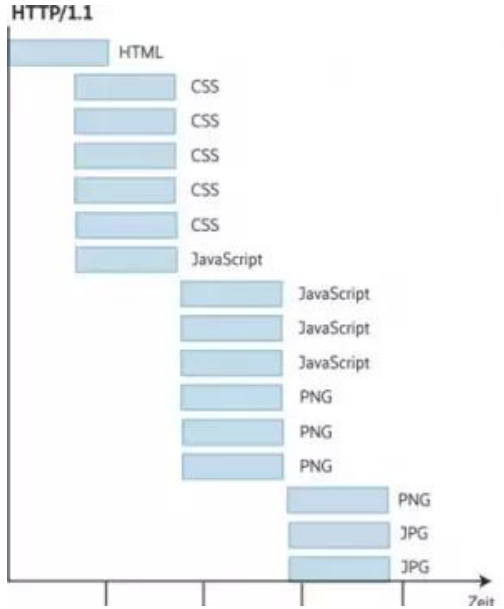
```
HTTP/1.1 200 OK
Date: Mon, 11 Mar 2013 11:17:09 GMT
Server: Apache
X-Powered-By: PHP/5.3.8
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 832
Keep-Alive: timeout=1, max=100
Connection: Keep-Alive
Content-Type: text/htm
```

Verbesserungen von HTTP/2

Multiplexing

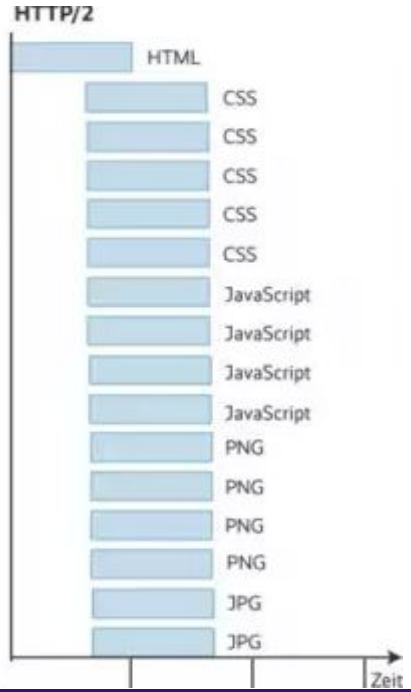
Multiplexing

- HTTP/1.1 erlaubt nur einen Request pro TCP-Verbindung
- Im Browser 6 Verbindungen gleichzeitig
- übrige Requests warten
- => Head of line Blocking



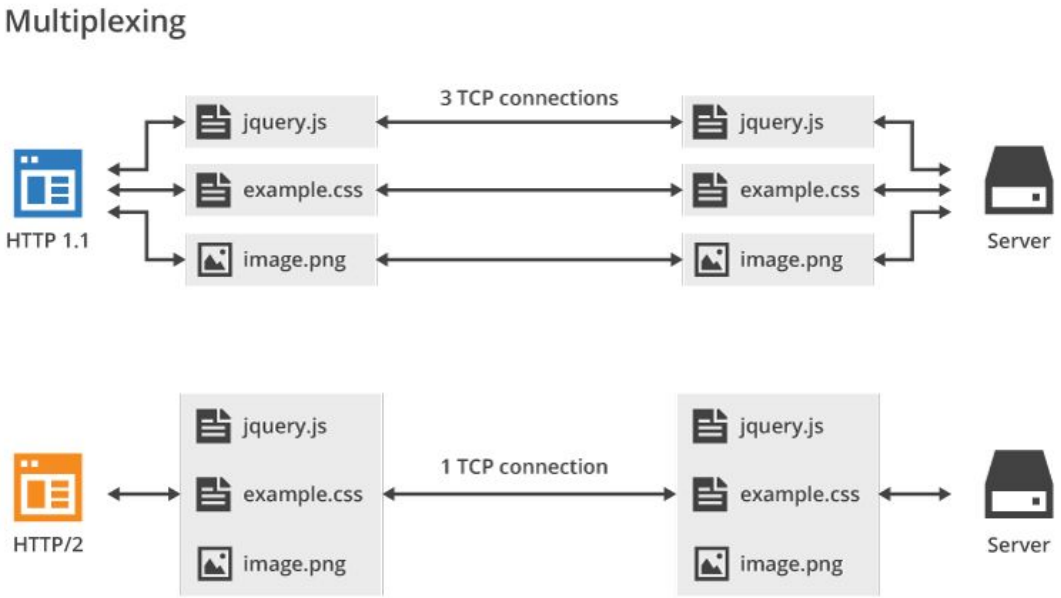
Multiplexing

- Alle Ressourcen werden über eine Verbindung geladen
- Browser kann beliebig viele Requests stellen



Multiplexing

- Nutzung der maximalen Bandbreite, da Bündelung der Request (nur 1 SSL Aufbau)
- Schnellere Ladezeiten



Binary framing Layer

Binary framing Layer

- HTTP/1.1 -> Text Protokoll
- HTTP/2 -> binäres Protokoll
  - weniger fehleranfällig, da u.a. kein Leerraum oder Zeilenende
  - effizienter zu parsen
- Kommunikation findet in kleinen Nachrichten und Frames statt, alle im Binärformat

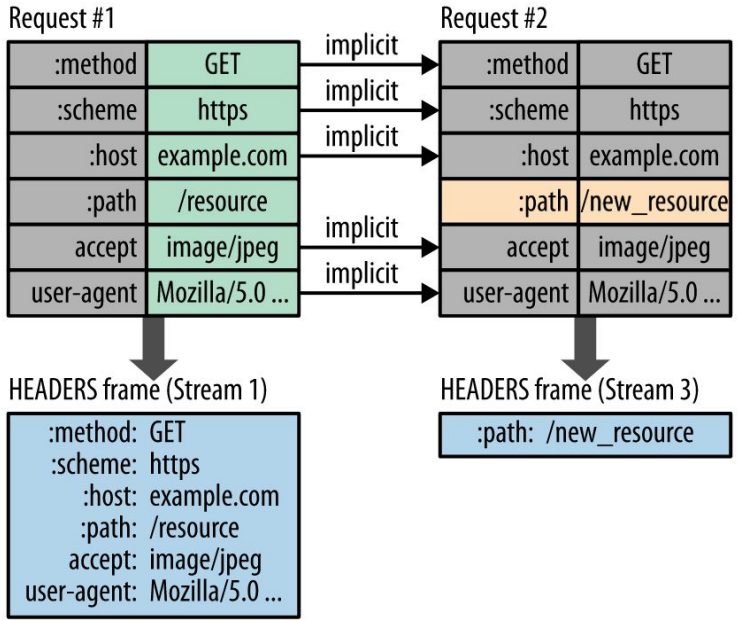
Header compression

Header compression

- Metainformationen (z.B. Version, Größe d. Ressource) bei jedem Request im HTTP Header
- bei HTTP/1.1 Header bei jedem Request vollständig und unkomprimiert
- 500–800 bytes of overhead pro transfer
- bei HTTP/2:
  - HPACK Algorithmus komprimiert Header durch Hufman Codierung
  - um ca. 88% komprimiert

Header compression

- Client und Server führen Index-Liste von vorherigen übertragenden Werten
- dadurch ist es mit der Hufman Codierung möglich einzelne Werte bei der Übertragung zu komprimieren und danach zu rekonstruieren



Prioritization

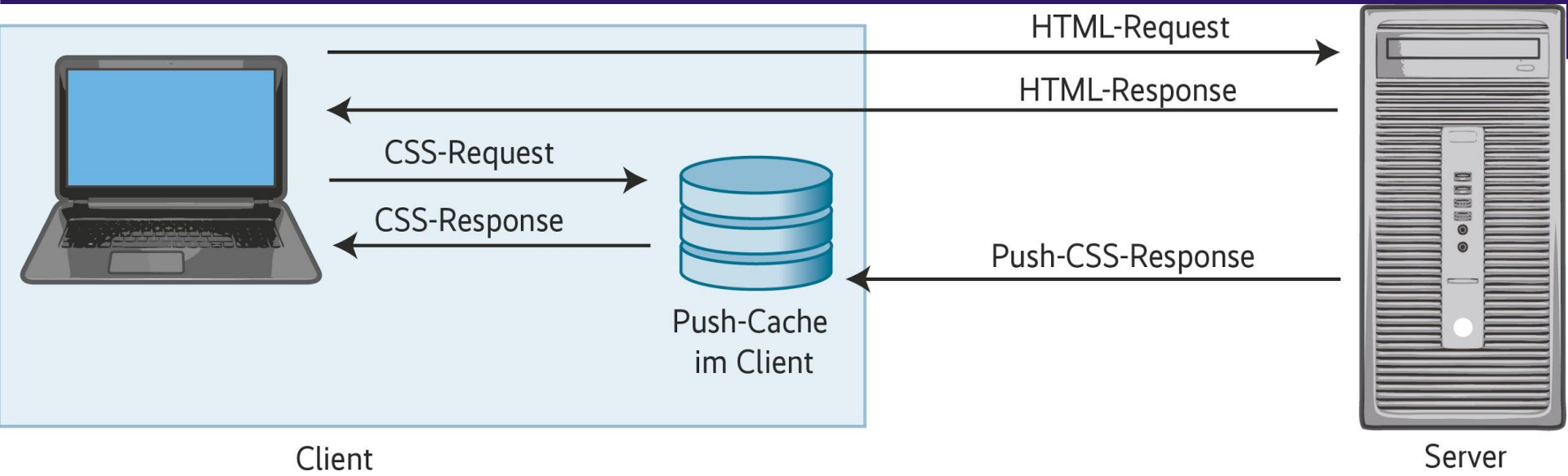
Prioritization

- HTTP/1.1 : Browser analysiert HTML um referenzierte Ressource zu priorisieren
- dadurch entsteht Reihenfolge von Requests

- bei HTTP/2 nicht möglich, da Ressourcen parallel über eine Verbindung angefragt werden
- Client teilt bei Request Server mit, ob Ressource abhängig ist, sowie eine Gewichtung, um die Bandbreite bestmöglich zuzuteilen
- Server entscheidet wie er die Response händelt

Server push

Server push



- Server sendet auf bestehenden Verbindung Ressourcen, ohne dass Client angefragt hat
- aktives Ressourcen senden, die vemutlich noch benötigt werden

Server push

HTTP/2 löst sich von der strengen Request/Response Semantik und ermöglicht on-to-many und server-initiated push workflows, die eine Welt neuer Interaktionsmöglichkeiten bietet.

Zusammenfassung

Zusammenfassung

- nur eine Verbindung nötig um alle Ressourcen zu laden
- binäres Protokoll, Übertragung der Daten in Frames
- Header Komprimierung mit Hilfe von HPACK
- Server Push -> Lockerung der Request/Response Mechanik
- feine Priorisierung der Requests
- erhöhte Sicherheit (TLS benötigt)
- Suchmaschinenoptimierung durch Geschwindigkeit (besseres Rating)

Zusätzliche Infos

HTTP - Fakten

HTTP - Versionskontrolle in der Chrome Console:

console.log(performance.getEntries()[0].nextHopProtocol)

Speed Test:

https://www.youtube.com/watch?v=QCEid2WCszM&feature=emb_logo

Usage statistics of HTTP/2 for websites:

https://w3techs.com/technologies/details/ce-http2

HTTP/3

- Im November 2018 hat die Internet Engineering Task Force (IETF) einen neuen Internet Entwurf verabschiedet
- QUIC-Transportprotokoll (by Google) -> HTTP/3
- Setzt auf UDP
- Chrome Implementierung seit Dezember 2019

HTTP - Methoden

Ausgewählte Methoden:

GET - fordert Daten von einer bestimmten Quelle (URL) an
POST - übergibt die zu verarbeitenden Daten an eine Quelle (kein Caching)
HEAD - Header-Infos abrufen, die mit URL verknüpft sind
PUT - Daten für bestimmte URL durch neue, vom Client gesendeten, Daten ersetzen
DELETE - Daten hinter der jeweiligen URL löschen

www.htw-berlin.de

Quellen

https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages
https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html
https://developers.google.com/web/fundamentals/performance/http2
https://www.golem.de/news/ietf-http-ueber-quic-wird-zu-http-3-1811-137655.html
https://www.heise.de/select/ix/2018/2/1517616631468499
https://www.heise.de/tipps-tricks/HTTP-was-ist-das-4607102.html#HTTP_was%20ist%20das
https://de.w3docs.com/html-lernen/http-anfragemethoden.html
https://www.mediaevent.de/tutorial/http-request.html
https://www.elektronik-kompendium.de/sites/net/0902231.htm
http://www.coder-welten.de/glossar/request-und-response-18.html
Letzter Zugriff: 23.02.2020 20:00 Uhr