SPDY

Patrick Robinson

Alex Laitenberger

Inhaltsverzeichnis

Was ist SPDY

- Entwickelt von Google 2009
- Drop-in replacement für HTTP
- Zentraler Beitrag zu HTTP 2
  - Erster Draft HTTP/2 = SPDY
  - Testen & Entwickeln von Konzepten
  - Tests sehr erfolgreich -> Produktiver Einsatz
  - Superset von HTTP 2
    - QUIC

Support

- Client
  - Chrome 4, Firefox 13, IE 11, Opera 12.10, Safari 8
  - Mobile: IE 11, Safari 8, Android 3, Chrome 42, Firefox 37, Opera Mobile, Silk
- Server
  - Apache, Nginx, Node.js, Ruby, Python, Jetty, Go, Erlang, …
- Große Unternehmen
  - Google, Twitter, Facebook, Wikimedia, Reddit, Yahoo, Wordpress, …
  - CDNs (CloudFlare, MaxCDN, …)
- SPDY Gateway
  - SPDY bis zum Gateway, HTTP vom Gateway zu nicht-SPDY Hosts
  - Amazon Silk

Ziele

- Behebung von Probleme des HTTP-Protokolls
- Anpassung an Veränderungen des Webs
  - HTTP 1.1: 1999
- Vermeidung von Round-Trips
  - Besonders wichtig in Netzen mit hoher Latenz oder häufigen Fehlübertragungen (z.B. Mobilfunk, DSL „letzte Meile“)
- Sicherheit

Zentrale Konzepte

- Request Multiplexing
- Request Prioritization
  - z.B. offener vs Hintergrundtabs, Elemente, die weiteres Rendern blokieren
- Header Compression
  - Viele Daten bleiben gleich, werden aber oft gesendet (z.B. User Agent)
- Server Push
- Gateway

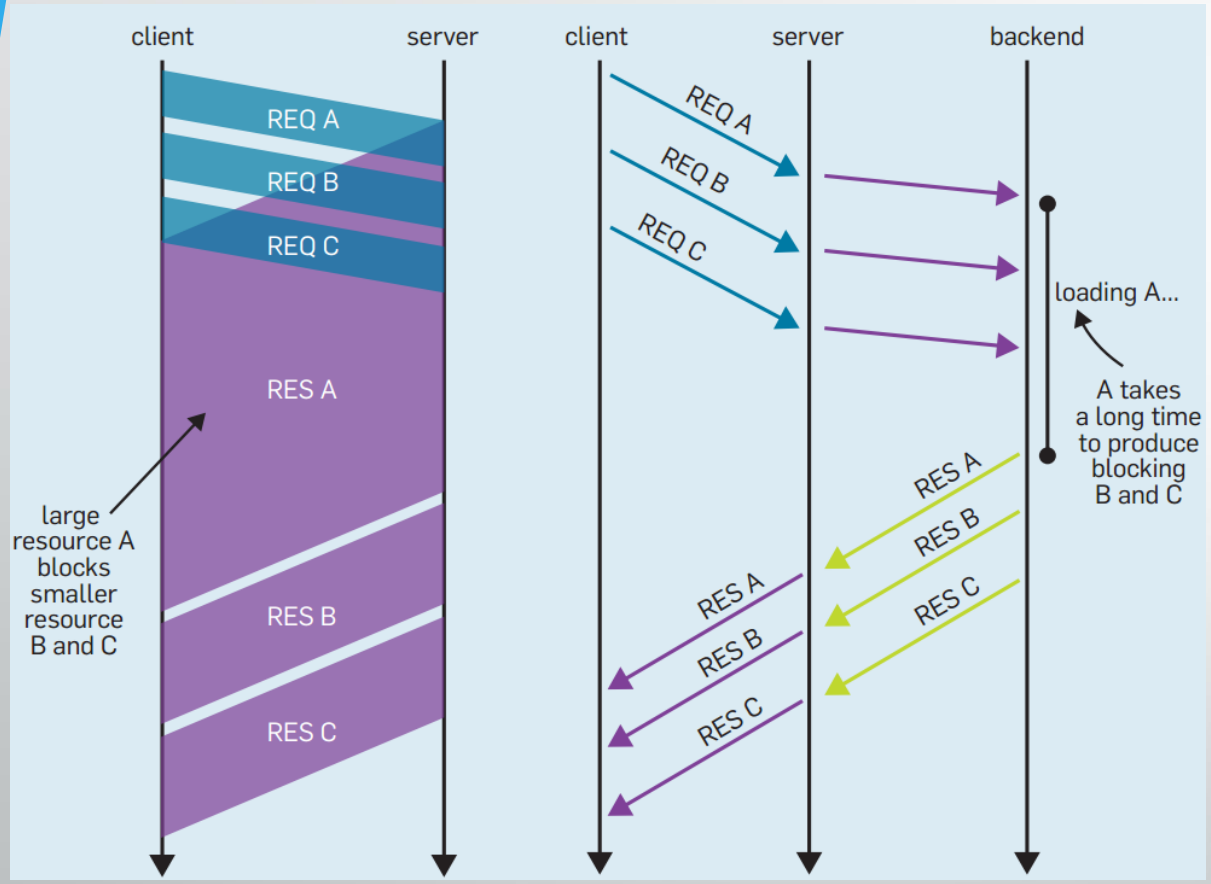
Request Multiplexing

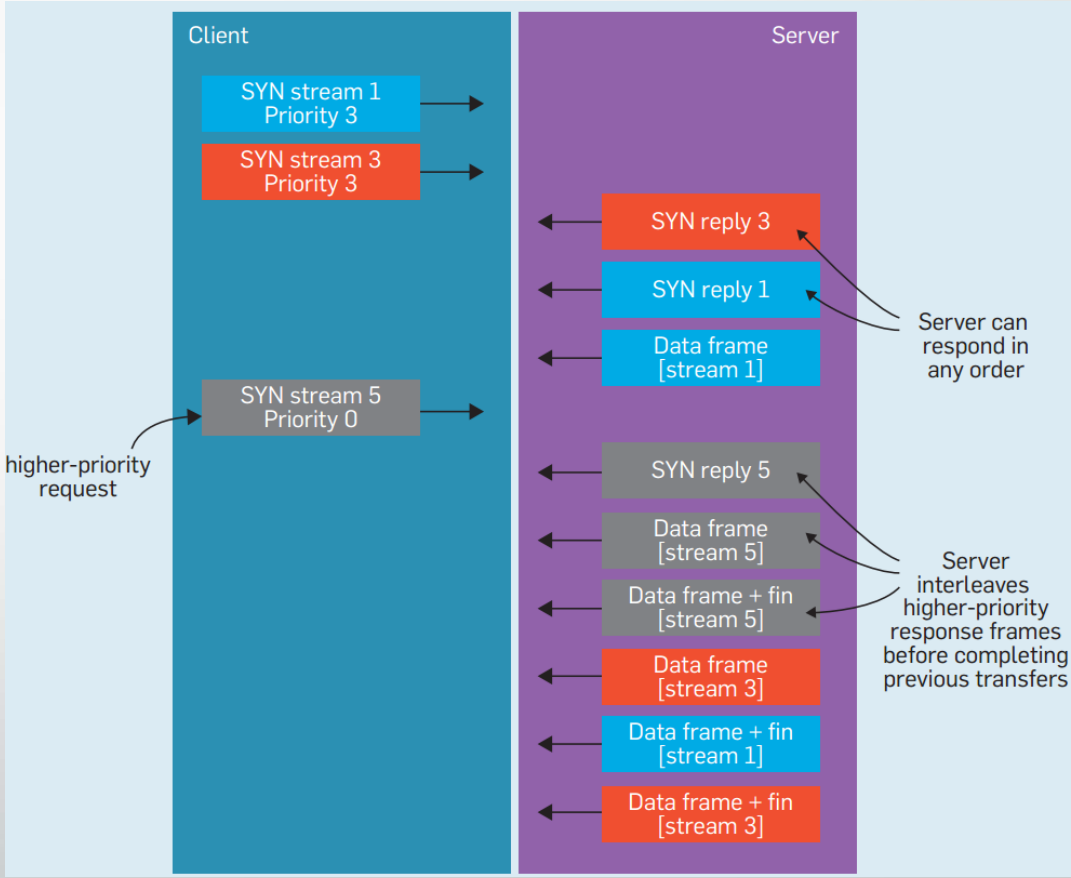
- Probleme HTTP 1.1/TCP
  - Aufbau großer Zahl HTTP-Verbindungen und multipler TCP-Verbindungen für jede HTTP-Verbindung
  - Jede TCP-Verbindung macht 3-Way-Handshake & Slow-Start durch und versucht Bandbreite zu regulieren
    - HTML: Burstartiger Traffic
  - Kurzlebige Verbindungen mit hohem Overhead und wiederholten Round-Trips
  - Widerspricht eigentlich HTTP Standard, ist aber in allen modernen Browsern
- SPDY kann mehrere Datenströme in einer Verbindung handhaben

Request Multiplexing

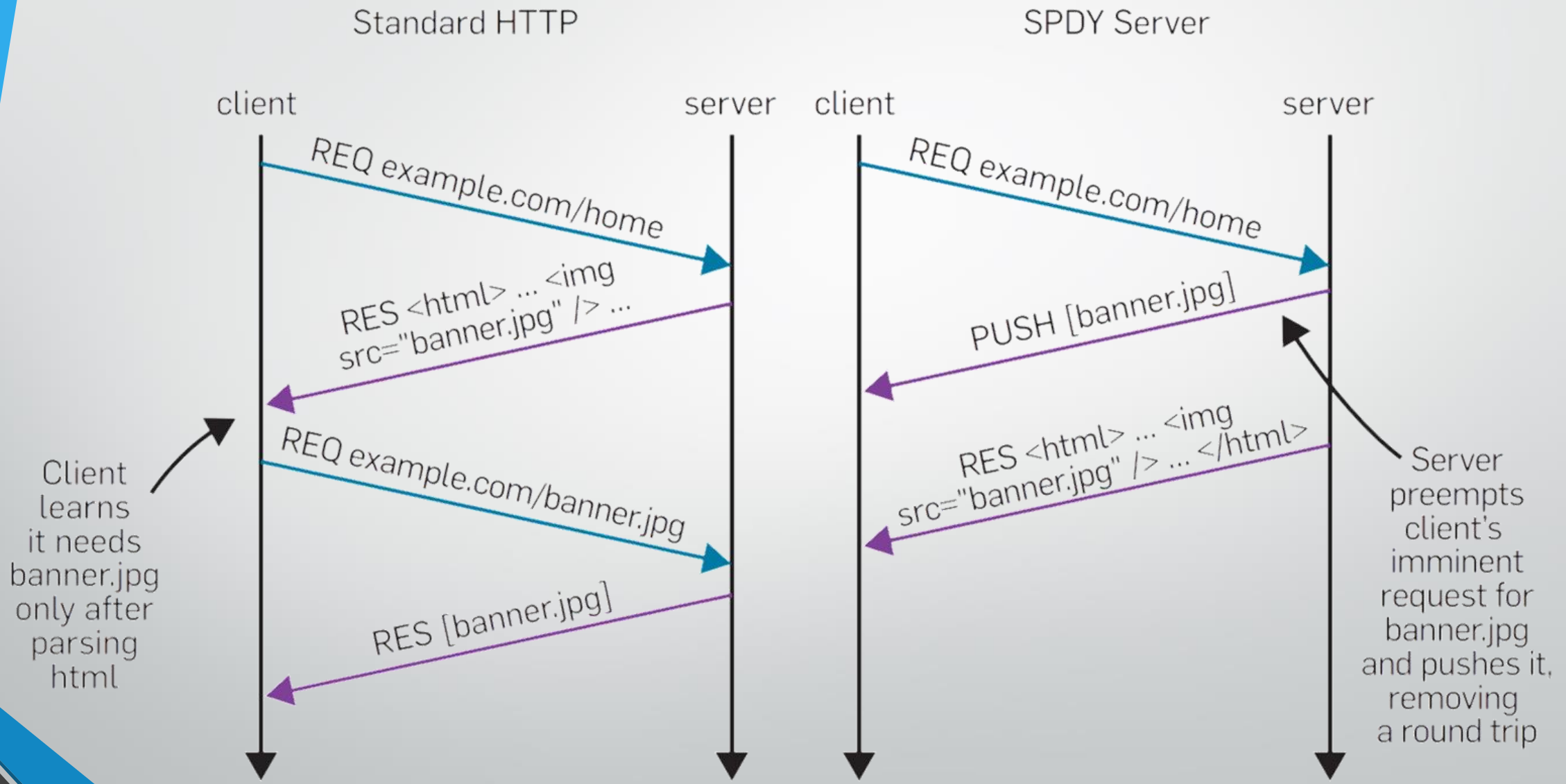
- Gleicher Host: Separate Ströme in einer SPDY(& TCP)-Verbindung
  - TCP Flusskontrolle wieder effektiv
    - Bessere Ausnutzung der Bandbreite
    - Geringe Anzahl von Retransmissions
  - Mehr „volle“ Pakete/Ethernetframes
    - Flüssigerer Datenstrom
  - Priorisierung von Paketen

Unterschiede zu HTTP Pipelining





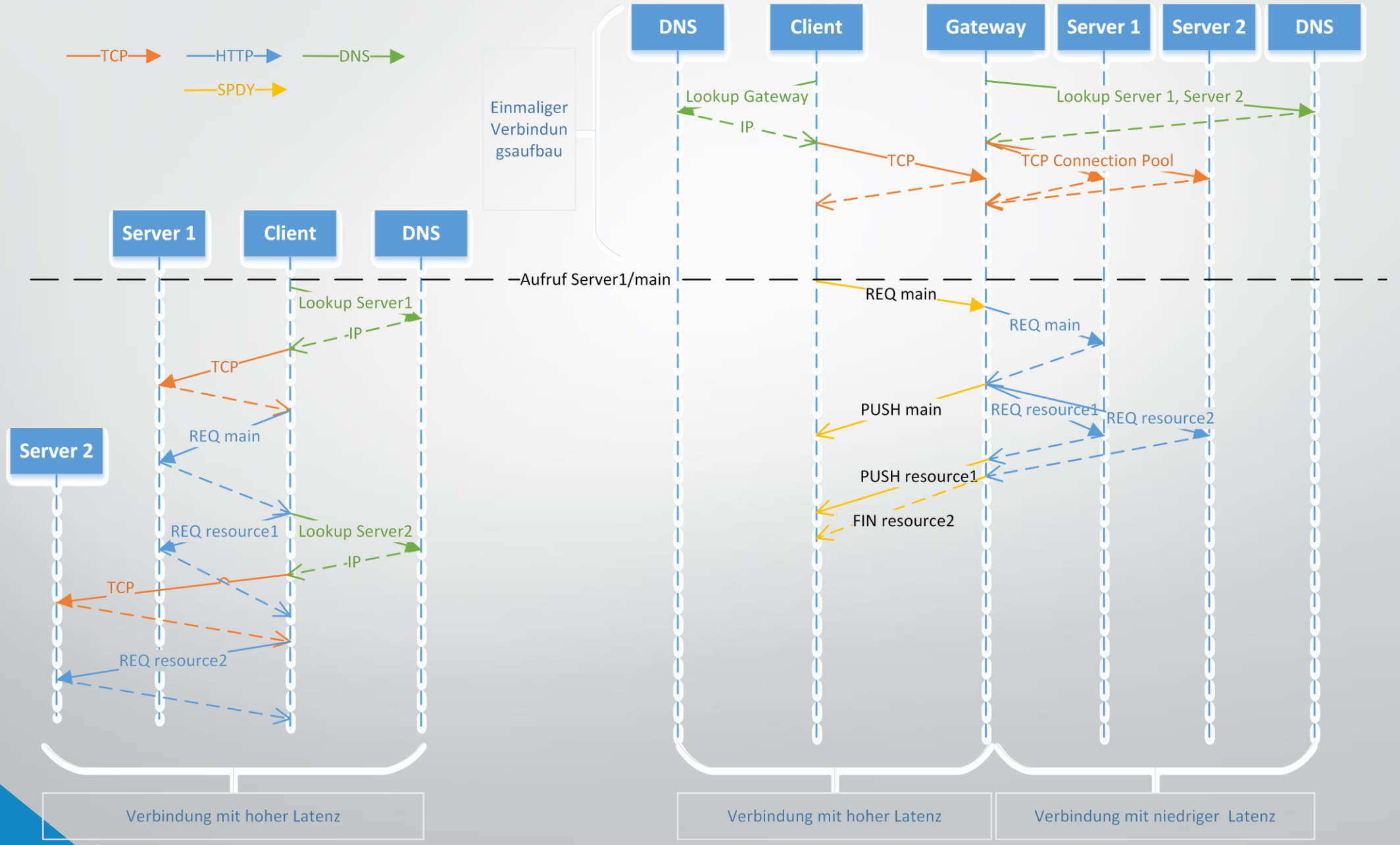
Server Push


Standard HTTP
client
server
REQ example.com/home
RES <html> ... ...
REQ example.com/banner.jpg
RES [banner.jpg]
Client learns it needs banner.jpg only after parsing html
SPDY Server
client
server
REQ example.com/home
PUSH [banner.jpg]
RES <html> ... ... </html>
Server preempts client's imminent request for banner.jpg and pushes it, removing a round trip

SPDY Gateway

- Platzierung von SPDY Gateway in Datenzentren
  - Niedrige Latenz zu Servern
  - Vorteile von SPDY über Hochlatenznetze auch ohne Serversupport
- Offene Frage: Sind SPDY Gateways auch nach einer umfassenden Nutzung von SPDY / HTTP 2 weiter sinnvoll?
  - Einige Konzepte weiterhin nützlich
  - Problem: Verschlüsselung

SPDY Gateway



SPDY Gateway

- Vorteile:
  - Nutzung der gleichen TCP-Verbindung und potenziell SPDY-Verbindung für mutiple Server z.B. eines Datacenters
    - Latenz durch Verbindungsaufbau einmal pro Datacenter statt einmal pro HTTP-Verbindung
  - Datacenter kann Pool von TCP-Verbindungen zu häufig genutzten externen Server vorhalten
  - DNS Lookup von Gateway aus (kürzere Zeiten & besser cachebar)
- Nutzung
  - Amazon Silk

Security

- Standardmäßig verschlüsselt
  - In Chrome in späteren Versionen zwangsweise Verschlüsselung
  - Teil von Googles & Mozillas Initiative zur vollständigen Verschlüsselung
- Gateways: Teilweise Verschlüsselung sonst unverschlüsselter Verbindungen

Resultate

- Verbesserungen (Durchschnitt aller Google Seiten & Dienste)
  - ~ 50 % geringerer Upload
  - ~ 20 % weniger Pakete
  - ~ 4 % geringerer Download
  - > 10 % geringere Latenz
    - Sehr unterschiedlich je nach Situation
- Nicht verändert
  - Transportprotokoll (QUIC)
  - TLS

Quellen

- SPDYing Up the Web, Bryce Thomas, Raja Jurdak & Ian Atkinson, Communications of the ACM, Volume 55 Issue 12, Dezember 2012
- Making the web speedier and safer with SPDY, Roberto Peon, Will Chang, 26.01.12