

MASTER'S THESIS

**DISTRIBUTION OF LARGE DATA IN NETWORKS
WITH LIMITED BANDWIDTH**

**WEBBASIERTE VERTEILUNG GROSSER DATENMENGEN IN
LOKALEN NETZWERKEN**

TIM FRIEDRICH

CHAIR

Internet Technologies and Systems

SUPERVISORS

Prof. Dr. Christoph Meinel

Dipl.-Inf. (FH). Jan Renz

October 5th, 2016

Tim Friedrich: *Distribution of large data in networks with limited bandwidth*, Master's Thesis, © October 5th, 2016

ABSTRACT

My english abstract

ZUSAMMENFASSUNG

Meine deutsche Zusammenfassung

ACKNOWLEDGMENTS

I would like to thank

CONTENTS

1	EINLEITUNG	1
1.1	Motivation	1
1.2	Low bandwidth networks	2
1.2.1	Educational networks	2
1.2.2	Corporate networks	2
1.3	Projekt Schul-Cloud	2
1.4	Slidesync	2
1.5	Ziele	2
1.5.1	Forschungsfrage	2
1.6	3
2	ABGRENZUNG	4
2.1	Annahmen	4
2.2	Technologische Abgrenzung	4
3	GRUNDLAGEN	5
3.1	Statische Ressourcen	5
3.2	CDN	5
3.2.1	Infrastruktur basierte CDNs	5
3.2.2	Peer To Peer basierte CDNs	6
3.2.3	Hyrid CDNs	7
3.3	Webrtc- Web Real-Time Communication	7
3.3.1	RTCPeerConnection	8
3.3.2	RTCDataChannel	8
3.3.3	MediaStream	8
3.3.4	Sigaling	8
3.3.5	SDP	8
3.3.6	TURN Server	8
3.3.7	STUN Server - Simple Traversal of User Data- gram Protocol [UDP] Through Network Address Translators	8
3.3.8	ICE	8
3.4	DataCache	8
3.5	IndexedDB	8
3.6	Service Worker	9
3.6.1	Lebenszyklus	9
3.7	Websockets	10
3.8	Distributed caches	10
4	KONZEPT	11
4.1	Mesh Zuordnung - Verbinden von Peers	12
4.1.1	Schul-Cloud	12
4.1.2	Slidesync	13

4.2	Objekt Zuordnung - finden von Ressourcen	13
4.2.1	Subnetzerkennung	14
4.2.2	Struktur von IP Adressen	14
4.3	Wiederverwendbarkeit	14
4.4	Open Source	14
5	IMPLEMENTIERUNG	15
5.1	Architectur	15
5.1.1	Service worker	15
5.1.2	Tests	15
5.2	Ressourcen Management	16
5.2.1	Update Protokoll	16
5.2.2	Quota limits	16
5.3	Signaling Server	16
5.4	Message protocol	16
5.4.1	hash updates	16
5.5	Data serialization	16
5.6	Mesh Zuordnung	17
5.7	Reusability	17
5.8	System Test	17
5.9	Storage quotas	17
5.10	Resource loading	17
6	EVALUATION	19
6.1	Prequisites	19
6.2	Technical evaluation	19
6.2.1	Bandwidth	19
6.2.2	Nutzerzufriedenheit	19
6.3	Browser compatibility	19
6.3.1	Browser Usage in corporate networks	19
6.3.2	Browser usage in educational networks	19
6.4	Security considerations	19
6.5	DRM licencing	19
7	CONCLUSION	20
A	AN APPENDIX	21
	BIBLIOGRAPHY	viii
	LIST OF FIGURES	ix
	LIST OF TABLES	x
	LIST OF LISTINGS	xi

ACRONYMS

EINLEITUNG

1.1 MOTIVATION

In den letzten Jahren hat sich das verwendete Datenvolumen des Internets immer weiter gesteigert. Waren es 2015 noch monatlich 72 Petabyte pro Monat sind es 2016 bereits 96 Petabyte. Zwar ist die vorhandene Bandbreite bei vielen Nutzern ebenfalls gestiegen jedoch bezieht sich die vor allem auf Ballungsgebiete. In ländlicheren Regionen ist die Bandbreite in Deutschland in vielen Fällen weiterhin nicht ausreichend. Insbesondere wenn viele Nutzer sich gemeinsam eine Internet Anbindung teilen müssen ist dies ein Problem.*Studie*

Immer mehr Unternehmen halten Ihre Hauptversammlungen, Kundengungen, und Pressemitteilungen über Live Streams im Internet ab.*Studie*

Dies stellt sie vor das Problem das trotz oftmals guter Internetanbindung zu viele Mitarbeiter das Video über die Internetanbindung laden müssen, was zu einer vollständigen Auslastung des WANs führen kann. Dies wiederum kann zur Folge haben, dass ein Arbeiten für die restliche Belegschaft schwierig bis unmöglich wird. Der dadurch entstandene Schaden ist oft nur schwer zu beziffern, geht jedoch schnell in die Millionen.(klingt ohne ref nicht gut)*Studie*

Nicht nur bei Unternehmen sondern auch in Schulen ist die Digitalisierung auf dem Vormarsch. Zunehmend werden Online-Lernplattformen im Unterricht eingesetzt. Diese Entwicklung wird jedoch stark ausgebremst durch fehlende Internet Bandbreiten. Viele Schulen haben eine schlechtere Internetanbindung als viele privat Haushalte. Um statische Inhalte anzeigen zu können, muss jeder Schüler einer Klasse sich diese über das WAN aus dem Internet herunterladen. Da jedoch Schüler in derselben Klasse oft die gleichen Inhalte benötigen, kann Bandbreite gespart werden, indem diese Inhalte nur einmal über das Internet geladen und anschließend im lokalen Netzwerk verteilt werden.

Beide Anwendungsfälle haben gemeinsam das viele Nutzer die selben Inhalte zur annähernd gleichen Zeit benötigen und zum aktuellen Zeitpunkt häufig über das Internet laden müssen. Diese Zeitliche und inhaltlich Lokalität kann genutzt werden um die benötigte Bandbreite zu reduzieren, indem die Inhalte nur einmal über das WAN geladen und anschließend im lokalen Netzwerk verteilt werden.

Die folgende Arbeit beschäftigt sich mit der Frage ob ein Peer to Peer Ansatz eingesetzt werden kann um die Ladezeiten in Netzwerken mit begrenzter WAN Bandbreite die Ladezeiten zu verringern.

1.2 LOW BANDWIDTH NETWORKS

1.2.1 Educational networks

1.2.2 Corporate networks

1.3 PROJEKT SCHUL-CLOUD

Das Projekt Schul-Cloud¹ ist ein Gemeinschaftsprojekt des Hasso-Plattner-Instituts und des nationalen Excellence-Schulnetzwerkes (MINT-EC). Im Mai 2017 startete die Pilotphase des Projektes mit insgesamt 27 Schulen. Ziel des Projektes ist die Förderung der Digitalisierung in Schulen. Zu diesem Zweck wurde eine Web basierte Plattform entwickelt die Lehrer und Schüler bei der Unterrichtsvorbereitung, Durchführung und Nachbereitung unterstützen soll.

Lehrer können Kurse anlegen und diese nutzen um Materialien sowie Aufgaben zu verteilen. Schülern ist es über die Plattform möglich Lösungen für Aufgaben einzureichen und ihr Ergebnis einzusehen. Über einen Kalender können sie Ihren Stundenplan abrufen.

Das Projekt wird als Open Source Projekt zur Verfügung gestellt und basiert auf ein Microservice Architektur. Durch die Verwendung von Microservices wird eine einfachere Anbindung an bestehende Infrastrukturen ermöglicht. Des weiteren können einzelne Services ersetzt werden um die Plattform an die Anforderungen der Schulen anzupassen. Bereitgestellt wird die Plattform mit Hilfe von Cloud Hosting Ansätzen, bei dem die Infrastruktur zentral und nicht von jeder Schule bereitgestellt wird. Dies ermöglicht eine einfache Skalierung. Neben der Web Anwendung existieren native Apps für Android und IOS.

unsicher wie detailliert ich hier werden soll

1.4 SLIDESYNC

1.5 ZIELE

1.5.1 Forschungsfrage

Diese Arbeit wird versuchen die Frage: Wie können in einem Netzwerk mit geringerer Internetanbindung Datenintensiven Ressourcen ausgeliefert werden? zu beantworten.

¹ <https://schul-cloud.org/>

Dabei wird ein Fokus auf Peer To Peer Technologien gesetzt. Zur Evaluation wird neben simulierten Benchmarks auch der Einsatz unter realen Bedingung getestet.

1.6

Hier muss klar sein was gemacht werden soll !!!

ABGRENZUNG

2.1 ANNAHMEN

2.2 TECHNOLOGISCHE ABGRENZUNG

- mehrere nutzer die gleiche ressourcen abrufen - Browserbasiertes
P2P CDN -

GRUNDLAGEN

3.1 STATISCHE RESSOURCEN

Statische Ressourcen sind Inhalte einer Website die für alle Nutzer gleich sind. Sie sind im Gegensatz zu dynamischen Inhalten nicht nutzerspezifisch und können daher gut über ein CDN verteilt werden. Insbesondere die so genannten Assets einer Internetseite sind meist statisch. Dies sind meist Javascript, CSS aber auch Bild Dateien. Auch Videos fallen häufig in diese Kategorie.

3.2 CDN

Unter einem CDN, auch Content Delivery Network versteht man ein Netzwerk in dem sich Clients Inhalte von einer Reihe von Knoten laden. Ein CDN stellt dem Nutzer Auslieferungs und Speicherkapazitäten zur Verfügung. Dadurch kann die Last auf dem Ursprungsserver und die Latenz auf Seiten der Nutzer reduziert werden. Die reduzierten Ladezeiten werden unter anderem durch eine bessere geographische Nähe und damit geringerer Netzlaufzeiten erreicht.

Es lassen sich drei Klassen von CDNs unterscheiden. Infrastruktur basierte CDN die auf einer geografisch verteilten Server Infrastruktur basieren, Peer To Peer basierte CDNs bei denen die Inhalte direkt zwischen den Teilnehmern verteilt werden und Hybride CDNs die aus einer Kombination aus Server Infrastruktur und Peer To Peer Verteilung beruhen.

3.2.1 *Infrastruktur basierte CDNs*

Infrastruktur basierte CDNs bestehen aus einem Ursprungsservern, der von dem Bereitsteller der Inhalte kontrolliert wird, und einem Netzwerk aus replica Servern. Die replica Server übernehmen die Verteilung der Inhalte an die Clients. Sie fungieren als ein möglichst regionaler cache in dem Inhalte des Ursprungsservers gespiegelt werden. Ein Distributionssystem ist dafür verantwortlich die Inhalte auf den replicas zu aktualisieren und übernimmt das Routing bei einer Anfrage eines Clients. Unter Zuhilfenahme verschiedener Metriken

versucht das Distributionssystem einen möglichst optimalen replica Server für den Client zu finden. Diese Metriken unterscheiden sich zwischen den Anbietern. Häufig werden jedoch geographische Entfernung, Latenzzeiten und die Übertragungsrate berücksichtigt. Um eine möglichst geringe Latenz zu erreichen sind infrastruktur basierte CDNs häufig geografisch sehr verteilt und bestehen aus mehreren tausend replica Servern. So hat Akamai, einer der größten CDN Anbietern, über 137000 Server in 87 Ländern. [8]

3.2.2 Peer To Peer basierte CDNs

Bei einem Peer To Peer Netzwerk handelt es sich um eine Netzwerk Struktur bei der alle Teilnehmer gleichberechtigt sind. Sie bildet damit das gegen Konzept zur klassischen Client-Server Struktur, bei der einer oder mehrere Server einen Dienst anbieten der von Clients genutzt werden kann. In einem Peer To Peer Netzwerk können die Teilnehmer sowohl Dienste anbieten als auch nutzen. Man unterscheidet zwischen unstrukturierten und strukturierten Peer To Peer Netzwerken.

Unstrukturierte Netzwerke haben keine Zuordnung von Objekten und Teilnehmern. Es ist nicht möglich gezielt nach einem Objekt zu suchen. Sie lassen sich einteilen in zentralisierte, reine und hybride Peer To Peer Netzwerke. Zentralisierte Netze haben zur Verwaltung einen Server der unter anderem die Verbindung der Teilnehmer übernimmt. Dadurch ist es möglich eine Verbindung aufzubauen ohne das die IP Adresse im Vorfeld bekannt ist. Reine Peer To Peer Netzwerke haben keinen zentralen Verwaltungsserver. Die Verwaltung des Netzwerkes wird von den Teilnehmern selber übernommen. Das hat zur Folge das eine Verbindung nur möglich ist, wenn die IP Adresse des anderen Teilnehmers bekannt ist.

Strukturierte Peer To Peer Netzwerke haben eine Zuordnung von Objekt und Teilnehmer. Es ist also möglich gezielt nach einem Objekt zu suchen. Dies wird häufig über verteilte Hash Tabellen, über die mit einem verteilten Index gesucht werden kann, realisiert.

Typische wenn auch nicht notwendige Charakteristika sind laut Steinmetz[7]:

- Heterogenität der Internetbandbreite der Teilnehmer
- Verfügbarkeit und Qualität der Verbindung zwischen Teilnehmern kann nicht vorausgesetzt werden
- Dienste werden von den Teilnehmern angeboten und genutzt
- Die Teilnehmer bilden ein Netz das auf ein bestehendes Netz aufgesetzt wird (Overlay Netzwerk) und stellen Suchfunktionen bereit

- Es besteht eine Autonomie der Teilnehmer bei der Bereitstellung von Ressourcen
- Das System ist selbstorganisiert
- Die restlichen Systeme müssen nicht skaliert werden und bleiben intakt

Reine Peer To Peer CDN Systeme sind im Kontext von Internet Anwendungen eher selten zu finden. Ein klassischer Anwendungsfall für diese Systeme ist das Filesharing.

3.2.3 Hybrid CDNs

Hybrid CDNs kombinieren Peer To Peer CDNs und Infrastruktur basierte CDNs. Bei hybriden CDNs wird zuerst versucht die Resource über das Peer Netzwerk zu laden. Ist dies nicht möglich wird auf ein Infrastruktur basiertes CDN zurück gegriffen. Dadurch kann die Last auf dem CDN verringert und durch die Kombination verschiedener CDNs eine bessere Ausfallsicherheit erreicht werden. Häufig kommt diese Art der CDNs zum Einsatz wenn Ressourcen für Websites mit einem Peer To Peer Ansatz verteilt werden sollen. Da in diesem Kontext nicht alle Teilnehmer die technischen Voraussetzungen mitbringen um an dem Peer To Peer Netzwerk teilzunehmen ist eine entsprechende alternative Lösung nötig. Da die viele Websites bereits mit einem Infrastruktur basierten CDN arbeiten ist es naheliegend dieses weiter zu verwenden.

3.3 WEBRTC- WEB REAL-TIME COMMUNICATION

Webrtc ist ein Framework mit dem Echtzeit Kommunikation zwischen Browser und mobilen Anwendungen ermöglicht wird. Mit Hilfe von Webrtc ist es möglich eine Peer To Peer Verbindung aufzubauen und Daten direkt zwischen den Clients auszutauschen ohne das externe Plugins erforderlich sind. Insbesondere der Austausch von Multimedia Inhalten soll ermöglicht werden. Neben der Unterstützung für video und audio Inhalten gibt es jedoch auch die Möglichkeit Daten auszutauschen. Webrtc ist ein vom W3C[2] standardisierter offener Standart an dem seit Frühjahr 2011 gearbeitet wird. Aktuell wird Webrtc von Chrome, Firefox, Android und iOS unterstützt.¹ Webrtc implementiert drei APIs: MediaStream, RTCPeerConnection und RTCDataChannel die im folgenden genauer beschrieben werden.

¹ <https://caniuse.com/#feat=rtcpeerconnection>

3.3.1 *RTCPeerConnection*

3.3.2 *RTCDataChannel*

3.3.3 *MediaStream*

Die Mediastream api, auch `getUserMedia`, ermöglicht es Echtzeit Daten wie audio oder Video aufzunehmen, anzuzeigen und an andere Clients weiter zu leiten und repräsentiert Medien Streams wie z.b. Audio oder Video Streams. Sie ermöglicht unter anderem den Zugriff auf Video Kameras und Mikrofone. Durch Sie ist es möglich auf die Hardwareunterstützung für Videos mittels open GL zuzugreifen. MediaStreams lassen sich mithilfe des `src` Attributes von HTML 5 video Elementen in das DOM einbinden. MediaStreams wurden von vom W3C in einem eigenen Standard definiert.[3]

3.3.4 *Sigaling*

3.3.5 *SDP*

3.3.6 *TURN Server*

3.3.7 *STUN Server - Simple Traversal of User Datagram Protocol [UDP] Through Network Address Translators*

Da die Anzahl von IPv4 Adressen begrenzt ist, verwenden die meisten Subnetze NATs. Das hat zur Folge das diese Clients nicht wissen wie über welche IP Adresse und welchen Port sie erreichbar sind. Daher ist der Einsatz von STUN Servern nötig um einen Verbindungsaufbau zu ermöglichen. Stun Server überprüfen eingehende Anfragen auf IP Adresse und Port und senden diese Informationen zurück an den Client, der somit in der Lage ist diese Information weiter zureichen und damit auch außerhalb seines lokalen Netzwerkes erreichbar ist. Das STUN-Protokoll ist im RFC 3489 [5] definiert und ist nicht auf WebRTC beschränkt.

3.3.8 *ICE*

3.4 DATACACHE

3.5 INDEXEDDB

IndexedDB ist ein HTML 5 Feature um Daten im Browser zu speichern. Es wurde vom W3C standardisiert[1] und soll den veralteten websql standard ablösen. Im Gegensatz zu websql hat die IndexedDB keine strukturierte Query Language und ihr liegt kein relationelles

Modell zu grunde. Sie stellt einen Key-Value Store bereit der in der Lage ist auch große Datenmengen effektiv bereit zu stellen. Dabei ist der Datenzugriff auf die selbe Domain beschränkt. Die API ist überwiegend asynchron und basiert auf Promises.

3.6 SERVICE WORKER

Service Worker sind Skripte die im Browser als separate Prozesse im Hintergrund laufen, so genannte Web Worker. Sie stellen die Funktionalitäten eines programmierbaren Network Proxies bereit. Durch Service Worker ist es möglich die Anfragen einer Seite zu kontrollieren auf sie zu reagieren und in den Prozess einzugreifen.[6] Service Worker haben keinen Zugriff auf das DOM. Sie können mehrere Browser-Tabs und mit Hilfe des PostMessage Protokolls können Nachrichten zwischen Service Worker und Browser-Tab ausgetauscht werden. Da Service Worker Zugriff auf den DataCache und die IndexedDB haben werden sie häufig verwendet um Internetseiten offline verfügbar zu machen. Bei der Registrierung eines Service Workers wird ein URL-Scope fest gelegt für den der Service Worker zuständig ist. Nur Anfragen die sich innerhalb des URL-Scopes des Service Workers befinden können von diesem bearbeitet werden.

3.6.1 Lebenszyklus

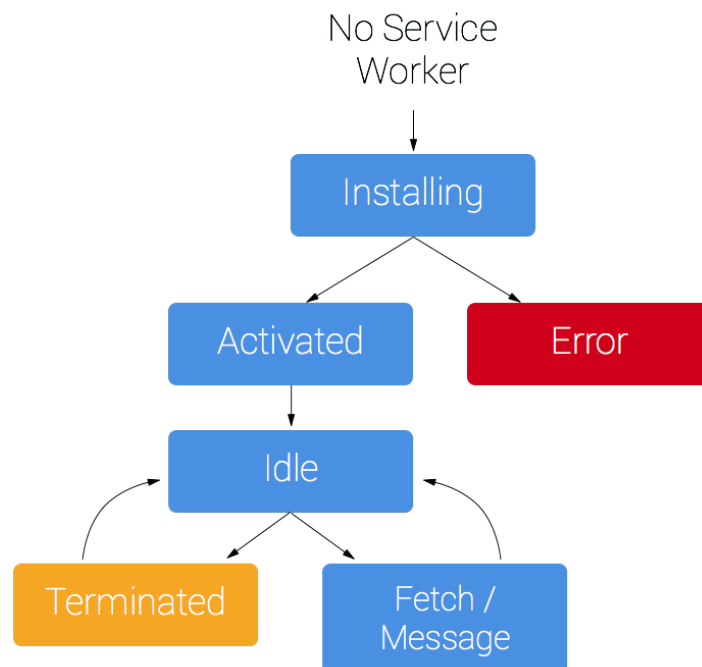


Figure 1: Lebenszyklus eines Service Workers²

Nach dem ein Service Worker registriert wurde befindet er sich im Zustand der Installation. Während der Installation werden häufig Inhalte in den Cache geladen. Wurde der Service Worker erfolgreich installiert wird er aktiviert. Ab diesem Punkt kann er Requests über das fetch event abfangen. Um Arbeitsspeicher zu sparen wird der Service Worker terminiert falls er keine fetch oder message events empfängt. Dies hat zur Folge das ein Service Worker sich nicht auf den globalen Zustand verlassen kann, sondern benötigten Zustand stattdessen auf die IndexedDb ausgelagert werden muss.

3.7 WEBSOCKETS

Websockets ist ein, auf TCP basierendes, Protokoll das bidirektionale Verbindungen zwischen Server und Webanwendung ermöglicht. Nachdem der Client eine WebSocket Verbindung zum Server aufgebaut hat ist es dem Server im Gegensatz zu HTTP möglich ohne vorherige Anfrage des Clients Daten an ihn zu senden. Zum initiieren einer Verbindung wird ein Handshake durchgeführt der vom Client angestoßen werden muss. Dazu wird wie bei HTTP der Port 80 verwendet. Der Server antwortet bei erfolgreichem Handshake mit dem HTTP status code 101. Um eine Abwärtskompatibilität zu gewährleisten werden ähnliche Header wie bei HTTP verwendet.

Neben dem unverschlüsseltem URI-Schema ws definiert RFC6455[4] auch das verschlüsselte Schema wss. Da sehr wenig Daten overhead bei der Kommunikation besteht eignen sich Websockets insbesondere für Anwendungen die eine geringe Latenz benötigen. Websockets werden von allen modernen Browsern unterstützt.

RFC6455[4]

3.8 DISTRIBUTED CACHES

KONZEPT

Beim klassischen Client Server Modell muss sich jeder Client Ressourcen über das WAN laden. Abbildung 2 zeigt den typischen Aufbau eines solchen Netzwerks. Sind die geladenen Ressourcen groß kann die WAN Anbindung zu einem Flaschenhals werden. Durch die dadurch resultierenden langen Ladezeiten kann es zu einer starken Einschränkung des Nutzererlebnisses und der Nutzerzufriedenheit kommen. *(studie einfügen)*

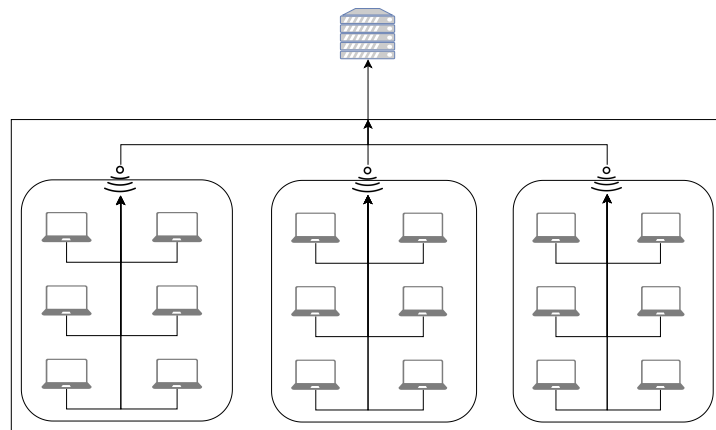


Figure 2: Netzwerkverkehr in einem herkömmlichen Netzwerk

In den Betrachteten Anwendungsfällen besteht eine hohe zeitlich und inhaltliche Lokalität der Daten. Dies kann genutzt werden um die benötigte Bandbreite zu reduzieren. Dazu soll im Folgenden eine interne Verteilung mittels eines hybriden Peer To Peer Ansatzes untersucht werden. Abbildung 3 zeigt exemplarisch den Aufbau eines solchen Netzwerkes. Anstatt das jeder Client sich die Ressource von einem externen Server lädt, lädt nur noch ein Nutzer je Subnetz die Ressource über das WAN. Dieser verteilt die Ressource dann im internen Netzwerk an andere Clients die diese dann ebenfalls wieder bereitstellen.

Benötigt ein Client eine Ressource versucht er zuerst die Ressource über sein Peer To Peer Mesh zu laden. Ist dies nicht möglich lädt er sie über einen externen Server. Hat ein Peer eine Ressource geladen speichert er sie zwischen und stellt sie für andere Clients bereit.

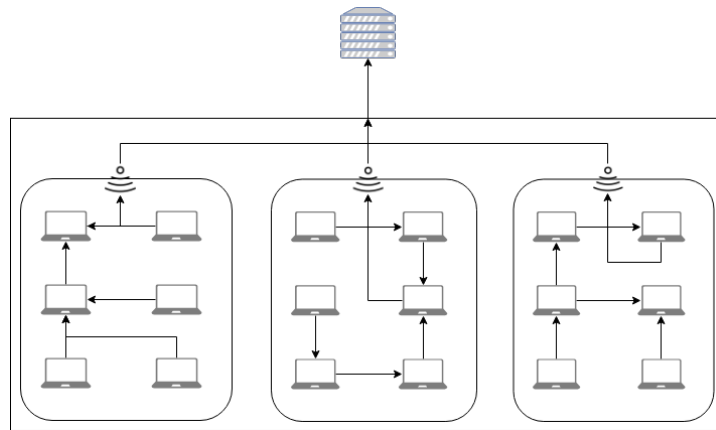


Figure 3: Netzwerkverkehr in einem Peer To Peer CDN

Da sowohl im Kontext der Schule als auch bei Unternehmen kein Wissen im Bereich der Computer Administration seitens der Nutzer vorausgesetzt werden kann, wurde ein Ansatz gewählt, der keine Installation auf Seiten der Nutzer benötigt. Das Nutzererlebnis soll dabei nicht negativ beeinflusst werden.

4.1 MESH ZUORDNUNG - VERBINDEN VON PEERS

Im Folgenden werden verschiedene Zuordnungsstrategien zur Bildung von Peer Meshes diskutiert. Dazu wird der typische Workload beider Anwendungen analysiert, um eine jeweils geeignete Strategie zu wählen.

4.1.1 Schul-Cloud

Analyse workload mit grafik

Schulcloud - Global einfache umsetzung Kurs unabhängige assets können geteilt werden Maximale Mesh größe Mehrere Meshes nötig Möglicherweise nicht im Mesh mit relevanten Peers Maximiert die Anzahl der Peers pro Mesh Wahrscheinlichkeit für selbes Subnetz eher gering Funktioniert auch mit wenigen Clients Ineffektiv wenn viele Clients vorhanden sind

- Schule Einfache Umsetzung Funktioniert auch mit wenigen Clients Relativ hohe trefferrate da schulen selten mehr als 1000 Schüler hat Max mesh größe um die 256(Chrome) Relativ wahrscheinlich gleiches Subnetz

- Kurs Hohe trefferrate Kleine Meshes benötigt mehr Clients Relativ wahrscheinlich gleiches Subnetz -> Daten die das bestätigen

hybride Ansätze: Zwei priorisierte Meshes Schule und Klasse

Berechnung eines Scores für jeden Peer: Möglichst viele gemeinsame kurse Liste von Kursen für jeden Peer Schnittmenge für jeden peer der online ist bilden Kardinalität der Menge = Score von Peer

Rechenaufwendig Aber machbar Beste Trefferrate Funktioniert wenn wenige und wenn viele Peers anwesend sind

4.1.2 Slidesync

Slidesync ist eine Plattform deren Nutzung stark durch die durchgeführten live Events dominiert wird. Ein Moderator erstellt das Event lädt die notwendigen Assets, z.B. Foliensätze, hoch. Live Events werden für eine bestimmte Zeit festgesetzt und Teilnehmer laden zum start des Events die Seite. Ein Großteil des entstandenen Traffics besteht aus HLS Videosegmenten. Jeder Teilnehmer eines Events benötigt die selben Inhalte.

Grafik visits

Grafik traffic

Die Peer Meshes in Slidesync werden als voll vermaschte Netze abgebildet. Da alle Teilnehmer eines Events zu großen Teilen die selben Daten benötigen können sie in dem selben Mesh untergebracht werden. Um zu gewährleisten das sich die Peers im Selben Subnetz befinden teilen sich nur solche ein Peer Mesh die sich in der Selben IP Range befinden. Ein weiterer wichtiger Factor ist der Kommunikationsoverhead der durch das halten von Verbindungen zu vielen Peers entsteht. Deshalb ist es nicht möglich bei größeren Events alle Peers im selben Peer Mesh unter zu bringen. Deshalb werden Sub Meshes gebildet in denen sich eine maximale Anzahl an Peers befinden können.

Abbildung 4 zeigt eine beispielhafte Aufteilung von Peer Meshes für ein Event. Für Netzwerk A und B werden jeweils zwei Meshes erzeugt und nur solche Clients werden miteinander verbunden die sich auch im Selben Subnetz befinden. Jedes Netzwerk wird wiederum in zwei Sub-Meshes unterteilt.

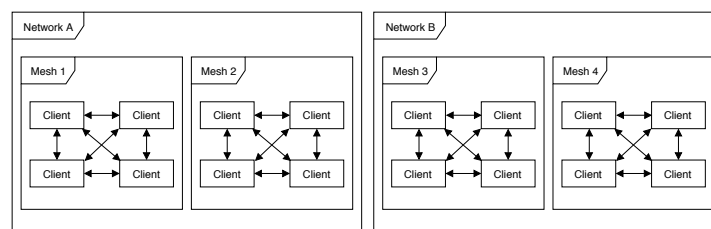


Figure 4: Peer to Peer Meshes - Slidesync

4.2 OBJEKT ZUORDNUNG - FINDEN VON RESSOURCEN

Um das Peer To Peer Netzwerk als CDN Nutzbar zu machen ist es wichtig das ein Peer in der Lage ist herauszufinden wer welche resource bereitstellt. Dazu lassen sich verschiedene Ansätze verfolgen.

DHT

Structured vs unstructured

distributed hash
tables erklären

Da das Routing von Ressourcen bei den Betrachteten Anwendungsfällen in einem Zeitkritischen Moment erfolgen muss wurde sich für einen anderen Ansatz entschieden. Jeder Peer hält eine Hashtabelle mit den Ressourcen seiner Peers vor. Fügt ein Peer eine neue Ressource zu seinem Cache hinzu oder entfernt sie muss er alle verbundenen Peers über diese Änderung informieren. Dadurch muss im Falle einer Anfrage nicht erst die Ressource im Netzwerk gesucht werden. Dies ist möglich durch die Struktur des Netzwerkes. Da nicht alle Peers miteinander verbunden sind sondern voll vermaschte sub-meshes gebildet werden ist es möglich alle relevanten Peers über Änderungen zu informieren. Dadurch ist es möglich die Rechenleistung für das auffinden von Ressourcen in einen weniger Zeitkritischen Moment zu verlagern. Jedoch hat dies zur Folge das die Meshes so gebildet werden müssen das die Peers möglichst viele Ressourcen gemeinsam benötigen. Ist eine Ressource nicht im Mesh vorhanden kann sie vom Server geladen werden.

4.2.1 *Subnetzerkennung*

4.2.2 *Struktur von IP Adressen*

eher Konzept

4.3 WIEDERVERWENDBARKEIT

4.4 OPEN SOURCE

IMPLEMENTIERUNG

5.1 ARCHITEKTUR

- Server architektur diagram -> Server, Stun, CDN, P2P CDN
- Plugin Architektur ->

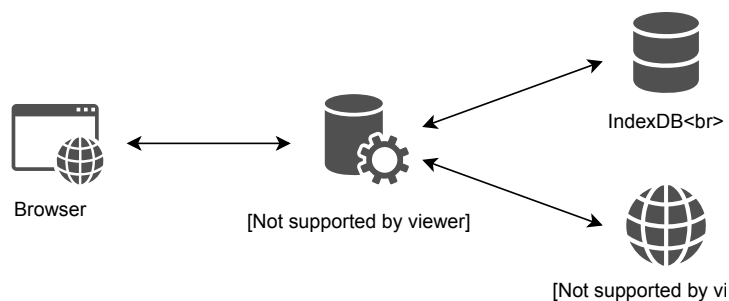


Figure 5: A Figure Title

5.1.1 Service worker

- warten bis sw active ist/alle verbindungen ready sind vs request normal durchgehen lassen bis alles fertig ist.
- Requests werden erst nach erfolgreicher registrierung behandelt
- clients.claim + skip waiting für den ersten aufruf (12)

5.1.2 Tests

- mocha
- chai
- karma
- event handler testen
- registrieren/abmelden -> test helper

5.2 RESSOURCEN MANAGEMENT

5.2.1 *Update Protokoll*

5.2.2 *Quota limits*

- benötigte speichermenge wird geschätzt
- headersize plus 'Content-Length' header wert
- solange löschen bis genug speicher frei ist

5.3 SIGNALING SERVER

The signaling server itself uses socket.io and can be found [here](#). The client ID is created there and is essential for the lifecycle of a peer in the whole network. It is not clear if the client IDs given by socket.io always have the same length. Therefore, client IDs will be padded to a maximal length of 24. This is necessary because the client IDs need to be sent via the binary datachannel and consequently, this requires a fixed length.

- websockets socket io
- gleicher data channel zur Bildung von meshes

5.4 MESSAGE PROTOCOL

5.4.1 *hash updates*

message types

5.5 DATA SERIALIZATION

- SCTP(DataChannel) message size limit 16kiB
- chunking reassembling nötig
- Performance einbussen
- abToMessage und sendToPeer

```

1 class ContextData < TransformStep
2
3   def transform(original_event, processing_units,
4     ↳ load_commands, pipeline_ctx)
5     processing_units.each do |processing_unit|
6       next if processing_unit[:in_context].nil? ||
7         ↳ processing_unit[:in_context][:user_agent].nil?
8
9       user_agent =
10        ↳ processing_unit[:in_context][:user_agent]
11
12       browser = Browser.new(user_agent)
13
14       processing_unit[:in_context][:platform]
15        ↳ = browser.platform.name
16       processing_unit[:in_context][:platform_version]
17        ↳ = browser.platform.version
18       processing_unit[:in_context][:runtime]
19        ↳ = browser.name
20       processing_unit[:in_context][:runtime_version]
21        ↳ = browser.version
22       processing_unit[:in_context][:device]
23        ↳ = browser.device.name
24     end
25   end
26 end

```

Listing 1: Some Code Snipped

EVALUATION

6.1 PREQUISITES

6.2 TECHINICAL EVALUATION

6.2.1 *Bandwidth*

6.2.1.1 *Simulierter Workload*

6.2.1.2 *Educational context*

6.2.1.3 *Live streaming in the coporate context*

6.2.2 *Nutzerzufriedenheit*

wie soll das aussehen?

6.3 BROWSER COMPATBILITY

6.3.1 *Browser Usage in coropate networks*

6.3.2 *Browser usage in educational networks*

6.4 SECURITY CONSIDERATIONS

6.5 DRM LICENCING

CONCLUSION

All in all, this is a nice thesis



AN APPENDIX

Some stuff here

BIBLIOGRAPHY

- [1] Ali Alabbas and Joshua Bell. “Indexed Database API 2.0.” In: *W3C Recommendation*. W3C, Jan. 2018. URL: <https://www.w3.org/TR/2018/REC-IndexedDB-2-20180130/>.
- [2] Adam Bergkvist, Daniel C. Burnett, Cullen Jennings, Anant Narayanan, Bernard Aboba, Taylor Brandstetter, and Jan-Ivar Bruaroey. “WebRTC 1.0: Real-time Communication Between Browsers.” In: *W3C Candidate Recommendation 27 September 2018*. W3C, Sept. 2018. URL: <https://www.w3.org/TR/webrtc/>.
- [3] Daniel C. Burnett, Adam Bergkvist, Cullen Jennings, Anant Narayanan, and Bernard Aboba. “Media Capture and Streams.” In: *W3C Candidate Recommendation*. W3C, Oct. 2017. URL: <https://www.w3.org/TR/2017/CR-mediacapture-streams-20171003/>.
- [4] I. Fette and A. Melnikov. “The WebSocket Protocol.” In: *Request for Comments: 6455*. Internet Engineering Task Force (IETF), Dec. 2011. URL: <https://tools.ietf.org/html/rfc6455>.
- [5] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs).” In: *Network Working Group*. The Internet Society, Mar. 2003. URL: <http://www.ietf.org/rfc/rfc3489.txt>.
- [6] Alex Russell, Jungkee Song, Jake Archibald, and Marijn Kruisselbrink. “Service Workers 1.” In: *W3C Working Draft, 2 November 2017*. W3C, Nov. 2017. URL: <https://www.w3.org/TR/service-workers-1/>.
- [7] Kurt Tutschku and Phuoc Tran-Gia. “Peer-to-Peer-Systems and Applications.” In: Jan. 2005.
- [8] Mingchen Zhao, Paarijaat Aditya, Ang Chen, Yin Lin, Andreas Haeberlen, Peter Druschel, Bruce Maggs, Bill Wishon, and Miroslav Ponec. “Peer-Assisted Content Distribution in Akamai NetSession.” In: *Proceedings of the 2013 conference on Internet measurement conference Pages 31-42*. ACM New York, NY, USA, Oct. 2013.

LIST OF FIGURES

Figure 1	A Figure Short-Title	9
Figure 2	A Figure Short-Title	11
Figure 3	A Figure Short-Title	12
Figure 4	A Figure Short-Title	13
Figure 5	A Figure Short-Title	15
Figure 6	A Figure Short-Title	17

LIST OF TABLES

LIST OF LISTINGS

Listing 1	Some Code Snipped	18
-----------	-----------------------------	----

DECLARATION OF ORIGINALITY

I hereby declare that I have prepared this master's thesis myself and without inadmissible assistance. I certify that all citations and contributions by other authors used in the thesis or which led to the ideas behind the thesis have been properly identified and referenced in written form. I also warrant that the above statement applies to the implementation of the project.

Hiermit versichere ich, dass ich die Masterarbeit selbständig und ohne unzulässige Hilfe Anderer angefertigt habe und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit besonders gekennzeichnet und die entsprechenden Quellen angegeben habe. Ich erkläre hiermit weiterhin die Gültigkeit dieser Aussage für die Implementierung des Projektes.

Potsdam, October 5th, 2016

Tim Friedrich