Optimierung im Web

Connor Gäde

12.01.2017

Inhaltsverzeichnis

Einführung Web

Einführung Web

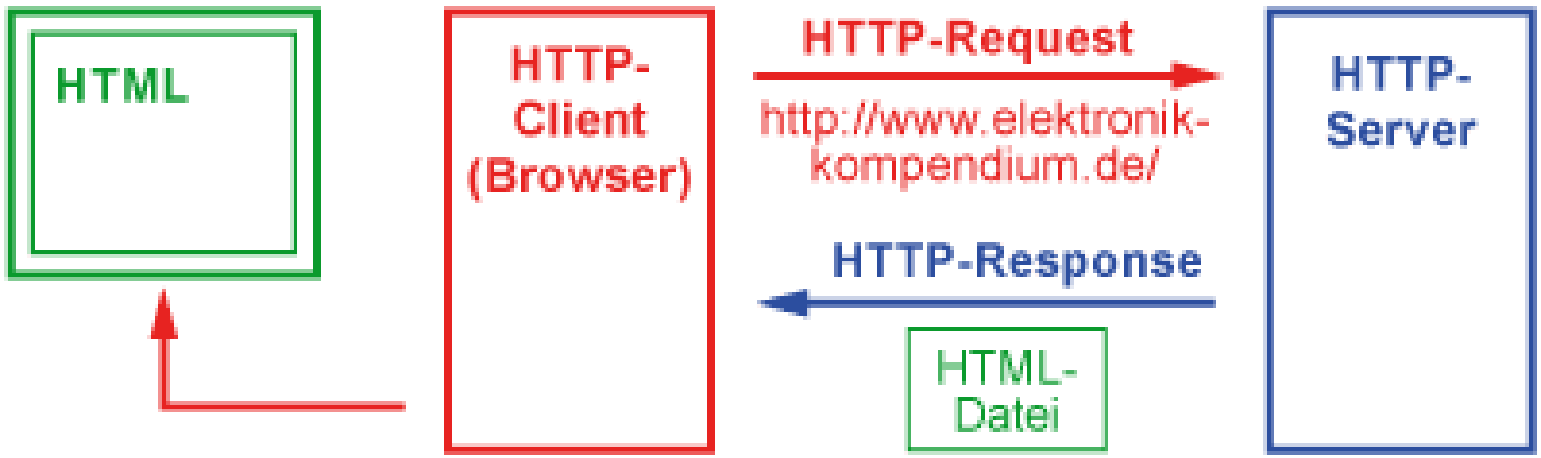

Abbildung: Client-Server Kommunikation bei Aufruf einer Webseite[15]

Einführung Web

Das HTML Dokument

```
<!DOCTYPE html>
<html>
  <head>
    <title>Beispiel</title>
  </head>
  <body>
    <p>Beispieltext</p>
    <img src="beispiel.jpg" alt="Beispielbild">
  </body>
</html>
```

- Manipulation mithilfe von Skript- und Stylingsprachen
- Genutzte Dateien müssen heruntergeladen werden

Optimierung von Ressourcen

Verwaltung von Komponenten

- Externe Dateien für Scripts und CSS
- Stylesheets im Dokumentenkopf
- Scriptdateien möglichst weit unten
- Komponenten auf mehrere Hostnamen auslagern
  - *Vorsicht*: DNS Lookups erzeugen ebenfalls Verzögerungen
- Cookie-freie Domäne für Komponenten nutzen

Reduzierung von Dateigrößen

- Gzip oder Deflate nutzen
- Entfernen von Kommentaren und Leerstellen
- Bilder nicht in HTML runterskalieren
- Metadaten entfernen
- Richtige Bildformate verwenden (gif, png, jpg)
- Optimierung durch Bildbearbeitungssoftware
- Farbkanäle minimieren

Reduzierung von HTTP Anfragen

- CSS und Script Dateien zusammenfassen
- nicht mehr Bilder als nötig
  - Vermeiden von 'Textgrafiken'
- Bilder zu CSS Sprites zusammenfassen
- Bilder mit dem 'data' URL Schema einbetten
- Imagemaps verwenden

Reduzierung von HTTP Anfragen

CSS Sprites Beispiel 1

Abbildung: Zwei Grafiken in einer Bilddatei

Variante 1:

```
#fass1 {
        object-fit: none;
        object-position: 0px 0px;
        width: 16px
        height: 16px
}

<img id="fass1" src="faesser.gif">
```

Reduzierung von HTTP Anfragen

CSS Sprites Beispiel 2

Abbildung: Zwei Grafiken in einer Bilddatei

Variante 2:

```
#fass2 {
        width: 16px
        height: 16px
        background: url("faesser.gif") -16px 0px
}
#fass2 a {
        height: 16px;
        display: block;
}
```

```
<p id="fass2"><a href="beispiel.html"></a></p>
```

Reduzierung von HTTP Anfragen

HTML Image Map Beispiel

Abbildung: Zwei Grafiken in einer Bilddatei

```
<img src="faesser.gif" usemap="#faesser">

<map name="faesser">
  <area shape="rect" coords="0,0,16,16"
   href="beispiel.html">
  <area shape="circle" coords="14,8,8"
   href="beispiel2.html">
</map>
```

Reduzierung von HTTP Anfragen

Data URL Schema Beispiel

Abbildung: Zugehörige Grafik zum unteren Schema (Konvertiert mit [10])

```
<img width="18" height="32" src="data:image/gif;
base64,R0lGODlhEgAgAMQAAOYpMaQYIMUgKQAgeyBz7kqs/
4vV///NMf+0GOasava9e//VpNViEP////b29lJSUkFBQSkpK
SAgIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAACwAAAAAEgAgAAAF7WAjjkNpjig5GCxbDCnZukUNp
0Ot7wVxkz1CQiKpEY4/0eAoFDiOEiTuuFA4E4RFNKkkKL7fh
ELC7RLDWF9MKUm43RBIZM0mJhBxeezxOEAOEgh8cQwpfAd9A
Ih8CAgSKAsAfweUlY4PEI8NVQkBfxIGiw8Sg49hnoCKfHykc
Y9uIoB9q6OlKbKifAAAAQEofhJ/tBBVCgm/mXGLccXHsXJEE
n55EMbODQcRwcHTeRERhdjf23F4cd/gIgjj2nLr6Okj6xLa4
+8M4Sj40UQQCPl0JOwaRQdFNFcFRxyEQDChiGAIHUKMmJCfJ
hQhAAA7" />
```

Bidirektionale Kommunikation

Grenzen von HTTP

Standardverhalten von HTTP:

- request/response Protokoll
- Client sendet Anfragen
- Server antwortet auf Anfragen
- keine selbstständige Verbindung durch Server
- keine ungefragte Antwort durch Server

⇒ Asynchrone Ereignisse nicht direkt mitteilbar

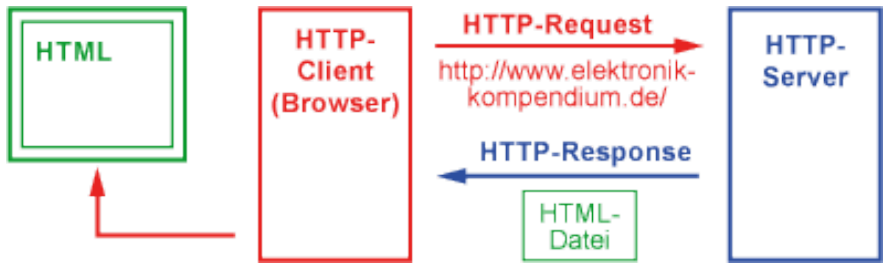

Abbildung: Client-Server Kommunikation bei Aufruf einer Webseite[15]

Polling

Verfahren:

1. Client sendet HTTP Anfrage
2. Server antwortet
3. Wiederholung in regelmäßigen Abständen

Probleme:

- Änderungen nur verzögert sichtbar
- Überflüssige Anfragen zwischen Änderungen
- HTTP overhead bei jeder Anfrage

Long Polling

Verfahren:

1. Client sendet HTTP Anfrage
2. Server wartet auf neue Änderung
3. Server antwortet
4. Client erneuert seine Anfrage

Vorteile:

- Änderungen direkt sichtbar
- Keine überflüssigen Anfragen zwischen Änderungen

Offene Probleme:

- HTTP Anfrage bei jeder Änderung
- HTTP overhead bei jeder Anfrage

WebSocket Protokoll

Verfahren:

1. Client sendet HTTP Upgrade request
2. Aufbau bidirektionaler TCP Verbindung

Vorteile:

- Server kann selbständig Änderungen mitteilen
- Kein HTTP overhead bei jeder Änderung

Nachteile:

- von älteren Browsern nicht unterstützt
- erschwerte Lastverteilung

Lastverteilung

Lastverteilung

- Viele Nutzer ⇒ Hohe Serverlast
- Mehrere Server ausfallsicherer
- Domain nur einem Server zuweisbar

⇒ Loadbalancer verteilt Anfragen

- Session Persistenz
- Hardware und Software Varianten

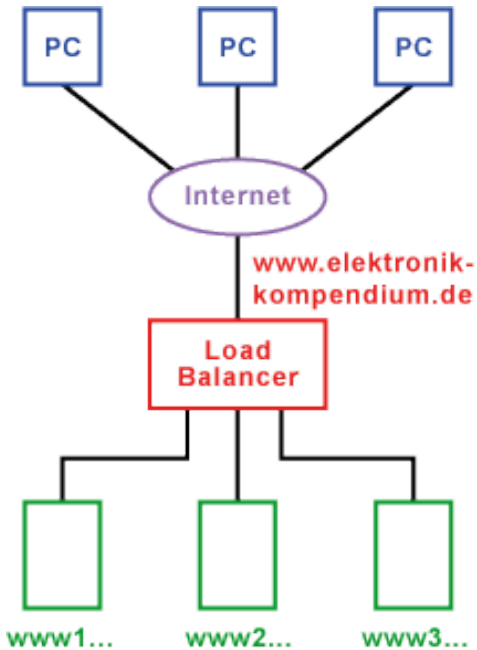

Abbildung: Anbindung mehrerer Server mit Loadbalancer[16]

DNS Lastverteilung

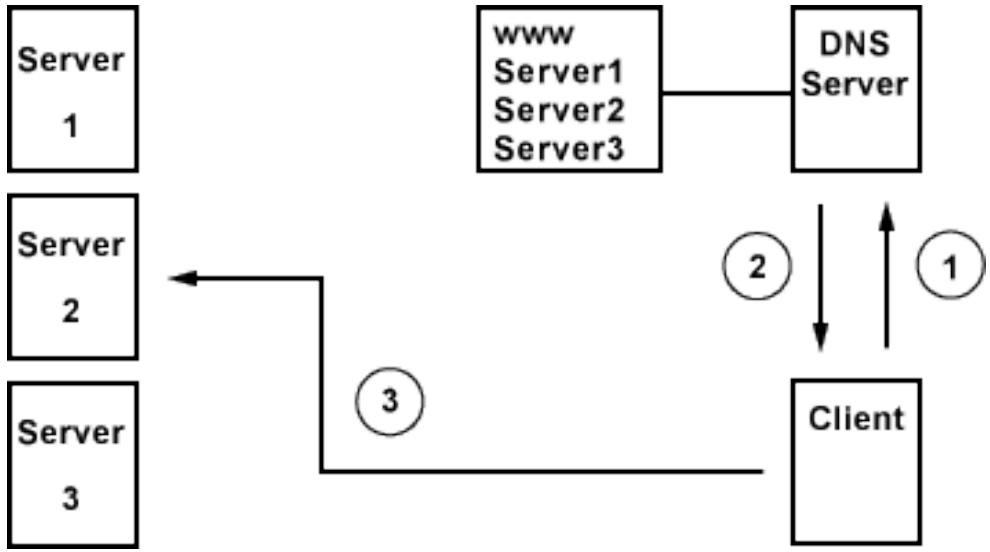

Abbildung: DNS Server verteilt abwechselnd IP's [17]

- Client speichert IP
- Serverlast nicht berücksichtigt
- IP für jeden Server

Hard-/Softwaregestützte Lastverteilung

Round-Robin-Verfahren

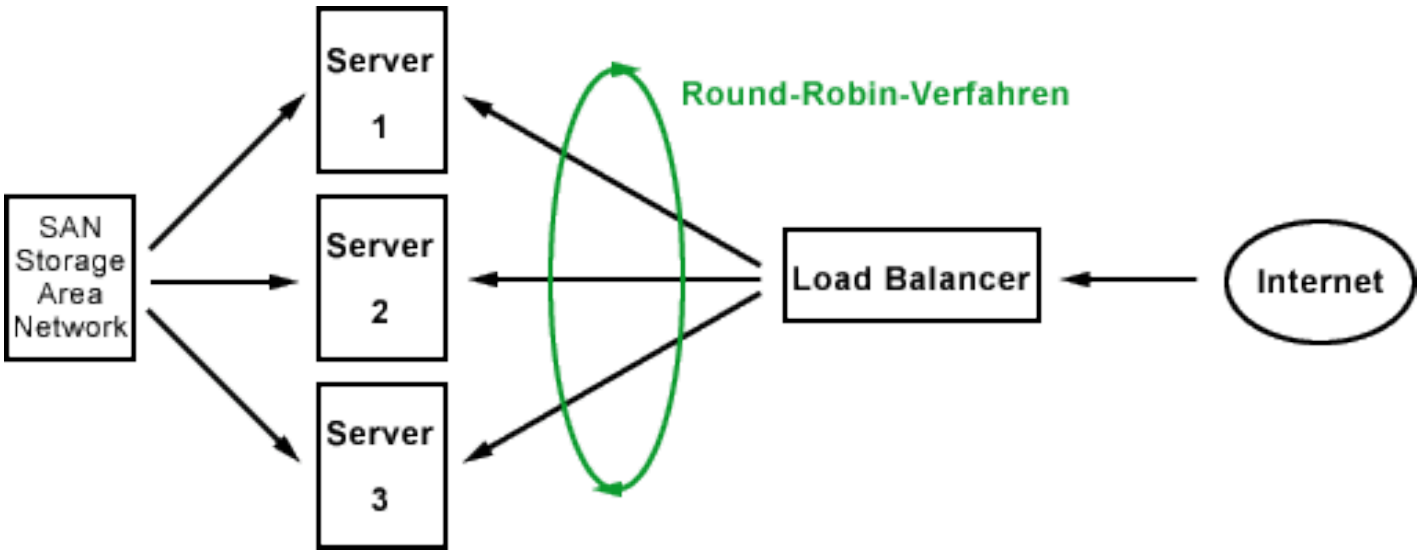

Abbildung: Abwechselnde Zuweisung durch Round-Robin-Verfahren[17]

- Loadbalancer speichert Verbindungen
- Serverlast nicht berücksichtigt

Hard-/Softwaregestützte Lastverteilung

Feedback-basiertes Verfahren

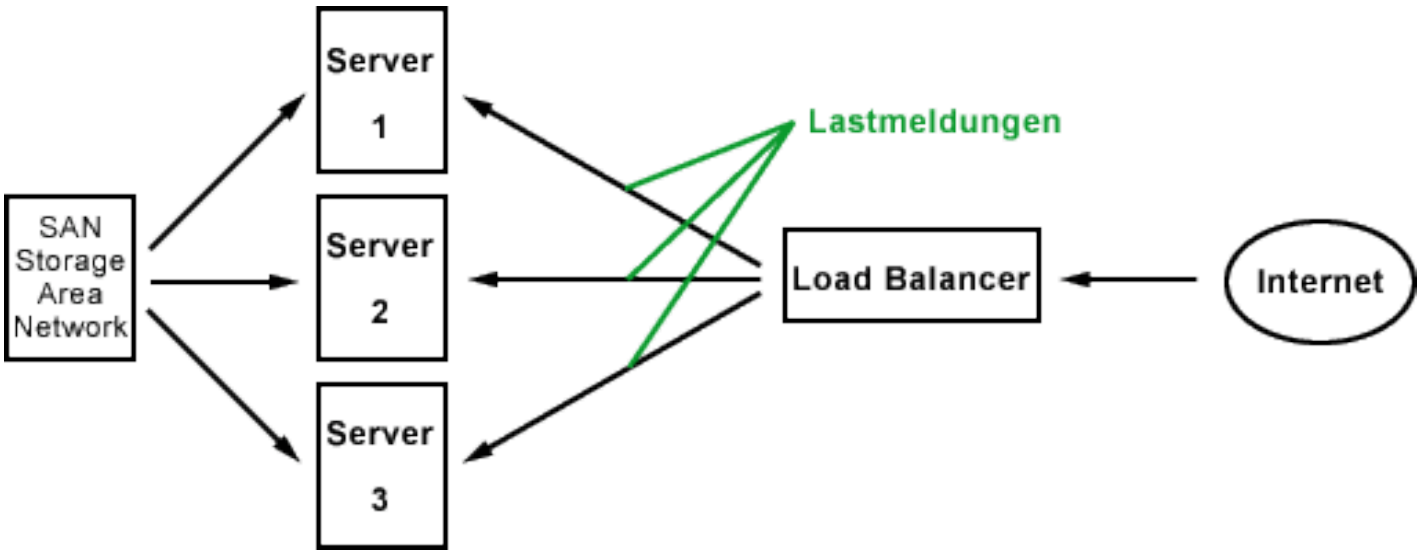

Abbildung: Lastverteilung durch Serverfeedback [17]

- Loadbalancer speichert Verbindungen
- höherer Konfigurationsaufwand
- Serverstatus durch Feedback bekannt

Hard-/Softwaregestützte Lastverteilung

URL-basiertes Verfahren

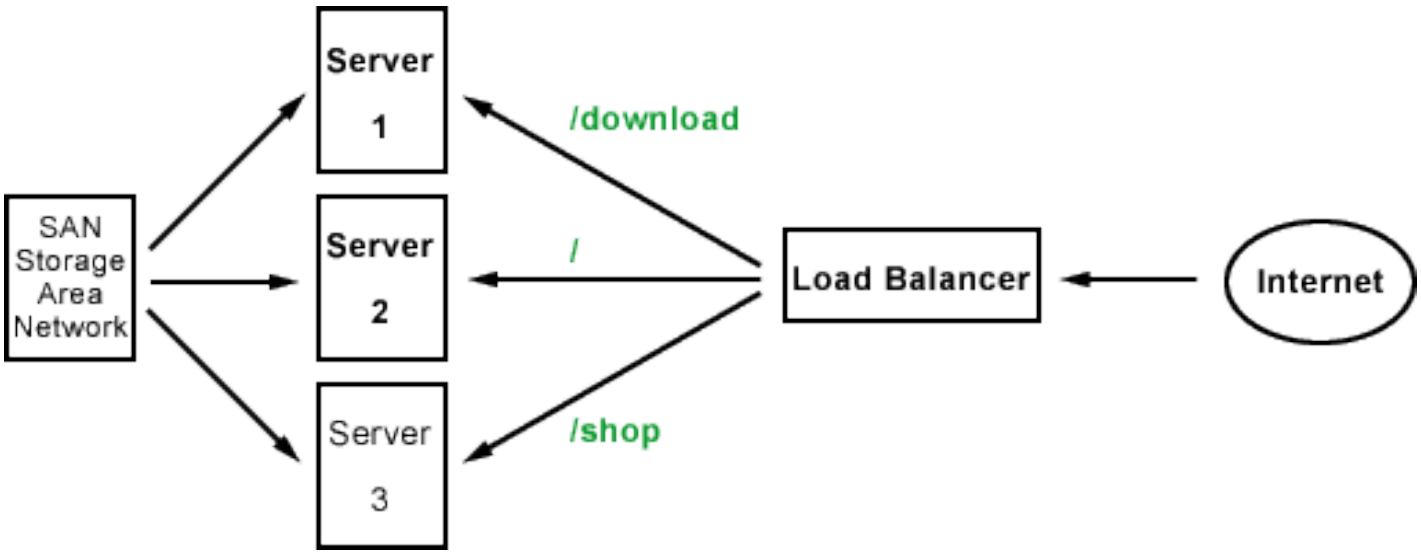

Abbildung: Verteilung spezifischen URL Anfragen auf verschiedene Server [17]

- Spezielle Hardware
- Vorhergehende Analyse

Hard-/Softwaregestützte Lastverteilung

Dienst-basiertes Verfahren

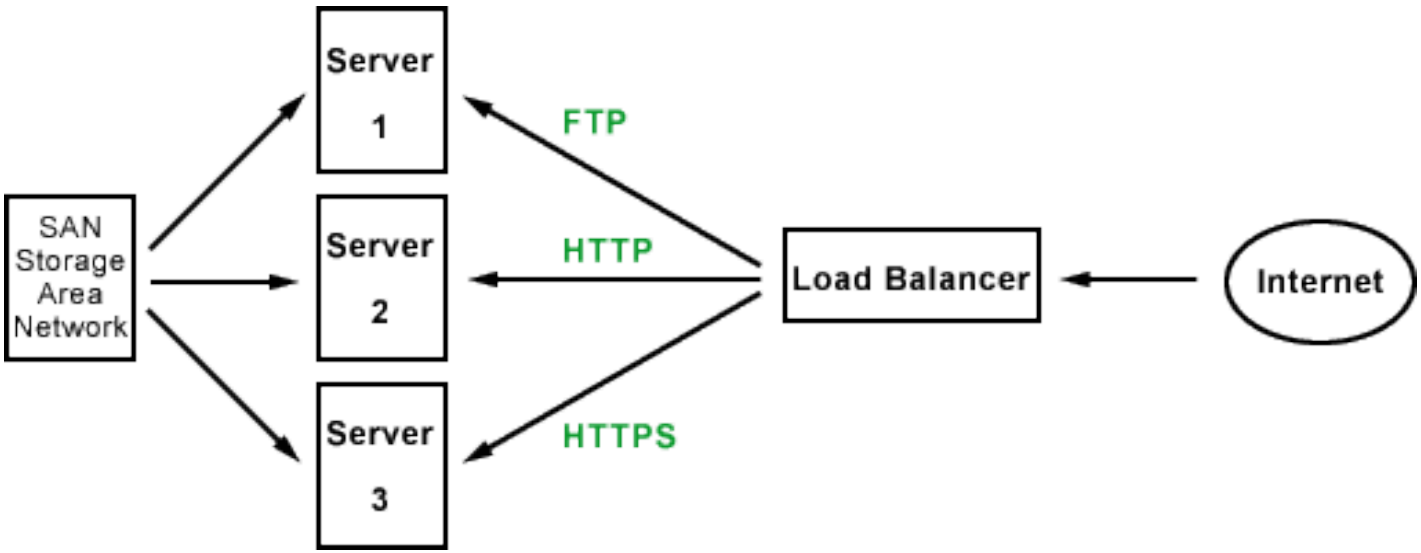

Abbildung: Verteilung verschiedener Dienste auf verschiedene Server [17]

- Dienstidentifizierung durch Ports
- Vorhergehende Analyse

Zusammenfassung

Zusammenfassung

- möglichst wenige Dateien
- Dateien komprimieren
- CSS oben, Scripts unten
- Dateien auf mehrere Hostnamen verteilen
- beidseitige Kommunikation mit WebSockets/Long Polling
- mehrere Server + Loadbalancer bei großer Serverlast

Quellen I

[1] Can i use websockets? http://caniuse.com/#search=websockets (letzter Zugriff 05.01.2017).

[2] w3schools.com.

[3] What is load balancing? https://www.nginx.com/resources/glossary/load-balancing/ (letzter Zugriff 05.01.2017).

[4] Best practices for speeding up your web site, 2006. https://developer.yahoo.com/performance/rules.html (letzter Zugriff 05.01.2017).

Quellen II

[5] Hovhannes Avoyan. 30 tips to optimize html/css/images for smooth web experience, 2011. http://www.monitis.com/blog/30-tips-to-optimize-htmlcssimages-for-smooth-web-exper (letzter Zugriff 05.01.2017).

[6] Stephen J. Bigelow. Hardware vs. software load balancer: Which is better for an enterprise?, 2016. http://searchitoperations.techtarget.com/answer/Hardware-vs-software-load-balancer-Which-is-better-for (letzter Zugriff 05.01.2017).

[7] Alex Danilo, Arron Eicholz, Steve Faulkner, and Travis Leithead. HTML 5.1. W3C recommendation, W3C, November 2016. https://www.w3.org/TR/2016/REC-html51-20161101/.

Quellen III

[8] I. Fette and A. Melnikov. The websocket protocol. RFC 6455, RFC Editor, December 2011. http://www.rfc-editor.org/rfc/rfc6455.txt.

[9] R. Fielding and J. Reschke. Hypertext transfer protocol (http/1.1): Message syntax and routing. RFC 7230, RFC Editor, June 2014. http://www.rfc-editor.org/rfc/rfc7230.txt.

[10] Mike Fosket. Image to data uri converter, 2016. https://websemantics.uk/tools/image-to-data-uri-converter/ (letzter Zugriff 05.01.2017).

[11] Wolfram Hempel. Load balancing websocket connections, 2016. https://deepstream.io/blog/load-balancing-websocket-connections/ (letzter Zugriff 05.01.2017).

Quellen IV

[12] S. Loreto, P. Saint-Andre, S. Salsano, and G. Wilkins. Known issues and best practices for the use of long polling and streaming in bidirectional http. RFC 6202, RFC Editor, April 2011. http://www.rfc-editor.org/rfc/rfc6202.txt.

[13] Larry Masinter. The 'data' url scheme. RFC 2397, RFC Editor, August 1998. http://www.rfc-editor.org/rfc/rfc2397.txt.

[14] Robin Rendle. Spriting with <img>, 2015. https://css-tricks.com/spriting-img/ (letzter Zugriff 05.01.2017).

[15] Patrick Schnabel. Http - hypertext transfer protocol. http://www.elektronik-kompendium.de/sites/net/0902231.htm (letzter Zugriff 05.01.2017).

Quellen V

[16] Patrick Schnabel. Load balancer (lastverteiler). http://www.elektronik-kompendium.de/sites/net/0904131.htm (letzter Zugriff 05.01.2017).

[17] Patrick Schnabel. Load balancing, 2006. http://www.elektronik-kompendium.de/sites/net/0906201.htm (letzter Zugriff 05.01.2017).