Hochschule für Telekommunikation Leipzig (FH)

Abschlussarbeit zur Erlangung des akademischen Grades

Bachelor of Engineering

Thema: Entwicklung einer BigBlueButton-basierten Anwendung zur Interaktion zwischen Dozent und Studierenden innerhalb dezentral gehaltener Vorlesungen

Vorgelegt von: Luise Kaufmann

geboren am: 05.12.1994
in: Leipzig

eingereicht am: 30. September 2016

Erstprüfer: Prof. Dr. rer. nat. Andreas Thor,
Hochschule für Telekommunikation Leipzig

Zweitprüfer: M. Eng. Michael Erler,
Hochschule für Telekommunikation Leipzig

Danksagung

Ich möchte mich an dieser Stelle bei meinen Prüfern für die sehr gute Betreuung bedanken.
Dabei danke ich in erster Linie Herrn Prof. Dr. Andreas Thor, der mich jederzeit in die richtige Richtung lenkte, für all die hilfreichen Hinweise und Anregungen, das Thema zu einer wissenschaftlichen Arbeit wachsen zu lassen.
Des Weiteren danke ich Herrn Michael Erler für die betriebliche Unterstützung und den wertvollen Input, meine Arbeit aus weiteren Blickwinkeln zu betrachten.

Ebenfalls möchte ich allen danken, die mich in den letzten drei Jahren durch mein duales Studium begleitet haben und mich fachlich und charakterlich zu der Person gemacht haben, die ich heute bin.

Ich danke allen Personen aus der Testgruppe, die ihre Freizeit für den Softwaretest geopfert haben.

Ein besonders großer Dank geht an meine Eltern und Tobias, die mich während der Bearbeitung unterstützt und den Rücken freigehalten haben, so dass ich intensiv Zeit hatte, mich mit dem Thema auseinander zu setzen.

DANKE!

Inhaltsverzeichnis

Abkürzungsverzeichnis

AJAX Asynchronous JavaScript and XML 53

API Application Programming Interface 20

App Application 23

APT Advanced Packaging Tool 34

ASCII American Standard Code for Information Interchange 54

BBB BigBlueButton 10

CSS Cascading Style Sheets 25

HfTL Hochschule für Telekommunikation Leipzig 10

HTML Hypertext Markup Language 20

HTTP Hypertext Transfer Protocol 36

HuP Hochschul- und Prüfungsamt 14

ID Identifikationsnummer 35

JSON JavaScript Object Notation 23

LDAP Lightweight Directory Access Protocol 58

LVS Lehrveranstaltungsstunden 12

MSC Message Sequence Chart 25

PDF Portable Document Format 55

PHP Hypertext Preprocessor 25

PSA Privacy and Security Assessment 21

QR Quick Response 50

RAM Random-Access Memory 30

SHA1 Secure Hash Algorithm 1 41

SMS Short Message Service 19

SWF Shockwave Flash 22

TCP Transmission Control Protocol 24

TIM Technology Innovation Management 20

TTs Teletutorien 20

URI Uniform Resource Identifier 26

URL Uniform Resource Locator 41

UTC Koordinierte Weltzeit 43

UTF8 8-Bit UCS Transformation Format 54

WebRTC Web Real-Time Communication 20

WLAN Wireless Local Area Network 27

XML Extensible Markup Language 41

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

In den letzten Jahren sind die Studierendenzahlen stark angestiegen. Die Ressourcen der Hochschulen und Universitäten wachsen allerdings in diesen Geschwindigkeiten nicht mit. Gerade kleine Einrichtungen, denen keine großen Räume zur Verfügung stehen, stoßen schnell an ihre Grenzen.

Auch an der Hochschule für Telekommunikation Leipzig (HfTL) gibt es für Studierendengruppen über 80 Personen kaum Möglichkeiten, Vorlesungen mit der gesamten Gruppe anzubieten. Folglich werden die Gruppen aus dieser Not heraus aufgeteilt und die Einheiten mehrfach gelesen. Das ist personalökonomisch gesehen unrentabel und steigert den Aufwand der Dozierenden.

In dieser Arbeit wird ein Weg aufgezeigt, wie Lehrveranstaltungen mit sehr großen Studierendengruppen über verteilte Vorlesungen an dezentralen Standorten abgebildet werden können, um vorhandene Ressourcen besser auszunutzen. Gleichzeitig wird eine Lösung zur Verfügung gestellt, um die zwangsläufig entstehenden Nachteile mittels bidirektionaler Kommunikation zwischen Dozent und Studierenden auszugleichen.

Dafür werden zuerst die aufkommenden Probleme erläutert, die mit immer größer werdenden Studierendengruppen auftreten. Darauf aufbauend werden Anforderungen an eine Lösung beschrieben und verschiedene Optionen aufgezeigt. Nach einer Entscheidung werden die Anforderungen an eine abgeleitete Systemlösung betrachtet. Anschließend wird die Entscheidung begründet, auf dem in der HfTL bereits in der Testphase befindlichen BigBlueButton (BBB)-System aufzusetzen. BBB wird beschrieben und der Entwicklungsstand – sowohl von BBB selbst, als auch der Implementierungsstand an der HfTL – festgehalten. Um die Anforderungen an eine Systemlösung zu erfüllen, werden davon schließlich zwei Konzepte abgeleitet, die es näher zu untersuchen gilt. Diese werden auf theoretische Vorteile, mögliche Probleme und technische Umsetzung hin verglichen. Um eine Entscheidung treffen zu können, welches Konzept entwickelt wird, folgt ein initialer Lasttest, der das Verhalten des Ausgangssystems untersucht und Aussagen über die Erfolgswahrscheinlichkeit beider Konzepte zulässt. Das ausgewählte Konzept wird als Anwendung entwickelt und ein erneuter Lasttest durchgeführt, um Aussagen über die Auslastung technischer Ressourcen treffen zu können. Abschließend wird ein Nutzertest durchgeführt und evaluiert, bei dem eine Testgruppe im Rahmen einer Probevorlesung die entwickelte Anwendung unter realistischen Bedingungen testet. Im Ergebnis hat die entwickelte Lösung das Potential, ein fester Bestandteil des Methodenkataloges der Dozierenden der HfTL zu werden. Jedoch besteht noch an einigen Stellen Optimierungsbedarf. Im Ausblick werden die Schritte aufgezeigt, die nötig sind, um die entwickelte Lösung als – sowohl von Studierenden als auch Dozierenden – akzeptierte Lehrmethode der Hochschule zu etablieren.

2 Problemstellung

Seit dem 14er Matrikel sind die dualen Studiengruppen an der HfTL so groß, dass sie ausschließlich im größten Hörsaal untergebracht werden können. Da jedes Jahr mindestens 2 duale Studiengänge starten, das Semester 3 Präsenzphasen beinhaltet, 3 Jahrgänge parallel unterrichtet werden, das Semester durchschnittlich 15 Wochen lang ist und nicht jede Woche für Präsenzphasen zur Verfügung steht, kommt es zu Gruppenkollisionen (vgl. [28]). Präsenzveranstaltungen können derzeit nur im direkten Kontakt zwischen Dozierenden und Studierenden abgehalten werden. Alle Personen befinden sich gemeinsam in einem Raum. Deshalb müssen seit mehreren Jahren einzelne Vorlesungen oder sogar ganze Semestergemeinschaften ausgelagert werden. Im Jahr 2016 sollen zusätzlich 232 Studierende[1] allein im dualen Bachelorstudiengang Wirtschaftsinformatik immatrikuliert werden. Dann reicht selbst der größte Hörsaal im Haus nicht mehr aus, um diese komplette Studiengruppe zu unterrichten.

Aber nicht nur kleine Hochschulen stoßen an ihre Grenzen. Seit 2008 steigen die Studierendenzahlen an Hochschulen gewaltig an. 2015 studierten deutschlandweit 42% mehr Studierende als noch im Jahr 2007. [1] Das bringt viele Probleme mit sich. Denn im Vergleich zu den Studienanfängern wachsen die zur akademischen Bildung gehörenden Ressourcen nur teilweise mit. Gerade die Ressourcen zur Befriedigung von Grundbedürfnissen passen sich nicht schnell genug an. Nach eigenen Beobachtungen konnte festgestellt werden, dass sich zu Pausenzeiten vor sanitären Einrichtungen lange Warteschlangen bilden, die sich bis zum Anfang der nächsten Veranstaltung nicht auflösen können. Ebenfalls ärgerlich ist es, wenn in Mensen und Cafeterien zur Mittagspause kein Sitzplatz frei ist oder Bäckereien und nahegelegene Imbisse, aufgrund von aufgebrauchten Kapazitäten, keine zeitnahe Verpflegung sichern können. Zusätzlich gibt es gesetzliche Grundlagen, die einem Anstieg von Studierendenzahlen entgegen stehen. Z.B. darf sich aufgrund von Statik und baulichem Brandschutz nur eine gewisse Anzahl an Personen gleichzeitig im Gebäude bzw. in Gebäudeteilen aufhalten. Zusätzlich müssen ausreichend große Fluchtwege zur Verfügung stehen. [26]

Auch ist der Stadtbau in gewissem Maße angehalten, bei wachsenden Studierendenzahlen zu handeln. Es sollten nicht nur genügend bezahlbare, gerade kleinere Wohnungen bzw. Studentenwohnheime vorhanden sein, auch die Infrastruktur sollte angepasst werden. Hauptverkehrsstraßen zu den Bildungseinrichtungen und der öffentliche Personennahverkehr müssen ausgebaut werden, damit es zu Stoßzeiten nicht zu gefährlichen Verkehrsbehinderungen oder Unfällen kommt.

Wenn an Hochschulen und Universitäten ein Gesamtanstieg an Studierenden zu verzeichnen ist, müssen auch in den einzelnen Veranstaltungen mehr Personen unterrichtet werden. Die Gruppengröße der Studiengänge steigt. Nach Schlenker und Beyer gibt es aus lernpsychologischer Sicht große Herausforderungen bei Vorlesungen mit hohen Teilnehmerzahlen. *„Eine Massenveranstaltung erlaubt nur geringen Kontakt zwischen den Lehrenden und Lernenden. Durch die damit verbundene Anonymität kann das Verantwortungsbewusstsein für das studentische bzw. eigene Lernen geringer ausgeprägt sein. Einfache Verhaltensregeln werden oft nicht eingehalten: Lehrende beklagen Störungen wie mangelnde Konzentrationsfähigkeit, themenfremde, laute Gespräche und willkürliches Kommen und Gehen."* [24, S. 4] Aus eigener Erfahrung stellen Studierende in Vorlesungen mit vielen Zuhörern kaum Fragen. Zum Einen kann

[1] offizielle Quote von Telekom Ausbildung an die HfTL

das am Selbstbewusstsein liegen, zum Anderen können die Fragenden akustisch nicht verstanden werden, da die Grundlautstärke im Hörsaal durch Nebengespräche sehr hoch ist und/oder der Fragende zu weit vom Dozenten entfernt sitzt. Durch den Mangel an Wortmeldungen aus den Reihen der Studierenden werden Vorlesungen immer unidirektionaler. Der Dozent spricht und die Studierenden hören zu. Dadurch fallen Motivation und Konzentration bei den Zuhörern, und der Dozent bekommt kein Feedback, um seine Veranstaltung zu steuern bzw. anpassen zu können. (vgl. [24, S. 3-5])

Ebenfalls kann bei Massenveranstaltungen nicht auf jeden Einzelnen eingegangen werden. Somit wird die Individualität des Lerners nicht berücksichtigt und Problemlösefähigkeiten sowie selbstständiges Denken können nicht gefördert werden (nach [24, S. 3]).

Daneben muss sichergestellt werden, dass Dozierende nicht die vom Staatsministerium festgelegten Obergrenzen an Lehrveranstaltungsstunden (LVS) zu je 45 Minuten bzw. Vorlesungen zu je 90 Minuten überschreiten. Im Bundesland Sachsen dürfen Professoren an Fachhochschulen beispielsweise nur neun Vorlesungen (18 LVS) in der Woche[2] und maximal drei (sechs LVS) am Tag halten. [17, S. 5] Wenn die Studierendenzahlen wachsen, muss beachtet werden, dass es nicht zu Termin[3]-, Raum[4]- oder Gruppenkollisionen[5] kommt.

Zusammenfassend lässt sich sagen, dass bei Massenlehrveranstaltungen die Qualität der Vorlesungen nachlassen. Zusätzlich wird die Planung immer schwieriger, da durch den Ansturm auf Hochschulen die Raumkapazitäten immer knapper werden.

[2] Durchschnittswert; über zwei Jahre gebildet
[3] Dozent ist für mehrere Vorlesungen zur gleichen Zeit verplant
[4] mehrere Gruppen sind mit unterschiedlichen Einheiten im selben Hörsaal geplant
[5] bei Ablaufplanung des dualen Studiums an der HfTL [28]

3 Lösungsansatz

3.1 Anforderungen an eine Lösung

Es soll eine Lösung in Form eines Prototyps entwickelt werden, damit die mit Massenlehrveranstaltungen einhergehenden Nachteile verringert werden können. Im besten Fall sollen alle Stakeholder profitieren: die Dozierenden, die Studierenden und die Verwaltung.

Für Dozierende soll der Mehraufwand dabei möglichst gering gehalten werden. Sie sollen aus dem Auditorium Rückmeldung erhalten und verschiedene Lehrmethoden nutzen können, um die Motivation bei den Studierenden aufrecht zu erhalten (vgl. [24, S. 5]). Um Rückmeldung zu erhalten und Lernkontrollen innerhalb der Vorlesung durchzuführen ist ein Feedbacksystem von Vorteil. Diese Systeme können *„auf die Probleme von Massenvorlesungen positiv[en] Einfluss [nehmen]"* [24, S. 7]. Schlenker und Beyer beschreiben unter Anderem *„eine höhere Beteiligung und Aufmerksamkeit der Studierenden, [...] eine aktivere Auseinandersetzung mit dem Lehrstoff durch die Studierenden [und] eine verbesserte Beteiligungsbereitschaft auch der weniger selbstbewussten Studierenden aufgrund der Anonymität der Abstimmungsergebnisse."* [24, S. 7]

Studierende sollen die Möglichkeit haben Fragen zu stellen und durch Feedback an den Dozierenden aktiv am Vorlesungsverlauf teilnehmen (z.B. durch Umfragen: „Sollen wir eine Übung rechnen oder mit dem nächsten Thema fortfahren?"). Wünschenswert ist eine Aufzeichnungsfunktionalität, so dass Studierende die Vorlesung im Nachgang reflektieren können. Um auf die soziale Komponente Bezug zu nehmen ist es nicht nur wichtig, den Dozenten zu sehen sondern auch den Austausch mit Kommilitonen nach den Veranstaltungen zu gewährleisten. Es ist nicht nur eine Verbesserung der Konzentration und Motivation zum Selbststudium wünschenswert sondern auch die Einhaltung der in Abschnitt 2 beschriebenen Verhaltensregeln. Studierende sollen untereinander die gleichen Inhalte und Schwerpunkte gehört haben, wenn sie an den Prüfungen teilnehmen. Deshalb ist von unterschiedlicher Behandlung abzusehen. Ebenfalls soll versucht werden, die verschiedenen Lerntypen individuell zu fördern. Nach Arrenberg und Kowalski gibt es vier verschiedene Lerntypen: visuell, auditiv, kommunikativ und motorisch. Der visuelle Typ lernt am besten über Fachliteratur. Das kann über das Lesen von Texten und Betrachten von Abbildungen bis hin zum Ansehen von Lernvideos reichen. Der auditive Typ ist in der Vorlesung bestens aufgehoben, denn er lernt über Zuhören. Der kommunikative Typ braucht den Austausch mit Anderen über die Fachthemen. Das kann in Form einer bidirektionalen Unterhaltung bzw. Erklärungen erfolgen. Der motorische Lerntyp ist in Vorlesungen am schlechtesten zu aktivieren. Er lernt am besten durch Ausprobieren, Bewegungen und eigens durchgeführte Versuche. (nach [2])

Hochschulverwaltungen betrachten Vorlesungen aus einer finanz- und personalökonomischen Perspektive. Das große Ziel besteht darin, den Input zu minimieren und den Output zu maximieren (vgl. Ökonomisches Prinzip). Somit müssen bei der Lösung die Ressourcen minimiert werden und dennoch qualitativ ähnliche (im besten Fall bessere) Vorlesungen angeboten werden. Ressourcen sind im konkreten Beispiel „Vorlesung" die finanziellen Mittel, Raumres-

sourcen[6], Personalressourcen[7] und Zeit[8]. Die Ressourcen stehen untereinander in Abhängigkeit. Z.B. steht – bei gleicher Beschäftigung des Lehrpersonals (Personalressource →) – ein Hörsaal aufgrund von Renovierungsarbeiten temporär nicht zu Verfügung (Raumressourcen ↓) muss die Verwaltung mehr Zeit aufbringen (Zeitressource ↑), um die Stundenpläne zu erstellen[9] oder andere Räume anzumieten (Finanzressource ↑).

3.2 Lösungsmöglichkeiten

Da die Raumressourcen der HfTL nicht für Massenlehrveranstaltungen ausgelegt sind, müssen andere Wege gefunden werden, Vorlesungen für alle Studierenden anzubieten. Die derzeit genutzte Lösungsmöglichkeit besteht darin, die Gruppen zu teilen und die Vorlesungen von einem Dozenten mehrfach halten zu lassen (vgl. Tabelle 1, Spalte *„Vorlesungen mehrfach halten – gleicher Dozent"*). Das kann sich negativ auf die Studierenden auswirken, da in einer Veranstaltung spannende Fragen gestellt werden können, von denen dann nur die aktuelle Gruppe profitieren kann. Außerdem kann man nicht sichergehen, dass die Dozierenden die exakt identischen Veranstaltungen lesen. Immer wieder kann es zu kleinen Änderungen kommen, in denen dann das entscheidende Prüfungsthema unterschiedlich intensiv behandelt wird (empfundene Prüfungsgerechtigkeit →). Außerdem ist die Mehrbelastung für die Dozierenden immens. Je nachdem wie oft die Gruppen geteilt werden, verdoppeln/verdreifachen/vervierfachen... sich die zu haltenden Präsenzeinheiten (Mehraufwand für Dozierende ↑). Dabei dürfen die gesetzlichen Rahmenbedingungen für die Gesamtanzahl an Lehrveranstaltung pro Dozent nicht überschritten werden. Außerdem muss gewährleistet werden, dass alle Veranstaltungen für alle Gruppen geplant sind. Um alle Einheiten zu organisieren, muss die Verwaltung viel Zeit aufbringen (Ressource – Zeit ↑).

Um den Mehraufwand für jeden einzelnen Dozenten gering zu halten ist es möglich, die Gruppen geteilt und jede Gruppe von einem anderen Dozenten unterrichten zu lassen (vgl. Tabelle 1, Spalte *„Vorlesungen mehrfach halten – mehrere Dozenten"*). Das erspart den Verwaltungsangestellten viel Zeit bei der Stundenplanerstellung (Ressource – Zeit ↓). Allerdings resultiert daraus ein höherer Beschäftigungsgrad an Lehrkräften (Ressource – Personal ↑), deren Gehälter, Versicherungen, Büroeinrichtungen, Weiterbildungen etc. bezahlt werden müssen (Ressource – Finanzen ↑). Außerdem wird diese Lösungsmöglichkeiten für Studierende im Hinblick auf die Prüfungsvorbereitung[10] nicht als gerecht empfunden, da jeder Dozent das Modul anders auslegt und somit andere Schwerpunkte setzt (empfundene Prüfungsgerechtigkeit ↓).

Eine weitere Alternative um den Mehraufwand für Dozierende gering zu halten besteht in der Aufzeichnung von Vorlesungen im Vorfeld und deren Vorführung im Anschluss (vgl. Tabelle 1, Spalte *„Vorlesungsaufzeichnung"* [19]). Allerdings gibt es dabei keinen synchronen Rückkanal zum Dozierenden (Rückmeldung aus dem Auditorium ↓). Somit können weder Fragen gestellt (Fragen stellen ↓) noch Umfragen durchgeführt (Feedbacksystem ↓) werden.

[6] Gespräch mit Hochschul- und Prüfungsamt (HuP) der HfTL: große Räume sind „wertvoller"; kleine Gruppen können bei Engpässen in große Räume, Gegenteil nicht möglich

[7] Lehrpersonal

[8] Zeit der Verwaltungsmitarbeiter um Semesterplan anzufertigen

[9] Zeitslots enger füllen unter der Bedingung, dass es nicht zu Raumkollisionen kommt

[10] unter der Voraussetzung, dass alle die gleiche Prüfung schreiben

Aufgrund der Anonymität können die Studierenden zudem nur schwer für das weitere Selbststudium motiviert werden (Motivation ↓). Der große Vorteil für die Studierenden liegt in der Verfügbarkeit. Wenn das Thema beim erstmaligen Hören noch nicht verstanden wurde, kann im Nachgang die Aufzeichnung erneut wiedergegeben werden (Wiederholung ↑). Zwar sind die Produktionskosten zur Erstellung der Aufzeichnung sehr hoch, die Wiederverwendbarkeit ohne erneuten personellen Aufwand reguliert die Kosten allerdings wieder (Ressourcen – Finanzen →). Da keine Dozierenden bei der Wiedergabe des Videos nötig sind (Mehraufwand für Dozierende ↓) und die Aufnahmen jederzeit verfügbar sind, ist der organisatorische Zeitaufwand zur Stundenplanung sehr gering (Ressource – Zeit ↓).

Bei der verteilten Vorlesung (vgl. Tabelle 1, Spalte *„Verteilte Vorlesung"* [7]) befinden sich die aufgeteilten Studierendengruppen in unterschiedlichen Hörsälen. In einem Hörsaal ist der Dozent, dessen Veranstaltung über ein Videokonferenzsystem in die restlichen Hörsäle übertragen und zusätzlich aufgezeichnet (Wiederholung ↑) wird. In allen Veranstaltungsräumen haben die Studierenden die Möglichkeit, über ein System dem Dozenten Fragen zu stellen (Fragen stellen ↑) und an Umfragen teilzunehmen (Feedbacksystem ↑). Das besondere an dieser Lösungsmöglichkeit ist das Ansprechen jedes Lerntyps. Die Aufzeichnung hilft auch dem motorischen Lerntypen (Lerntyp – motorisch →), im Nachhinein die Veranstaltung zu reflektieren und die Inhalte zu verinnerlichen, weil er hier die Veranstaltung pausieren kann, um z.B. die gezeigten Algorithmen nachzuprogrammieren. Zwar werden zur Betreuung der Übertragung Assistenten benötigt, aber im Vergleich zu *„Vorlesungen mehrfach halten – mehrere Dozenten"* kann hier auch auf studentische Hilfskräfte oder ähnliches Personal zurückgegriffen werden, die zur Kostensenkung beitragen (Ressource – Finanzen →).

Der Engpass an großen Veranstaltungsräumen kann auch über zusätzliche Anmietung von Hörsälen entschärft werden (vgl. Tabelle 1, Spalte *„Anmietung"*). Somit müssen die Semestergemeinschaften nicht geteilt werden und können zusammen unterrichtet werden. Allerdings gibt es für Dozierende An- und Abfahrtszeiten, die durch die Verwaltungsmitarbeiter aufwändig geplant werden müssen (Ressource – Zeit ↑). Das hält den Mehraufwand für Dozierende mäßig. Zwar können die Studierenden Fragen stellen und an Umfragen teilnehmen (Feedbacksystem ↑) – was dem Dozenten ein gewisses Feedback gibt (Rückmeldung aus dem Auditorium ↑) – jedoch kann aufgrund der großen Teilnehmerzahl die Gruppe nur bedingt für das weitere Selbststudium motiviert werden (Motivation →, vgl. [24][27]). Die eigenen Raumressourcen werden nicht stark beansprucht (Ressource – Raum ↓), dafür steigen die Kosten für die Anmietung von externen Möglichkeiten (Ressource – Finanzen ↑).

Der geringste Ressourcenbedarf an Raum, Personal und Zeit entsteht, wenn die Vorlesungen komplett über eLearning Technologien abgebildet werden (vgl. Tabelle 1, Spalte *„komplett eLearning"*). Die Studierenden haben keine Präsenzveranstaltungen innerhalb der Hochschule. Das hält den Mehraufwand für Dozierende gering. Allerdings können die Studierenden nur noch wenig zum Selbststudium motiviert werden (Motivation ↓), da die soziale Komponente in den Veranstaltungen fehlt (Austausch mit Kommilitonen ↓, Lerntyp – kommunikativ ↓, vgl. [24, S. 4]) und ein Medienwechsel der Dozierenden nur bedingt möglich ist (Medienwechsel →, vgl. [24, S. 5]). Zwar kann auf unterschiedliche Medien (Präsentation, Videos, Desktopsharing, etc.) zurückgegriffen werden, jedoch erscheinen alle Medien bei den Studierenden gleich – in einer Art „Mediencontainer" – in Form eines Videostreams.

Tabelle 1: Vergleich von Lösungsmöglichkeiten

Kriterien	**Vorlesungen mehrfach halten – gleicher Dozent**	**Vorlesungen mehrfach halten – mehrere Dozenten**	**Vorlesungs-aufzeich-nung**	**Verteilte Vorlesung**	**Anmietung**	**komplett eLearning**
Mehraufwand Dozierende	hoch	niedrig	niedrig	mittel	mittel	niedrig
Rückmeldung aus dem Auditorium	ja	ja	nein	ja	ja	ja
Medien-wechsel	ja	ja	teilweise	ja	ja	teilweise
Motivation	ja	ja	nein	ja	teilweise	nein
Feedback-system	ja	ja	nein	ja	ja	ja
Fragen stellen	ja	ja	nein	ja	ja	ja
Dozent sichtbar	ja	ja	teilweise	ja	ja	teilweise
Austausch mit Kommili-tonen	ja	ja	teilweise	ja	ja	nein
empfundene Prüfungsge-rechtigkeit	teilweise	nein	ja	ja	ja	ja
Wiederholung	nein	nein	ja	ja	nein	ja
Lerntyp – visuell	teilweise	teilweise	teilweise	teilweise	teilweise	teilweise
Lerntyp – auditiv	ja	ja	ja	ja	ja	ja
Lerntyp – kommunika-tiv	teilweise	teilweise	nein	teilweise	teilweise	nein
Lerntyp – motorisch	nein	nein	teilweise	teilweise	nein	teilweise
Ressource – Finanzen*	mittel	hoch	mittel	mittel	hoch	mittel
Ressource – Raum*	mittel	mittel	mittel	mittel	niedrig	niedrig
Ressource – Personal*	mittel	hoch	niedrig	mittel	niedrig	niedrig
Ressource – Zeit*	hoch	niedrig	niedrig	mittel	hoch	niedrig

*Die Wertungen sind im Verhältnis zueinander betrachtet

Um sich für eine Lösungsmöglichkeit zu entscheiden, muss man die Kriterien abwiegen. Bildungseinrichtungen verfügen *„teilweise [über] unzureichende personelle und finanzielle Ausstattung"* [24, S. 4] Deshalb sind die Ansätze *„Vorlesungen mehrfach halten – mehrere Dozenten"* und *„Anmietung"* für ein langfristiges Szenario ungeeignet.

Aus personalökonomischer Perspektive macht es ebenso wenig Sinn, die Vorlesung von einem Dozenten mehrmals lesen zu lassen. Die Zeit, die für die erneuten Lesungen verloren geht, fehlt den Dozenten in anderen wichtigen Aufgabenbereichen, z.B. der Forschung. Daher kann dieser Weg nicht zukunftsfähig sein.

Eine Lösungsanforderung ist, dass die Nachteile einer Massenlehrveranstaltung verringert werden. Bei der Vorlesungsaufzeichnung werden die Nachteile nicht verringert sondern eher noch erweitert. Es gibt keine Rückmeldung aus dem Auditorium an den Dozierenden, der eingesetzte Medienwechsel wird durch den „Mediencontainer" entkräftet, die Motivation zum Selbststudium kann nicht entfacht werden, und es können keine Fragen gestellt und Lernkontrollen (Umfragen) durchgeführt werden. Aus den genannten Gründen entfällt dieser Lösungsansatz.

Bei der Variante *„komplett eLearning"* ist das Verhalten ähnlich. Die Studierenden werden im Rahmen des immer gleichen „Mediencontainers" ungenügend motiviert, und die sozialen Interaktionen sowohl zwischen Dozent und Student als auch zwischen Studierenden fehlen. Außerdem ist die Gefahr groß, dass sich Studierende bei mobil empfangbaren Vorlesungen vom Hörsaal fern halten, was sich nach Kerres und Preußler nachteilig auf das Verständnis der Lehrinhalte auswirkt (vgl. [14], S. 87).

Nach Abwägung der Kriterien in Tabelle 1 stellt der Ansatz der *„Verteilten Vorlesung"* einen soliden Mittelweg dar. Die Vorteile einer „klassischen" Vorlesung (Rückmeldung aus dem Auditorium, Medienwechsel, Motivation, Feedbacksystem, Fragen stellen, Dozent sichtbar, Austausch mit Kommilitonen und empfundene Prüfungsgerechtigkeit) werden beibehalten. Durch das Übertragungssystem kann zusätzlich eine Aufzeichnung angefertigt werden. Das kann gerade dem visuellen und motorischen Lerntyp entgegen kommen. Die Ressourcen werden alle nur mäßig beansprucht.

3.3 Anforderungen an ein System

Wie im oberen Abschnitt beschrieben wird ein System gesucht, das Vorlesungen übertragen kann und einen Rückkanal zur Verfügung stellt. Zusätzlich soll die Übertragung aufgezeichnet und Studierenden zu Verfügung gestellt werden. Tabelle 2 zeigt eine Zusammenstellung aller gewünschten Funktionalitäten, damit ein abwechslungsreicher Medienwechsel für eine motivierende Lehrveranstaltung durchgeführt werden kann. Alle Funktionen soll ein einziges System abdecken, damit die Bedienung für die Nutzer einfach bleibt und es zu möglichst wenig Verzögerungen kommt. Wünschenswert ist die Erweiterbarkeit des Systems, damit bei Bedarf neue Funktionen entwickelt werden können.

Tabelle 2: Funktionsanforderungen des Systems

Medien	**Vorlesungsübertragung Hinkanal**	**Ansicht der Studierenden Rückkanal**
Audio	ja	nein
Präsentationsfolien	ja	nein
Whiteboard	ja	nein
Application Sharing	ja	nein
Chat	ja	ja
Feedbackemojis	nein	ja
Umfragen	ja – erstellen	ja – teilnehmen
Video	ja	nein

Um der Vorlesung zu folgen (Hinkanal) reicht die Hörsaalübertragung aus. Damit keine Zusatzkosten für Nutzerhard- und Software entstehen, sollen Interaktionen in Richtung des Dozenten (Rückkanal) über die privaten Endgeräte der Studierenden, wie Smartphones oder Notebooks, laufen, ohne dabei Programme bzw. Apps installieren zu müssen. Die Anwendung auf studentischer Seite soll schmal gehalten werden, damit das System nicht von der eigentlichen Vorlesung ablenkt und die Nutzerfreundlichkeit erhöht wird (vgl. [18], S. 83 Sf.). Die bevorzugten Betriebssysteme sind Android und iOS.

Damit die Studierenden die Anwendung mit allen Funktionen (siehe Tabelle 2) nutzen können, ist eine Anmeldung erforderlich. Hierbei werden der Name des Benutzers und die Veranstaltung abgefragt, da es passieren kann, dass mehrere Vorlesungen gleichzeitig übertragen werden. Die Veranstaltung soll über einen Radiobutton ausgewählt werden. Zusätzlich muss der Nutzer durch einen Sicherheitsmechanismus beweisen, dass er berechtigt ist, den Rückkanal der Vorlesung zu nutzen. Das kann über ein Passwort abgebildet werden, das nur Studierende der verteilten Vorlesung wissen. Nach dem Login sollen die Nutzer über ein Menü entscheiden welche Aktion sie ausführen wollen. Nach Tabelle 2 ist es sinnvoll, die Aktionen „Umfrageteilnahme", „Feedbackemojis" und „Chat" anzubieten. Der Chat kann nochmals in „lesen" und „schreiben" unterteilt werden. Bei laufenden Umfragen sollen unter „Umfrageteilnahme" alle Antwortmöglichkeiten in Form von Buttons angezeigt werden. Durch Klick auf einen der Buttons wird die Antwortmöglichkeit ausgewählt und über den Rückkanal zum Dozenten übermittelt. Unter „Feedbackemojis" sollen Studierende ihren aktuellen Zustand anzeigen. Mögliche Kriterien sind: „Vorlesung zu schnell", „Vorlesung zu langsam", „Lachen", „Klatschen" oder „Ja" und „Nein", um somit schnell Rückmeldung auf eine geschlossene Frage zu geben. Damit bekommen Dozierende noch mehr Feedback von studentischer Seite. Obwohl der Chat über den Hinkanal im Hörsaal über den Beamer angezeigt wird, sollen die Studierenden die Möglichkeit haben, auf die Nachrichten zurückzugreifen, die am Anfang der Sitzung geschrieben wurden und durch das automatische Scrollen schon verschwunden sind. Unter dem Menüpunkt „Chat lesen" sollen deshalb alle Chatnachrichten der Vorlesung angezeigt werden. Bei „Chat schreiben" soll es gerade für die Studierenden im entfernten Hörsaal die Möglichkeit geben dem Dozenten eine Nachricht zu schreiben. Hilfreich ist es hier, die letzten geschrieben Nachrichten aus dem Chat mit anzeigen zu lassen, damit in der eigenen Nachricht darauf Bezug genommen werden kann. Um einen sozialen Bezug zu den Nachrichten herzustellen ist es hilfreich, wenn

zu jeder Nachricht der eingegebene Name aus dem Loginfenster angezeigt wird.

Auf dem Markt gibt es kostenlos nutzbare Software, die einen Teil der geforderten Funktionen abdeckt. Z.B. kann das Feedbacksystem invote[11] kostenlos von Dozierenden verwendet werden, um Umfragen innerhalb der Vorlesung durchzuführen. Die Teilnahme erfolgt über Short Message Service (SMS) oder eine Webseite. Der Nachteil ist, dass diese Systeme keine Komplettlösungen darstellen. Zwar kann man für jeden Verwendungsfall aus Tabelle 2 Teillösungen finden – z.B. Whiteboard über OneNote[12], für Audio eine Festnetztelefonverbindung usw. – diese machen die Verwendung aber unnötig kompliziert, da die Bedienung von mehreren Systemen gleichzeitig sehr anstrengend ist und mehr potenzielle Fehlerquellen entstehen können. Im Rahmen einer Bachelorthesis [25] wurden vorhandene Konferenzsysteme verglichen. Kommerzielle Systeme bringen oft einen hohen Lizenzkostenaufwand mit sich. Die Erweiterbarkeit ist nur bedingt möglich. Zusätzliche Features, z.B. der funktionsreduzierte Rückkanal, müssen teuer eingekauft werden (vgl. [25]). Nach Siebert bietet es sich an Open Source Software[13] zu nutzen. Der Vorteil ist, dass keine Lizenzkosten anfallen und die Software verändert und nach eigenen Bedürfnissen angepasst werden kann. Das 2014 an der HfTL getestete Videokonferenzsystem BigBlueButton (BBB)[14] bietet von vornherein den kompletten Funktionsumfang des Hinkanals (vgl. Tabelle 2). Zudem gibt es für den Rückkanal ebenfalls fertige Konzepte. Auch wenn diese Konzepte nicht identisch mit den Vorstellungen aus Tabelle 2 sind, ist die Erweiterbarkeit des Systems einfach, da die Open Source Software sehr gut dokumentiert ist und die Community bei Problemen schnell weiterhilft (vgl. [4][5][16, S. 1337]). Aufgrund der enormen Erleichterung, dass der Hinkanal schon komplett fertig ist, erweist es sich als unwirtschaftlich, ein System von Grund auf neu zu entwickeln. Neun Jahre Entwicklererfahrung und das Know-how eines ganzen Teams sind bisher in das Projekt geflossen. Ein Nachbau bzw. Neubau der Software ist wirtschaftlich gesehen nicht sinnvoll. Auf diesen Grundlagen fällt die Entscheidung auf die Weiterentwicklung von BBB.

[11] `http://invote.de/`
[12] `https://www.onenote.com/`
[13] frei zugänglich, meistens kostenfrei, eigene Anpassungen des Quelltextes möglich (vgl. [15, S. 1, 15])
[14] `http://bigbluebutton.org/`

4 BigBlueButton als Lösungsansatz

4.1 Entstehung von BigBlueButton

Die Open Source Software BBB wird seit 2007 entwickelt und ist unter der GNU Lesser General Public License[15] verfügbar. Der Quellcode ist auf GitHub[16] zugänglich.

Absolventen der Carleton University in Ottawa starteten im Rahmen eines Technology Innovation Management (TIM)-Programms die Entwicklung eines Web-Konferenz-Systems. Schon damals war es das Ziel, jedem, der einen Webbrowser besitzt, Zugang zu qualitativ hochwertiger Bildung zu ermöglichen (vgl. [6]). Damit einhergehend musste eine Lösung gefunden werden, dass alle Teilnehmer an Vorlesungen teilnehmen können, ohne physisch vor Ort sein zu müssen. Die Idee von BigBlueButton entstand.

Seit 2008 ist BBB frei verfügbar, so dass auch andere Nutzer die Software auf eigenen Servern installieren können. Seitdem konnte die Community von Entwicklern, Administratoren und Nutzern wachsen (vgl. [6]). Die frühesten BBB-Versionen basierten lediglich auf Red5[17]- und anderen Open Source Komponenten (vgl. [6]). Nach und nach wurden mit jeder neuen Version auch neue Funktionen entwickelt. Wichtige Neuerungen sind Desktopsharing[18], das Application Programming Interface (API)[19], die Aufzeichnungs- und Wiedergabefunktion, Web Real-Time Communication (WebRTC) Audio, Feedbackemojis und das integrierte Umfragetool. Seit dem 17. Mai 2016 ist BBB 1.0 verfügbar. Ab dieser Version gibt es die Möglichkeit, zusätzlich zum Flash-Client einen Hypertext Markup Language (HTML)5 Client zu nutzen (vgl. [4]).

4.2 Einsatz als Videokonferenzsystem an der HfTL

Im Rahmen einer Bachelorthesis [25] an der HfTL wurde bereits 2014 die Integration des Videokonferenzsystems BBB in die Infrastruktur der Hochschule untersucht und dokumentiert. Da sich das Produkt von kommerziellen Videokonferenzsystemen unter anderem durch eine einfache Administration und schnelle Erweiterbarkeit abhebt (vgl. [25, S. 14-17, 29-35,47-48]), wurden im Vorfeld dieser Arbeit weitere Tests durchgeführt, um auch die Benutzerfreundlichkeit zu eruieren. Mit der damals getesteten Version 0.81 gab es noch viele Fehler und eine unzureichende Sprachqualität(vgl. [3]). Seit Mai 2016 arbeitet die Hochschule mit der stabilen BBB-Version 1.0, die nun auch WebRTC unterstützt.

Hauptsächlich wurden Teletutorien (TTs) über das Konferenzsystem getestet. Dafür wurden einige Dozenten der Hochschule in eine Testgruppe aufgenommen, und die Studierenden konnten Bugreports[20](vgl. [3]) erstellen. Diese halfen Fehler aufzudecken, Verbesserungsvorschläge einzureichen oder Feedback zu geben. Zusätzlich konnten Studierende BBB nutzen, um sich innerhalb von Projektgruppen auszutauschen und dezentrale Meetings durchzuführen. Seit 2016 wird der Vorkurs ebenfalls über BBB abgebildet.

[15] `http://www.gnu.org/licenses/lgpl.html`
[16] Plattform zur Softwareentwicklung, `https://github.com/`
[17] Open Source Streaming Server, `http://red5.org/`
[18] Bildschirmübertragung
[19] Programmierschnittstelle
[20] ähnlich Ticketsystem

4.3 Stand der Implementierung an der HfTL

Eine komplette Umstellung auf das Open Source Produkt erfolgte bislang noch nicht, da die Hochschule über unzureichende personelle Ressourcen verfügt, um das erforderliche Service Level Agreement[21] inklusive Ausfallsicherheit der Dienste zu garantieren. Zusätzlich müsste ein langwieriges und komplexes Privacy and Security Assessment (PSA)-Verfahren der Deutschen Telekom durchlaufen werden, um Sicherheit und Datenschutz nach Konzernrichtlinien zu gewährleisten (vgl. [8]). Infolge dessen befindet sich die BBB-Installation der HfTL noch in einer experimentellen Testumgebung mit der aktuellsten BBB-Version 1.0. Es gibt kein Service Level Agreement und somit auch keine garantierte Ausfallsicherheit. Der Dienst wird hausintern gehostet[22] und administriert. Alle Entwicklungsschritte und Änderungen werden auf einer virtuellen Maschine durchgeführt, die nicht für Nutzertests verwendet wird. Erst wenn sich Neuerungen als positiv erweisen, findet eine Migration auf das System statt, auf das Nutzer einer Testgruppe Zugriff haben.

Eine Videokonferenz inklusive der Übertragung von einem Whiteboard ist schon jetzt mit dem System möglich. Auch die Chat- und Umfragefunktionen sind implementiert. Die Studierenden können sowohl mit ihrem Notebook als auch mit dem Smartphone – mit Hilfe des HTML5-Clients – an der Sitzung teilnehmen. Allerdings stellen beide Möglichkeiten einen sehr großen Funktionsumfang zur Verfügung, der für das beschriebene Szenario (vgl. Abschnitt 2) nicht erforderlich ist.

[21] Dienstgütevereinbarung
[22] bereitgestellt

4.4 Architektur von BigBlueButton

BBB ist ein Client-Server System, das sich aus verschiedenen Open Source Komponenten, wie z.B. Redis, FreeSwitch und Red5, zusammensetzt. Der Standardclient ist in Actionscript geschrieben. Die von den BBB-Entwicklern erstellten Serverkomponenten basieren auf Java, Grails und Scala.

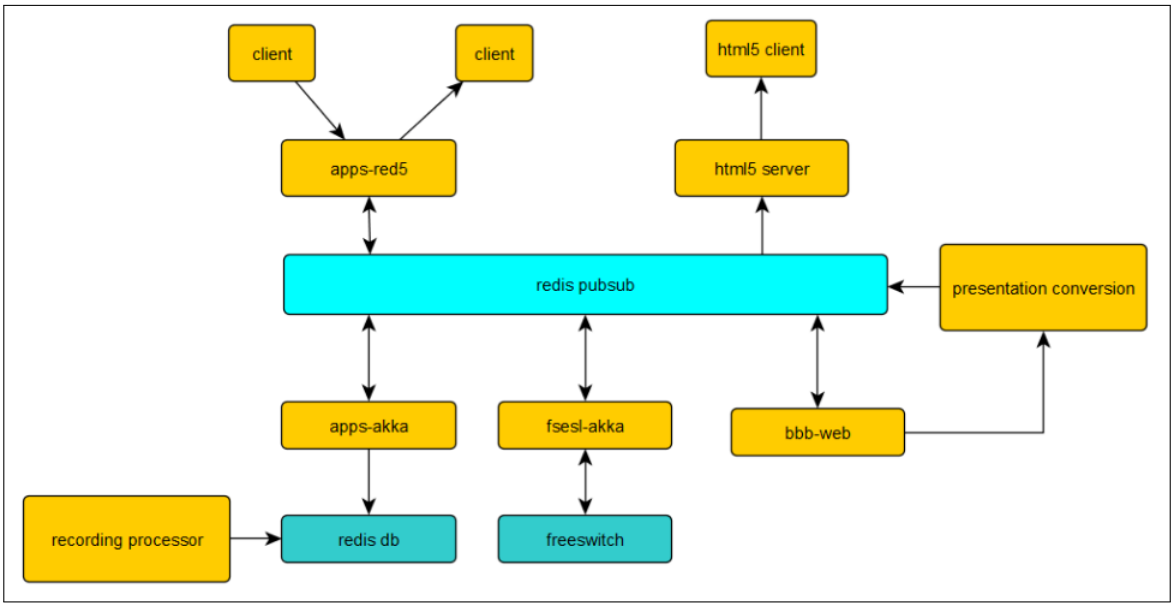

Abbildung 1: BigBlueButton 1.0 Architektur [5]

In Abbildung 1 ist eine Architekturskizze von BBB 1.0 zu sehen. Im Browser (Client) läuft eine Flash-Anwendung, die sich zum Red5 verbindet. Red5 ist ein Medien-Streaming Dienst, der die Clients mit der eigentlichen BBB-Anwendung verbindet. Dabei werden Informationen über Voice, Video, Deskshare und Apps (User, Chat, Whiteboard und Präsentationen) verwaltet und zu den Clients ausgeliefert. Parallel zum Flash-Client kann der HTML5-Client verwendet werden. Dieser ist über den HTML5-Server (Meteor[23]) mit dem Redis PubSub[24] verbunden.

BBB-web ist besser bekannt als BBB-API, worüber z.B. Sitzungen erstellt oder Präsentationsfolien hochgeladen werden können. Präsentationsfolien müssen bei „presentation conversion[25]" in Shockwave Flash (SWF) umgewandelt werden. Dafür werden LibreOffice, SWFTools und ImageMagick/Ghostscript verwendet.

Im Audiomixer FreeSwitch[26] werden die einzelnen Audiospuren der Nutzer gemultiplext.

Die RedisDB speichert nicht nur alle Sitzungsdaten sondern verwaltet auch die Aufnahmen. Aufnahmen werden über den „recording processor" kompiliert.

Der zentrale Kommunikationskanal Redis PubSub verbindet die verschiedenen serverseitigen Anwendungen. [5]

[23] Web-Framework, `https://www.meteor.com/`
[24] Publish and Subscribe (veröffentlichen und abonnieren)
[25] Präsentationsumwandlung
[26] `https://freeswitch.org/`

4.5 Architektur des HTML5-Clients

Der seit BBB-Version 1.0 mitgelieferte HTML-5 Client befindet sich zum Zeitpunkt der Erstellung dieser Arbeit noch im Experimentalstatus und kann separat zum Flash-Client verwendet werden. Zur Nutzung muss ein Script ausgeführt werden. Dieses startet den Meteor-Prozess, der die HTML5-Application (App) kompiliert und bereitstellt. Meteor kann von nun an die Kommunikation, die über den Redis PubSub läuft, mithören (vgl. Abbildung 2).

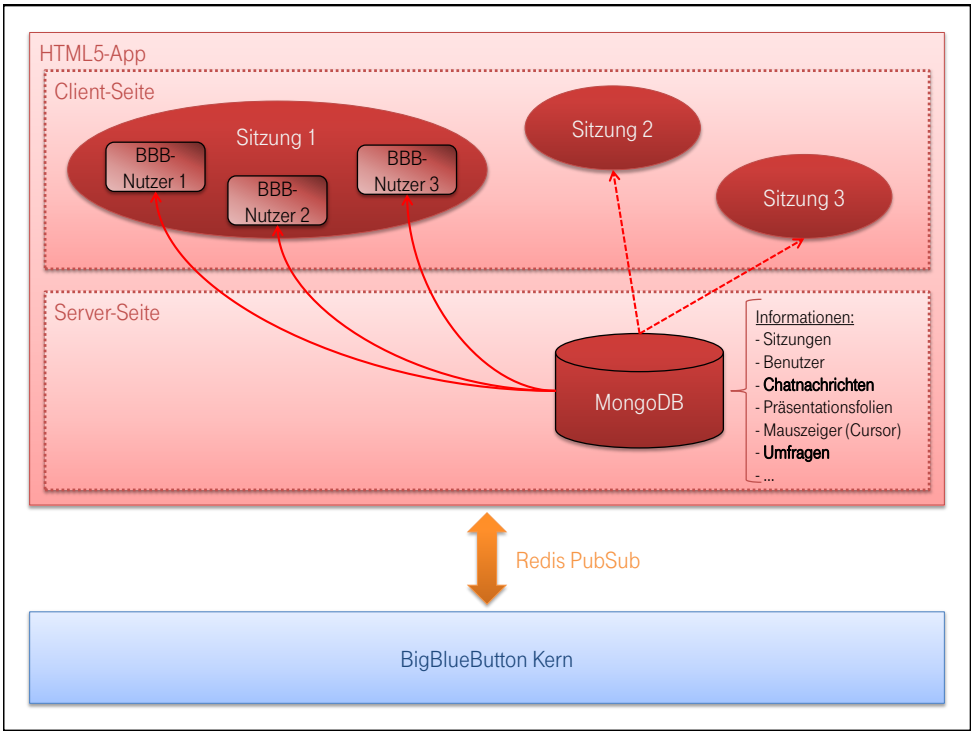

Abbildung 2: Funktionsweise mitgelieferter HTML5 Client, nach [4]

Eine initiale JavaScript Object Notation (JSON) Nachricht „get_all_meetings_request" wird über Redis PubSub gesendet. Die Antwort – „get_all_meetings_reply" enthält Informationen über alle derzeit laufenden BBB-Sitzungen, die eingeloggten Benutzer, Chatnachrichten, Präsentationsfolien, Mauszeiger und Umfragen. Alle Informationen werden in der MongoDB[27] gespeichert. Neue Events[28] und Heartbeats[29] werden ebenfalls über den Redis Channel[30] mitgeteilt und in der MongoDB gespeichert. (vgl. [4])

Wenn ein Nutzer über den HTML5-Client der Sitzung beitritt, werden alle für ihn bestimmten und zur Sitzung gehörenden Informationen aus der MongoDB gelesen und an den Client weitergereicht (siehe Abbildung 2). Dieser speichert alle Informationen in MiniMongo[31].

[27] NoSQL-Datenbank, eng verbunden mit Meteor, `https://www.mongodb.com/de`
[28] Aktionen
[29] periodisches Signal, Verbindung ist noch aktiv
[30] Kanal
[31] Clientseite zu MongoDB

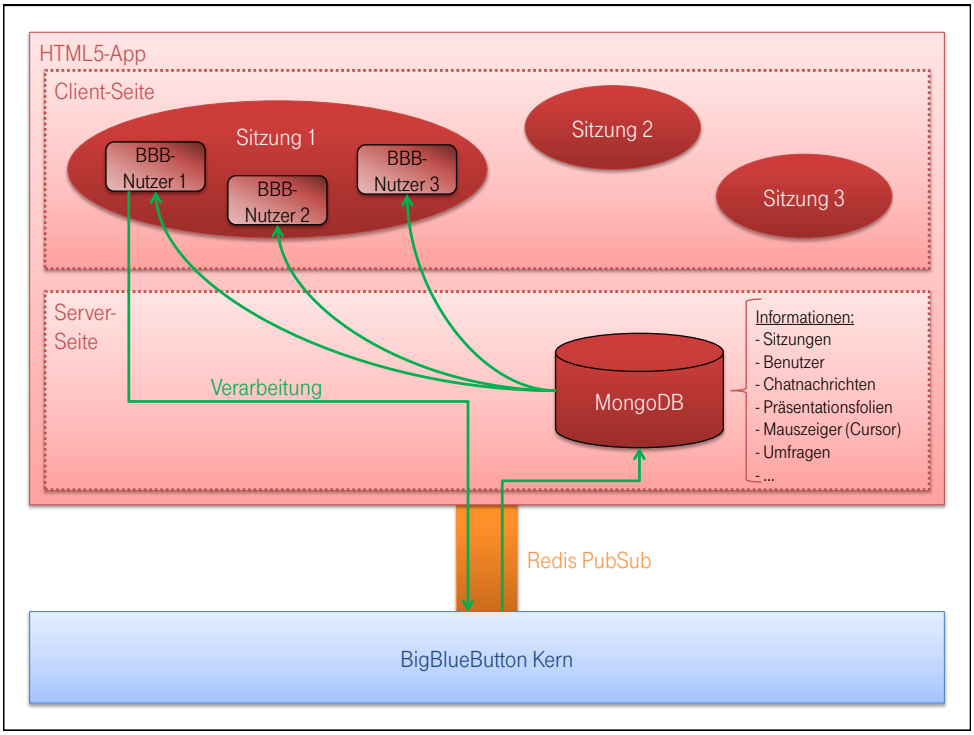

Abbildung 3: Funktionsweise mitgelieferter HTML5 Client, Interaktion des BBB-Nutzers 1, nach [4]

Möchte ein Nutzer selbst in den Sitzungsverlauf eingreifen (siehe Abbildung 3), schickt er die vom BBB-Kern benötigten Daten über die HTML5-Serverkomponente und anschließend über den Redis Channel zum BBB-Kern. Die Serverkomponente übernimmt dabei die Validierung der Daten und prüft, ob der Nutzer die benötigten Berechtigungen besitzt. Der BBB-Kern verarbeitet die gesendeten Daten und schickt anschließend eine Event-Nachricht über den Redis PubSub an MongoDB. MongoDB prüft nun, für wen die Event-Nachricht relevant ist und pusht entsprechende Nachrichten über eine dauerhaft geöffnete Transmission Control Protocol (TCP)-Websocket Verbindung an die einzelnen Clients. Somit wird sichergestellt, dass die Clients immer die aktuellen Sitzungsinhalte sehen können. (vgl. [4]) Von dem vorhandenen Aufbau ausgehend können zwei Lösungskonzepte abgeleitet werden, die in Unterabschnitt 4.6 und 4.7 beschrieben werden.

4.6 Lösungskonzept – Anpassung des HTML5-Clients

4.6.1 Beschreibung

Das erste Konzept basiert auf dem mitgelieferten HTML5 Client der BBB-Entwickler. Dieser bietet nicht nur die gewünschten Funktionen Textchat, Umfrageteilnahme und Emojis, sondern auch die Ansicht der Präsentationsfolien, Cursor, Audio über WebRTC und die komplette Teilnehmerliste. Nach Linder sollen Anwendungen einfach gehalten werden, damit die Nutzerfreundlichkeit möglichst groß ist (vgl. [18, S. 83-85]). Deshalb ist es notwendig, die Anwendung ausschließlich auf die gewünschten Funktionen zu reduzieren. Wie der Name es beschreibt, besteht der HTML-Client aus HTML-Seiten. Diese kann man verändern, indem man die „ungewollten" Funktionalitäten auskommentiert bzw. löscht. Anschließend muss das Design mittels Cascading Style Sheets (CSS) angepasst werden, damit die Usability nicht beeinträchtigt wird. In Abbildung 4 ist ein Message Sequence Chart (MSC) für den Zugang – gemäß

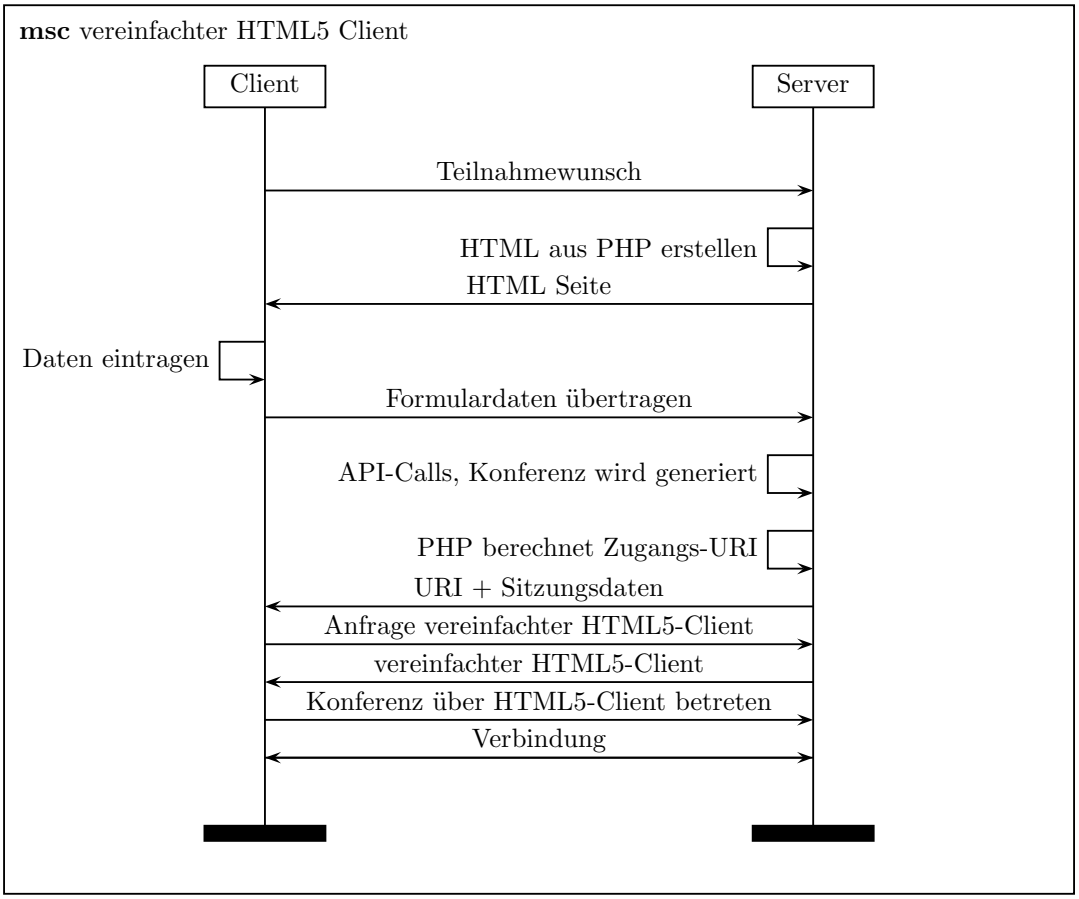

Abbildung 4: Message Sequence Chart: Zugang nach dem HTML5-Lösungskonzept

dem HTML5-Lösungskonzept – dargestellt. Der Client stellt eine Anfrage an den Server und drückt damit seinen Teilnahmewunsch aus. Der Server erstellt aus hinterlegten Hypertext Preprocessor (PHP)-Seiten eine für den Client passende HTML-Seite und schickt diese zusammen mit Stylesheets an den Client. Daraufhin gibt der Nutzer über die Formularfelder seine Daten,

wie z.B. Name, ein. Nach Übertragung der Formulardaten führt der Server einen API-Aufruf durch, um Sitzungsdaten zu generieren. Die Sitzungsdaten werden in einen Uniform Resource Identifier (URI) überführt und an den Client übertragen. Der Client kann nun mit Hilfe der URI den HTML5-Client beziehen und mittels der gelieferten Sitzungsdaten die im Hintergrund generierte Konferenz betreten. Die Verbindung ist etabliert.

Wie in Abbildung 5 ersichtlich, befinden sich alle Nutzer in derselben BBB-Sitzung und sind dieser logisch zugeordnet. Den Nutzern des vereinfachten Clients fehlen lediglich die für sie irrelevanten Informationen. Ansonsten unterscheiden sich die HTML5-Clients nicht. Alle Teilnehmer sind durch Pushbenachrichtigungen aus der MongoDB auf dem neuesten Stand.

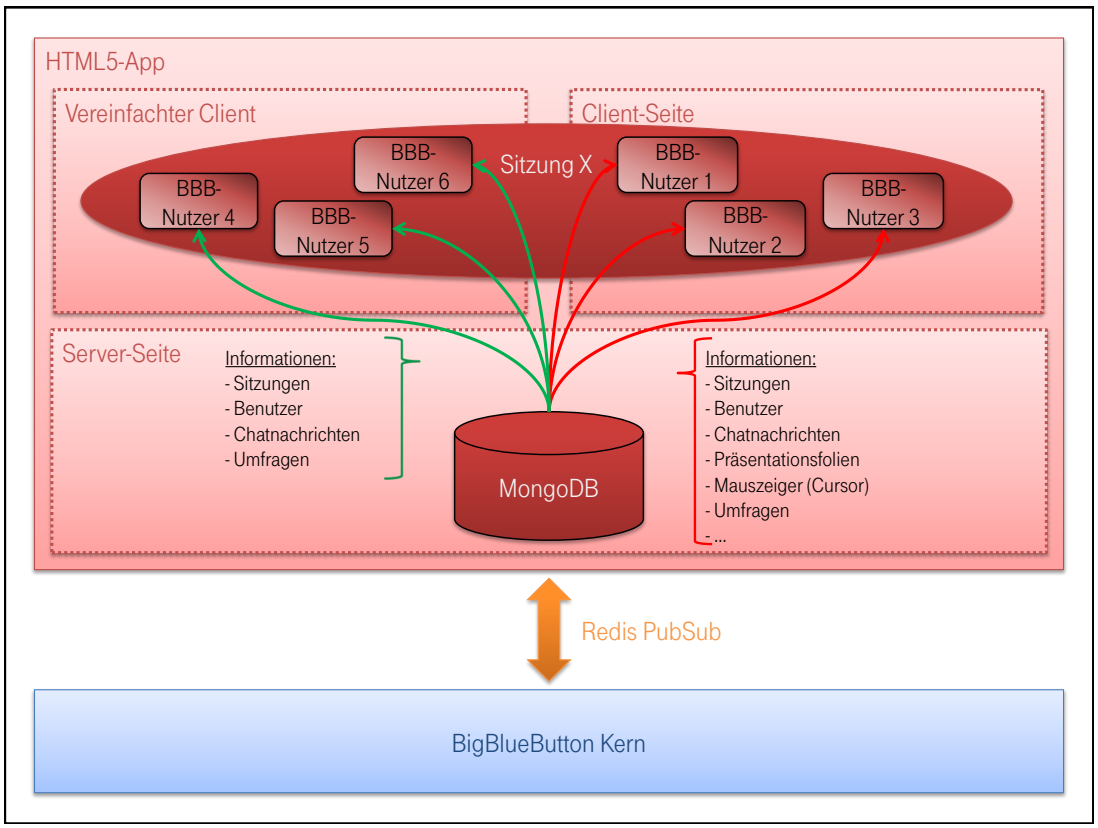

Abbildung 5: Funktionsweise angepasster HTML5 Client

4.6.2 theoretische Vorteile

Diese Variante ist besonders einfach, da nur die entsprechenden Zeilen entfernt werden müssen. Es muss kein neuer Quellcode entwickelt werden. Dabei bleiben alle gewollten Funktionen (vgl. Tabelle 2) erhalten.

Nach Rücksprache im BBB-Entwicklerforum erachtet der BBB-Entwickler Anton Georgiev diesen Entwicklungsansatz als besser, da die Umsetzung leichter ist und es zu keinen unvorhersehbaren Problemen im Backend kommen kann (vgl. [5], Anlagen Abschnitt 9.2).

4.6.3 mögliche Probleme

Innerhalb laufender Sitzungen ist es problematisch, mit diesem Konzept zu arbeiten, denn eine Umfrageteilnahme ist nur möglich, wenn der Nutzer vor dem Erstellen der Umfrage bereits in die Sitzung eingeloggt ist. Sonst werden die Antwortmöglichkeiten nicht angezeigt. Gerade weniger motivierte Studierende oder Teilnehmer, die zu spät erscheinen, könnten vergessen, rechtzeitig in die Sitzung einzutreten.

Wie in Abbildung 4 zu sehen, ist während der Veranstaltung für jeden Teilnehmer eine bidirektionale Verbindung geöffnet. Gerade in unsicheren bzw. instabilen Netzen kann das zum Problem werden. Wenn die Verbindung aufgrund von Netzüberlastung oder durch Standby-Funktionen des Endgerätes beendet wird, ist ein erneuter Login notwendig. Die Sitzung muss neu aufgebaut werden. Da das Szenario vorsieht, dass alle Teilnehmer sich in zwei oder mehr Hörsälen befinden, liegt es nahe, dass der Großteil das Campusnetz-Wireless Local Area Network (WLAN) nutzen möchte. Eine dauerhafte stabile Verbindung kann nicht garantiert werden.

Zusätzlich kann es passieren, dass die BBB-Entwickler den Original-HTML-Client bzw. das Backend bei Updates soweit verändern, dass der angepasste HTML-Client nicht mehr funktioniert. Somit müsste bei jedem Update die Funktionalität ausgiebig getestet und evtl. angepasst werden. Die Wartbarkeit wird dadurch immens erschwert.

4.7 Lösungskonzept – Nutzung des Redis PubSub Channels

4.7.1 Beschreibung

Wie in Abbildung 2 zu erkennen, läuft sämtliche Kommunikation beim mitgelieferten HTML-Client über den Redis Channel. Der Channel kann JSON-Strings verarbeiten und die Befehle an den BBB-Kern weiterleiten. Das zweite Lösungskonzept setzt an diesem zentralen Kommunikationskanal an.

In Abbildung 6 ist der Hauptbestandteil der zu entwickelnden Anwendung ein PHP-Script. Dieses stellt für alle Clients eine Kommunikationsschnittstelle dar. Nutzer kommunizieren nicht direkt mit dem BBB-Kern, sondern mit dem Script. Dieses leitet Anfragen an den Redis Channel im JSON-Format weiter. Der BBB-Kern kann nicht unterscheiden woher die Nachrichten stammen und leitet somit auch die manuell erzeugten Aktionen an die MongoDB zurück. In Folge muss das Script mit der MongoDB kommunizieren, um alle Informationen über Chatverlauf und Umfrageauswahl zu erhalten.

In Abbildung 7 ist das MSC für dieses Lösungsmodell am Beispiel des Absendens von Chatnachrichten dargestellt. Bis zu dem Punkt, an dem Sitzungsdaten mit Hilfe von API-Aufrufen erstellt werden, gibt es keinen Unterschied zum HTML5-Modell. Danach beginnt ein periodischer Vorgang. Der Nutzer sendet seine Chatnachricht an den Server. Der Server verarbeitet diese und erstellt einen JSON-String. Diesen übergibt er an den Redis Channel.

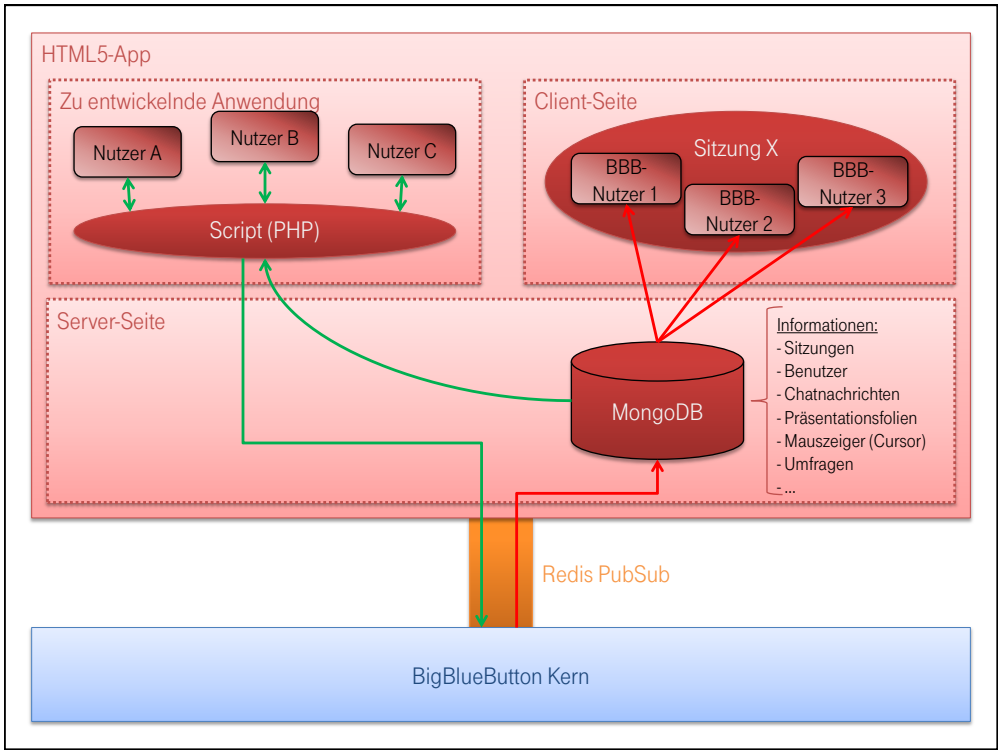

Abbildung 6: Funktionsweise Redis PubSub Channel

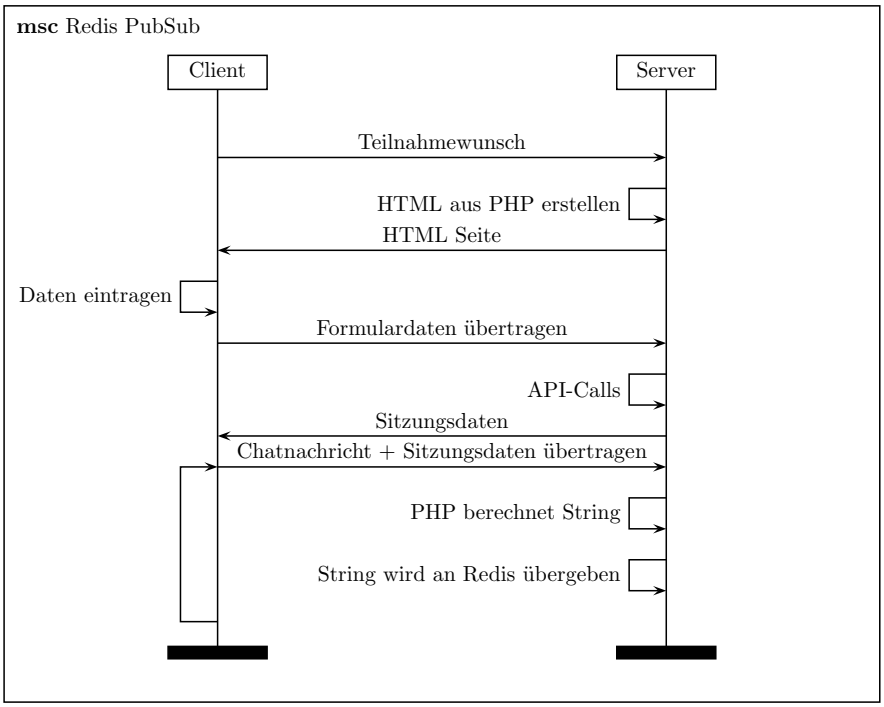

Abbildung 7: Message Sequence Chart: Zugang nach dem Redis PubSub-Lösungskonzept

4.7.2 theoretische Vorteile

Dieser Ansatz hat – im Gegensatz zum HTML5-Client Konzept – den Vorteil, dass keine dauerhafte Verbindung aufrecht gehalten werden muss. Gerade in instabilen Netzen ist das wichtig, damit es nicht zu Sitzungsabbrüchen auf Anwendungsebene kommt. Die Nutzer können jederzeit Aktionen durchführen und müssen nicht bereits am Veranstaltungsbeginn die Anwendung öffnen.

Da die Teilnehmer nicht logisch einer BBB-Sitzung zugeordnet sind, kann man auf initiale API-Aufrufe verzichten.

Nach eigenen Beobachtungen ändert sich das Nachrichtenformat des Redis PubSub-Channels selten, daher ist eine längere Beständigkeit dieses Lösungsansatzes vorstellbar.

4.7.3 mögliche Probleme

Da die Clients logisch keiner Sitzung zugeordnet sind, werden diese auch nicht in der Nutzerübersicht des Dozenten angezeigt. Das hat zur Folge, dass die Nutzer ihre Emotionen nicht über Emojis anzeigen können, da diese immer direkt neben dem Nutzernamen in der Übersicht erscheinen. Eine Kommunikationskomponente fällt somit weg (vgl. Tabelle 2).

Es ist nicht sichergestellt, dass dieses Lösungskonzept zum Erfolg führt, da es keine Erfahrungswerte gibt. Auch der BBB-Entwickler Anton Georgiev hat Zweifel, dass dieser Ansatz zielführend ist (vgl. [5], Anlagen Abschnitt 9.2).

Zusätzlich kann es zu unerwarteten Problemen im Backend kommen, die zum Anfang der Entwicklung noch nicht absehbar waren.

5 Initialer Lasttest

5.1 Testziele

Ohne eine stabile Audioverbindung kann keine qualitativ hochwertige Übertragung stattfinden. Frühere Untersuchungen ergaben, dass die Serverauslastung ausschlaggebend für die Qualität der Audioverbindung ist [13][25]. Deshalb soll der Test die Serverauslastung und benötigte Bandbreite in Abhängigkeit von der Nutzerzahl unter Verwendung des HTML5-Clients untersuchen. Dabei soll jederzeit eine stabile Audiokommunikation zwischen zwei Nutzern des Flash-Clients möglich sein. Diese stellen später den Dozentenrechner und den entfernten Übertragungsrechner dar. Um den späteren Verwendungsfall zu skizzieren, sollen die Testfälle „Nutzer nehmen an Umfrage teil" und „Nutzer schreiben Chatnachrichten" untersucht werden. Die Testergebnisse stellen den Ausgangszustand der Softwareentwicklung dar und werden im Anschluss mit den Endergebnissen verglichen.

5.2 Testumgebung

Der Testserver ist unter VMware vSphere[32] virtualisiert. Das VMWare Cluster besitzt drei Hosts mit je mindestens 24 logischen Prozessoren und mindestens 64GB Random-Access Memory (RAM). Die virtuelle Maschine läuft unter Ubuntu 14.04.1 mit 4GB RAM, was die Softwaremindestanforderungen erfüllt (vgl. [4]). Die dem BBB-Server zur Verfügung gestellten sechs virtuellen Prozessoren mit je einem Kern sind vom Typ Intel(R) Xeon(R) CPU E5-2640 und haben eine Taktung von $2,5GHz$.

Für den initialen Test wurde Nagios[33] 3.5.1 inklusive PNP4nagios[34] 0.6.16 installiert. Zusätzlich wurden zwei Scripte eingebunden, um die benötigte Bandbreite [21] und die derzeit verbunden Clients (siehe Anlagen) darzustellen. Getestet wurde das Verhalten des von den BBB-Entwicklern zur Verfügung gestellten HTML5 Clients. Um Fehler durch eine instabile WLAN-Verbindung zu vermeiden, wurde der Test im Teletutoring-Raum innerhalb des Campusnetzes durchgeführt. Dort standen vier Rechner zur Verfügung. Es wurde eine BBB-Demo-Sitzung gestartet, in die zwei Moderatoren eingetreten sind. Zusätzlich sollen möglichst viele Teilnehmer ohne Audioverbindung eintreten. Dafür wurden manuell an den Testrechnern Tabs in den Browsern Mozilla Firefox[35] und Google Chrome[36] geöffnet und die Seite zur Demo-Sitzung aufgerufen. Da der Aufwand für eine Automatisierung sehr groß ist, wurde darauf verzichtet (vgl. [13][25]). Im späteren Anwendungsfall soll kein Audiosignal zu jedem einzelnen Client übertragen werden. Deshalb wurden im Test nur die zwei Moderatoren in die WebRTC-Audiobrücke gebracht. In regelmäßigen Abständen sollen Umfragen erstellt werden, an denen möglichst alle Nutzer teilnehmen. Zusätzlich sollen Chatnachrichten geschrieben werden.

[32] Softwareprodukt von VMware zur Servervirtualisierung
[33] Monitoringprogramm
[34] Plugin für Nagios um grafische Anzeigen zu generieren
[35] `https://www.mozilla.org/de/firefox/`
[36] `https://www.google.de/chrome/browser/desktop/`

5.3 Ergebnisse

Wie Abbildung 8 zeigt, waren bis zu 83 Nutzer in der BBB-Sitzung eingeloggt. Dieser Wert wurde nach ca. 30 Minuten erreicht und konnte bis zur 70. Minute annähernd konstant gehalten werden. Ab diesem Zeitpunkt musste die Anzahl der aktiven Sitzungen verringert werden, da die Clientrechner überlastet waren und somit keine weiteren auswertbaren Tests durchgeführt werden konnten. Nach 90 Minuten wurde der Test beendet. Über den gesamten Testzeitraum wurden Chatnachrichten geschrieben und Umfrageantworten gesendet.

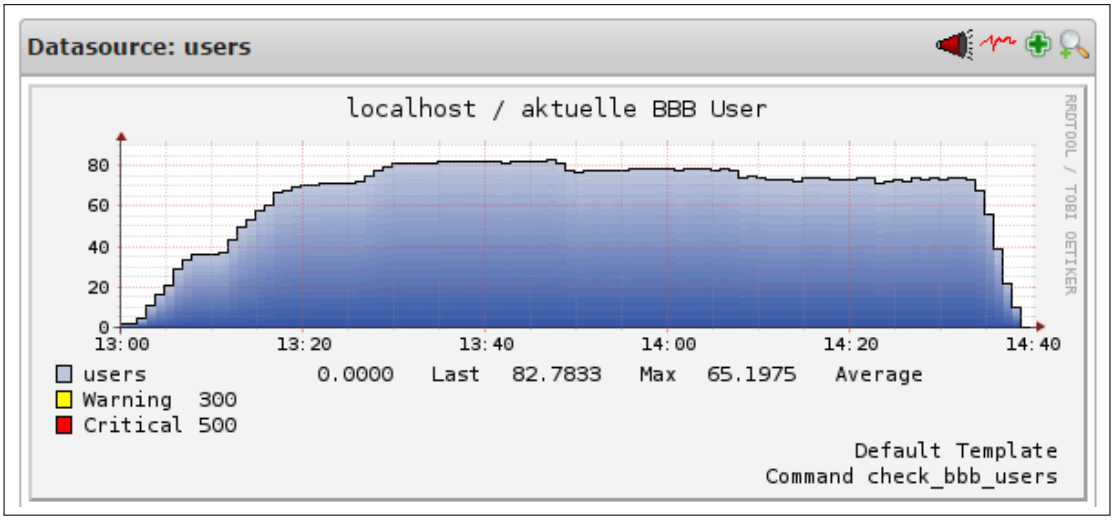

Abbildung 8: initialer Lasttest – Verlauf der eingeloggten BBB-Nutzer während des Lasttests

In Abbildung 9 wird deutlich, dass der eingehende Netzwerkverkehr (grün) gleichbleibend gering ist und somit nur geringfügig von der Nutzeranzahl abhängt. Beim ausgehenden Netzwerkverkehr (blau) ist ein Trend zu beobachten. Immer wenn ein Nutzer die Sitzung betritt, wird der HTML5 Client ausgeliefert und somit Daten übertragen. Am Anfang der Sitzung wurde das Maximum von $1775,7 \frac{kbit}{s}$ erreicht. Die Einbrüche bei 10 und 20 Minuten spiegeln sich auch in Abbildung 8 wider. Hier sind keine neuen Nutzer in die Sitzung gekommen. Ab Minute 45 ist ein wechselndes Verhalten zu beobachten, obwohl die Nutzerzahlen nicht stark schwanken. Das liegt daran, dass immer wieder vereinzelt Nutzer die Sitzung verlassen und neu betreten. Jedes Mal wird der Client ausgeliefert.

In Abbildung 10 ist die Serverlast dargestellt. Die rote Kurve zeigt den Lastdurchschnitt der letzten 15, die Orangefarbene der letzten 5 Minuten an. Die gelbe Kurve zeigt den Durchschnitt der Serverlast der letzten Minute an. Da die Nutzerzahlen nur schrittweise ansteigen (siehe Abbildung 8, Minute 0 bis 30), kann man verallgemeinern, dass die gelbe Kurve die derzeitige Serverlast darstellt. In den ersten 20 Minuten kann man erkennen, dass die Last (gelb) mit den Nutzerzahlen ansteigt. Diesen Wert kann man als Grundlast betrachten, der durch eingeloggte Nutzer (sowohl Flash-Client, als auch HTML5-Client) entsteht. Das Maximum liegt bei 0,51. Im weiteren Verlauf bleibt die gemittelte Last (rot) ausgeglichen, obwohl die derzeitige Last (gelb) stark variiert. Hier ist eine Auslastung zwischen 0,1 und 0,63 eingetreten. Die Lastspitzen sind durch das Senden von Chatnachrichten und Umfrageergebnissen zu begründen.

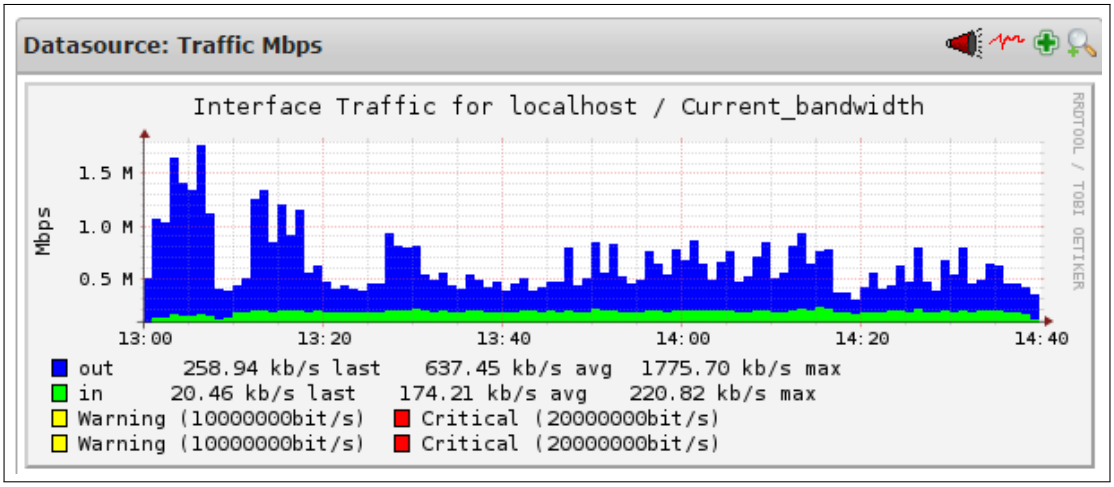

Abbildung 9: initialer Lasttest – Bedarf an Bandbreite während des Lasttests

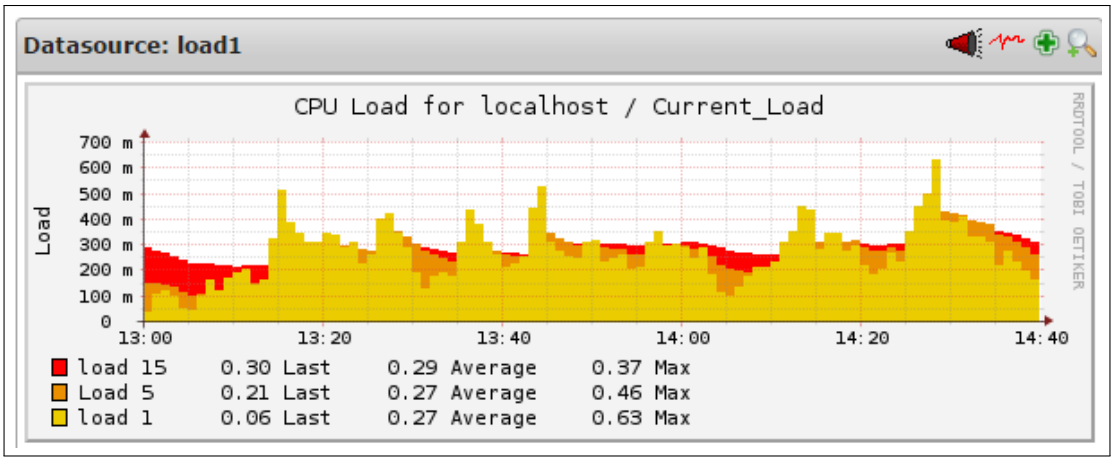

Abbildung 10: initialer Lasttest – Verlauf der Serverauslastung während des Lasttests

5.4 Bewertung

Die 83 simulierten Clients spiegeln noch nicht annähernd den realen Bedarf wider. Bei der dualen Wirtschaftsinformatik Gruppe, die 2016 immatrikuliert werden soll, ist mit knapp dreifach so vielen Teilnehmern zu rechnen. Der Netzwerkverkehr stellt dabei kein Problem am Server dar. Einkommende Daten (Abbildung 9, grüne Kurve) sind mit durchschnittlich $174,21\frac{kb}{s}$ verschwindend gering. Auch die Spitze an ausgehendem Netzwerkverkehr ist zu gering, um ein Problem darzustellen.

Lediglich die Prozessorauslastung könnte zum Problem werden. Je mehr Nutzer über den HTML5-Client in der Sitzung sind, umso größer wird die Grundlast des Servers. Die Lastspitzen, die durch Event-Nachrichten erzeugt werden (vgl. Abbildung 3), sind von der Anzahl der Nachrichten abhängig. Der Test bestätigt, dass es einen Zusammenhang zwischen teilnehmenden Nutzern und Servergrundlast gibt. Ebenfalls ist die Anzahl an Event-Nachrichten, die vom Server gepushten werden, proportional zur Nutzerzahl, da für jede Abstimmung und Chatnachricht eine ganze Event-Nachrichtensequenz angestoßen und verarbeitet werden muss.

5.5 Schlussfolgerung

Die verwendete virtuelle Maschine war für dieses Testszenario ausreichend. Da eine Abhängigkeit zwischen Nutzeranzahl und Serverauslastung gezeigt werden konnte, ist davon auszugehen, dass diese Maschine im realistischen Anwendungsfall bei deutlich mehr verbundenen Clients nicht mehr ausreicht. Jeder einzelne Client verursacht eine Grundlast, auch wenn dieser nicht in der Sitzung interagieren möchte sondern nur zuhört.

Da sich beim ersten Lösungskonzept (Anpassung des HTML5-Clients, Unterabschnitt 4.6) nur die Ansicht für den Nutzer ändert und das Backend des Clients gleich bleibt, ist anzunehmen, dass die Grundlast des Servers erhalten bleibt.

Im Gegensatz dazu ist es denkbar, dass die Grundlast beim zweiten Lösungskonzept verringert wird, da keine permanente Verbindung besteht. Die Lastspitzen bei Event-Nachrichten müssten erhalten bleiben.

Es ist nicht möglich, dass Nutzer erst bei Bedarf in die Sitzung eintreten. Denn bei Umfragen erscheinen die Antwortmöglichkeiten nur, wenn der Nutzer vor Erstellung einer Umfrage der Sitzung beigetreten ist. Da diese Bedingung des HTML5-Clients die Usability sehr einschränkt, ist das ein Ausschlusskriterium des ersten Lösungskonzeptes.

Auf Grundlage der Lasttestergebnisse und der Darstellungen in den Unterabschnitten 4.6 und 4.7 fällt die Entscheidung zur Entwicklung schlussendlich auf den Redis-PubSub Lösungsansatz, auch wenn das einen Wegfall der Feedbackemojis bedeutet. Diese wurden nach Rücksprache mit dem Auftraggeber als optional und damit verzichtbar betrachtet. Das Risiko, dass das Entwicklungskonzept aufgrund unvorhersehbarer Fehler nicht funktioniert, wird dabei in Kauf genommen.

6 Softwareentwicklung

6.1 verwendete Technologien

Um den Einstiegsaufwand für alle Teilnehmer möglichst gering zu halten, soll die Anwendung nicht in eine App eingebettet werden sondern auf den Browsern von Smartphones laufen. Damit entfallen eine Installation auf Endnutzerseite und die Anpassung der Software für verschiedene Appstores[37]. Android und iOS sind die derzeit am weit verbreitetsten Betriebssysteme für Smartphones und decken zu 98,9% den Markt ab [9]. Die Anwendung soll in Browsern auf beiden Plattformen laufen. Eine Nutzung mit Windows Phone, BlackBerry oder anderen Smartphones wird nicht unterstützt und getestet, da diese einen geringen Verbreitungsgrad haben.

Die Webseite, die an die Browser ausgeliefert wird, ist in HTML geschrieben. Um die serverseitige Funktionalität zur Verfügung zu stellen, gibt es verschiedene Sprachen zur Auswahl, wie z.B. Java, Perl, PHP oder Python. Für die Entwicklung wird auf PHP zurückgegriffen, da diese Programmiersprache für Webseiten am häufigsten genutzt wird [22], der Quelltext direkt im HTML-Dokument eingebettet werden kann [11, S. 2], es fertige Funktionen für die Anwendungsfälle in den Unterabschnitten 6.4 und 6.5 gibt und die Performance sehr gut ist [11, S. 4].

Da bei der Standardinstallation von BBB noch kein PHP mitgeliefert wird, müssen über Advanced Packaging Tool (APT)[38] die Pakete *php5* und *php5-fpm*[39] installiert werden.

Es bietet sich an, die Anwendung auf dem Nginx-Webserver laufen zu lassen, der schon durch die BBB-Installation mitgeliefert wurde. Damit PHP ordnungsgemäß eingebunden wird, muss die Konfigurationsdatei des Webservers angepasst werden (siehe Anlagen). Die Pakete *php-pear*, *php5-dev*, *libpcre3-dev*, *php5-mongo* und *php-redis* müssen für die in Unterabschnitt 6.6 beschriebenen Funktionen über APT installiert werden. Eine Einbindung der MongoDB Treiber über *„pecl install mongodb"* ist ebenfalls notwendig. Im Anschluss müssen noch die Extensions[40] für MongoDB und Redis in */etc/php5/fpm/php.ini* eingefügt werden (siehe Anlagen). Abschließend ist ein Neustart der Dienste *php5-fpm*, *nginx* und *bbb-html5* erforderlich.

Die entwickelte Anwendung setzt sich aus sieben Dateien zusammen. Für jede der fünf verschiedenen Menüseiten gibt es eine PHP-Seite. Zusätzlich befindet sich eine *parameters.php* im gleichen Verzeichnis. Hier sind alle Funktionen und dynamische Informationen gespeichert, die individuell bei Anwendungsumzug anzupassen sind.

Damit eine schnelle Navigation zur Startseite (*index.php*) möglich ist, befinden sich die PHP-Dateien im Wurzelverzeichnis des Nginx-Webservers (siehe Listing 7, Zeile 2). Im Unterverzeichnis *./css* liegt das Stylesheet *landingpage.css* zur formatierten Darstellung der HTML-Inhalte.

[37] Vertriebsplattform für Smartphone-Software
[38] Paketverwaltungssystem
[39] FastCGI (Common Gateway Interface) Process Manager
[40] Erweiterungen

6.2 Startseite

Die Einstiegsseite ist die *index.php*. Diese wird durch die in Unterabschnitt 6.1 vorgenommenen Einstellungen geladen, sobald man den Nginx-Webserver ansteuert. In Abbildung 11 ist der Aufbau visualisiert.

Abbildung 11: entwickelte Software – Einstiegsseite

Die Benutzer tragen ihren Namen ein und geben über eine Dropdown-Auswahlliste an, in welchem Raum sie sich physisch befinden. Die Auswahlliste wird aus der Parameterdatei gespeist. Danach wird eine Veranstaltung über Radiobuttons gewählt und das Raumpasswort eingegeben. Das Raumpasswort ist die eigentliche externe Meeting-Identifikationsnummer (ID) der Sitzung und befindet sich in der Kopfzeile des Flash-Clients. Nur Nutzern, die an der Vorlesungsübertragung teilnehmen, ist es möglich, das Raumpasswort zu sehen. Der alleinige Besitz der Meeting-ID stellt kein Problem dar, da nur mithilfe des Secret Salts in die Sitzung eingegriffen werden kann. Das Secret Salt ist ein Shared Secret und nur für Administratoren bestimmt.

Um die Meeting-ID darzustellen, muss der Flash-Client umprogrammiert und neu kompiliert werden (vgl. [4] [12]).
Anpassungen müssen in *[Entwicklungsverzeichnis]/bigbluebutton-client/src/org/bigbluebutton/core/managers/UserConfigManager.as* (Listing 1) und *[Entwicklungsverzeichnis]/bigbluebutton-client/src/org/bigbluebutton/main/views/MainToolbar.mxml* (Listing 2) vorgenommen werden.

```
44 public function getMeetingID():String {
45  if (conferenceParameters)
46  return conferenceParameters.externMeetingID;
47  else
48  return null;
49 }
```

Listing 1: UserConfigManager.as

Die Änderungen in Listing 1 erstellen eine Funktion, die die externe Meeting-ID zurück liefert. In Listing 2 wird die Beschriftung der Client-Kopfzeile während und nach dem Ladevorgang des Flash-Clients angepasst.

```
148 private function retrieveMeetingName(e:ConferenceCreatedEvent):
    void {
149 if (toolbarOptions.showMeetingName) {
150 var meetingTitle:String = BBB.initUserConfigManager().
    getMeetingTitle();
151 var meetingID:String = BBB.initUserConfigManager().getMeetingID();
152 if (meetingTitle != null) {
153 meetingNameLbl.text = meetingTitle + ", Sitzungspasswort: " +
    meetingID;
154 meetingNameLbl.toolTip = meetingID;
155 }
156 }
157 logFlashPlayerCapabilities();
158 }
159 ...
366 ...
367 if (toolbarOptions.showMeetingName) {
368 var meetingTitle:String = BBB.initUserConfigManager().
    getMeetingTitle();
369 var meetingID:String = BBB.initUserConfigManager().getMeetingID();
370 if (meetingTitle != null) {
371 meetingNameLbl.text = meetingTitle + ", Sitzungspasswort: " +
    meetingID;
372 }
373 }
```

Listing 2: MainToolbar.mxml

Alle eingetragenen Daten (Abbildung 11) werden über die Hypertext Transfer Protocol (HTTP)-Methode *POST* an die nächste Seite (*start.php*) übergeben.

6.3 Hauptmenü

Auf der Hauptseite der Anwendung findet zuerst die Validierung des Raumpasswortes statt. Dafür werden über den API-Aufruf *getMeetings* alle laufenden Meetings des Servers abgefragt (vgl. Unterabschnitt 6.6) und die Meeting-Namen aus der Antwort und der POST-Variable gemappt. Die zum Meeting-Namen gehörende Meeting-ID wird mit dem übergebenen Raumpasswort verglichen. Falls das angegebene Passwort nicht mit der Meeting-ID übereinstimmt, findet ein Redirect[41] auf die *index.php* statt, wo eine passende Fehlermeldung ausgegeben wird. Eine Übersicht aller Variablen ist in Tabelle 3 dargestellt.

Tabelle 3: Übersicht aller Variablen im Hauptmenü

Variable	**Bedeutung**	**Quelle**
Name	Name des Nutzers	POST (vom Nutzer eingegeben)
Raum	Hörsaal	POST (vom Nutzer ausgewählt)
Meeting-Name	Veranstaltung, zur Meeting-ID gehörend	POST (vom Nutzer ausgewählt)
Raumpasswort	identisch mit Meeting-ID	POST (vom Nutzer eingegeben)
Meeting-ID	identifiziert Meeting-Name	API-Aufruf (getMeetings): Meeting-Name
Sitzungspasswort	entscheidet, ob Nutzer Moderationsrechte bekommt oder nicht	API-Aufruf (getMeetings): Meeting-Name & Raumpasswort
Nutzer-ID	gültige BBB-ID	API-Aufruf (join): Name, Meeting-ID & Sitzungspasswort

Damit die Nutzer eindeutig über eine valide BBB-Nutzer-ID identifiziert werden können, muss diese über den API-Aufruf *join* unter Angabe des Namens, der Meeting-ID und des vorher bestimmten Sitzungspasswortes generiert werden. Zu beachten ist, dass *redirect=false* gesetzt wird, um die individuellen Sitzungsinformationen zu erhalten und keine Weiterleitung zum Flash-Client stattfindet.

Bis auf das Sitzungspasswort werden alle Werte aus Tabelle 3 in Browser-Cookies gespeichert, um eine erneute Anmeldung zu umgehen.

Wie in Abbildung 12 dargestellt, werden im Hauptmenü drei große Buttons angezeigt, um in die einzelnen Untermenüs zu navigieren. Dabei stehen die Funktionen „an Umfrage teilnehmen", „Chatnachricht schreiben" und „Chat lesen" zu Verfügung.

[41] Umleitung

Abbildung 12: entwickelte Software – Hauptmenü

6.4 Umfragemodul

Im Untermenü *Umfrage* werden bei laufender Umfrage alle Antwortmöglichkeiten angezeigt (siehe Abbildung 13).

Die Nutzer können pro Umfrage einmalig teilnehmen. Dabei wird die Umfragen-ID in einem Browser-Cookie gespeichert, um eine erneute Teilnahme zu erschweren.

Die Antwortmöglichkeiten werden aus der MongoDB Collection *bbb_poll* ausgelesen (vgl. Unterabschnitt 6.6).

Nachdem die Nutzer nun die für sie passende Antwortmöglichkeit ausgewählt und angeklickt haben, muss die Information noch an den BBB-Server übermittelt werden. Dafür wird eine Redis Nachricht erzeugt (vgl. Unterabschnitt 6.6).

Die Nutzer können jederzeit über den *Zurück*-Button in das Startmenü gelangen oder die Seite neu laden, wenn es bei Seitenaufruf keine aktive Umfrage gab.

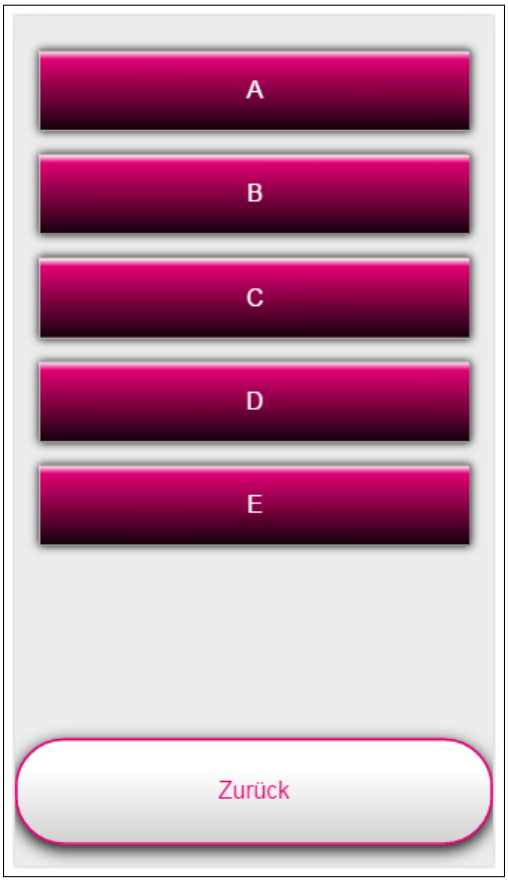

Abbildung 13: entwickelte Software – Antwortmöglichkeiten bei aktiver Umfrage

6.5 Chatmodule

6.5.1 Chatnachricht schreiben

In dem Modul werden am oberen Rand die letzten drei Chatnachrichten angezeigt (siehe Abbildung 14). Diese müssen aus der MongoDB ausgelesen werden (vgl. Unterabschnitt 6.6).

Wenn man in das Textfeld eine Nachricht eingibt und auf *Senden* klickt, wird die Chatnachricht in einer POST-Variable gespeichert und die Seite neu geladen. Wenn die POST-Variable beim Laden der Seite gesetzt ist, wird die Nachricht über den Redis-Channel an den Server übertragen (vgl. Unterabschnitt 6.6).

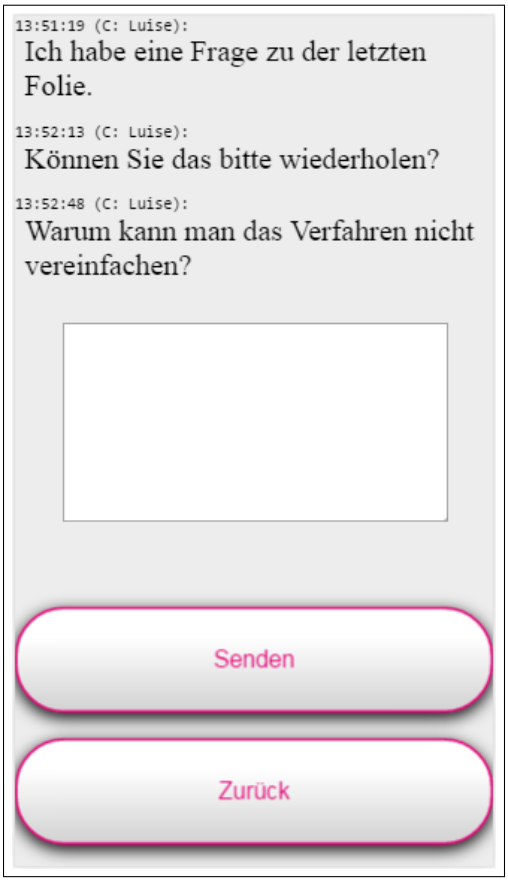

Abbildung 14: entwickelte Software – Chatnachricht schreiben

6.5.2 alle Chatnachrichten lesen

In diesem Untermenü werden alle bereits versendeten Nachrichten aus der zugeordneten Sitzung chronologisch dargestellt (siehe Abbildung 15).

Dafür müssen die MongoDB Collection *bbb_chat* abgefragt und das zurückgegebene Array in eine lesbare Form gebracht werden.

Zu jeder Chatnachricht wird der Zeitstempel und der Verfasser inkl. Hörsaalbezeichnung ausgegeben.

Über Javascript (siehe Listing 3) scrollt die Ansicht der Chatnachrichten an das untere Ende und zeigt somit die aktuellsten Nachrichten an.

```
77 <script>
78  var scrbtm = document.getElementById("chat_scroll");
79  scrbtm.scrollTop = scrbtm.scrollHeight;
80 </script>
```

Listing 3: JavaScript Funktion, scrollt ans Ende des Divs

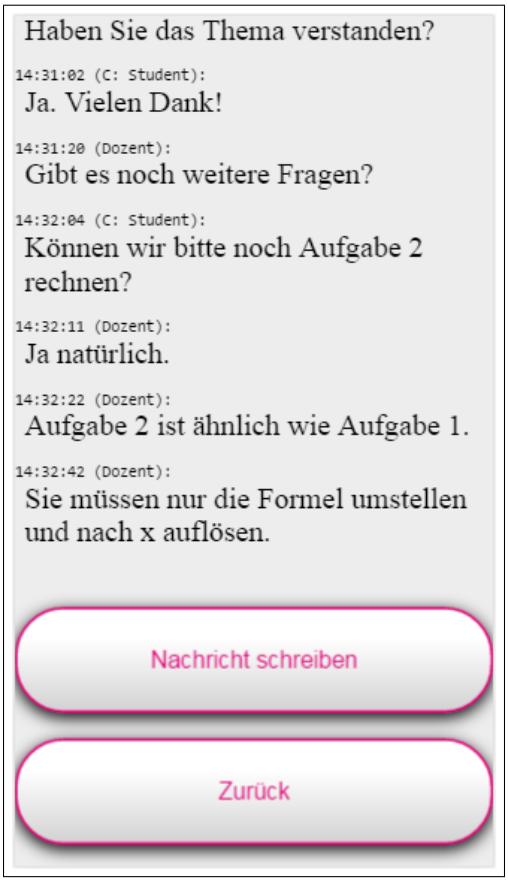

Abbildung 15: entwickelte Software – gesamten Chatverlauf lesen

6.6 Beschreibung der Kommunikationsschnittstellen

6.6.1 BBB API

Das BBB-API ist eine Schnittstelle, welche die Durchführung verschiedener Aktionen am Server ermöglicht. Über diese Aktionen ist es nicht nur möglich Informationen abzurufen sondern auch den Server zu manipulieren.

Die Schnittstelle wird über einen Uniform Resource Locator (URL) angesprochen und gibt eine Extensible Markup Language (XML)-Nachricht zurück (vgl. [4]).

http://[BBB-Server]/bigbluebutton/api/[API-Call]?[Parameter]&checksum=[Checksumme] (1)

Die Form der URL ist in (1) dargestellt (nach [4]). Der API-Call stellt dabei die auszuführende Funktion dar. Bei der entwickelten Anwendung werden die API-Aufrufe *getMeetings* und *join* genutzt. Bei *join* werden noch zusätzliche Parameter – der Name des Benutzers, die Meeting-ID und das Sitzungspasswort – benötigt. Diese werden an die URL angehängt. Zum Schluss muss eine Checksumme mit übergeben werden. Diese wird mit API-Aufruf, Parameter und Secret Salt[42] über den Secure Hash Algorithm 1 (SHA1) gebildet (siehe (2)). Dabei muss das Frage-

[42] Shared Secret von BBB

zeichen zwischen API-Aufruf und Parameter und das letzte &-Zeichen vor der Checksumme weggelassen werden.

$$\text{Checksumme} = SHA1(\text{[API-Call][Parameter][Secret Salt]}) \tag{2}$$

Die API-Aufrufe in der entwickelten Anwendung werden über eine Funktion geregelt, die in den Anlagen abgebildet ist.

6.6.2 Redis PubSub-Channel

Der einzige Kommunikationskanal zwischen der Anwendung und dem BBB-Kern ist der Redis PubSub-Channel. Durch den Befehl *redis-cli monitor* werden alle Channel-Nachrichten auf der Konsole ausgegeben. Abbildung 16 zeigt die Syntax einer Redis Channel-Nachricht des Typs „Chatnachricht schreiben". Die dazugehörige Tabelle gibt eine Übersicht, woher die Informationen bezogen werden können. Alle Elemente, die nicht markiert sind, werden bei neuen Nachrichten so beibehalten. Wenn man die Nachricht nun kopiert, nach eigenen Vorstellungen anpasst (z.B. Text der Nachricht, Name und Zeitstempel) und den String dem Konsolenprogramm *redis-cli* übergibt, erscheint eine neue Nachricht mit gewünschten Inhalten im Textchat. Auch eine Datenkontrolle in der MongoDB zeigt einen richtigen – von „originalen" Chatnachrichten nicht zu unterscheidenden – Eintrag.

Da mit dem Schreiben von Chatnachrichten so gute Erfolge erzielt wurden, sollte darauf aufbauend das Verhalten des Umfrage-Channels untersucht werden. Die Syntax mit Beschreibung und möglicher Informationsquelle für das Beispiel „Nutzer nimmt an Umfrage teil" ist in Abbildung 17 dargestellt. Das größte Problem stellt hier die Nutzer-ID dar. Serverseitig wird geprüft, ob der Umfrageteilnehmer logisch der Sitzung zugeordnet ist. Nur dann darf an der Umfrage teilgenommen werden. Dadurch kann die im Browser-Cookie gespeicherte Nutzer-ID nicht verwendet werden. Eine mehrfache Verwendung einer „erlaubten" Nutzer-ID ist hingegen unproblematisch. Deshalb wird bei jeder Umfrageteilnahme die Nutzer-ID des Erstellers der Umfrage verwendet.

In Listing 4 ist dargestellt, wie Redis-Nachrichten innerhalb der Anwendung versendet werden. Die Aktion ist im entwickelten Szenario immer *„publish"*. Die Channel sind je nach Verwendung *„bigbluebutton:to-bbb-apps:chat"* oder *„bigbluebutton:to-bbb-apps:polling"*.
Die Strings werden wie in Abbildung 16 bzw. Abbildung 17 zusammengesetzt.

```
80 function RedisPubSub ($action, $channel, $string) {
81 $redis = new Redis();
82 $redis->connect($GLOBALS["redis_host"], $GLOBALS["redis_port"]);
83 $redis->$action($channel, $string);
84 $redis->close();
85 }
```

Listing 4: Funktion – Kommunikation mit Redis Channel

Abbildung 16: Redis Channel-Nachricht – Chatnachricht


```
"PUBLISH" "bigbluebutton:to-bbb-apps:chat" "{\"payload\":{\"message
   \":{\"message\":\"Hallo Welt!\",\"fromUserID\":\"iqacdgvolmx4\",\"
   chatType\":\"PUBLIC_CHAT\",\"fromUsername\":\"Luise Kaufmann\",\"
   fromTimezoneOffset\":\"-120\",\"toUsername\":\"
   public_chat_username\",\"toUserID\":\"public_chat_userid\",\"
   fromTime\":\"1.473424019049E12\",\"fromColor\":\"0xE20074\"},\"
   meeting_id\":\"
   183f0bf3a0982a127bdb8161e0c44eb696b3e75c-1473423875574\"},\"header
   \":{\"timestamp\":3351294334,\"name\":\"send_public_chat_message
   \",\"version\":\"0.0.1\"}}"
```


Element	Beschreibung	Quelle
Redis PubSub Kommando	Channel-Nachricht wird veröffentlicht/„geschrieben”	fest gesetzt
Redis Channel	„Chat”-Channel	fest gesetzt
Chatnachricht		durch Nutzereingabe
Nutzer-ID des Absenders		aus Browser-Cookie
Art der Chatnachricht	Public oder Private	fest gesetzt
Name des Absenders		aus Browser-Cookie
Zeitzone	Verschiebung in Minuten zur Koordinierte Weltzeit (UTC)	Serverzeit oder fest gesetzt
Name des Empfängers		da immer Public –> fest gesetzt
Nutzer-ID des Empfängers		da immer Public –> fest gesetzt
Unix Zeitstempel in Millisekunden	in Exponentenschreibweise	PHP Funktion time() bzw. microtime()
Farbe der Chatnachricht		frei wählbar
Interne Meeting-ID		API-Aufruf (getMeetingInfo): externe Meeting-ID (aus Cookie), Moderatorpasswort (API-Aufruf: getMeetings)
weiterer Zeitstempel	unbekannter Ursprung, kein Einfluss feststellbar	fest gesetzt

Abbildung 17: Redis Channel-Nachricht – Umfragenteilnahme


```
"PUBLISH" "bigbluebutton:to-bbb-apps:polling" "{\"payload\":{\"
   answer_id\":0,\"user_id\":\"qmdd87oalrpx_2\",\"meeting_id\":\"
   183f0bf3a0982a127bdb8161e0c44eb696b3e75c-1473433263476\",\"question_id
   \":0,\"poll_id\":\"
   d2d9a672040fbde2a47a10bf6c37b6a4b5ae187f-1473433263477/1/1473433514413
   \"},\"header\":{\"timestamp\":7600510401,\"name\":\"
   vote_poll_user_request_message\",\"version\":\"0.0.1\"}}"
```


Element	Beschreibung	Quelle
Redis PubSub Kommando	Channel-Nachricht wird veröffentlicht/„geschrieben"	fest gesetzt
Redis Channel	„Umfrage"-Channel	fest gesetzt
Antwort-ID		MongoDB Abfrage –> Unterunterabschnitt 6.6.3
Nutzer-ID	muss für Validierung mit *„_2"* verknüpft werden; Nutzer muss logisch der Sitzung zugeordnet sein	API-Aufruf (getMeetingInfo): Meeting-ID (aus Cookie), Moderatorpasswort (API-Aufruf: getMeetings)
Interne Meeting-ID		API-Aufruf (getMeetingInfo): externe Meeting-ID (aus Cookie), Moderatorpasswort (API-Aufruf: getMeetings)
Umfragen-ID		MongoDB Abfrage –> Unterunterabschnitt 6.6.3
weiterer Zeitstempel	unbekannter Ursprung, kein Einfluss feststellbar	fest gesetzt

6.6.3 MongoDB

MongoDB ist die zentrale Datensammlung aller Sitzungsinformationen (vgl. Abbildung 2). Über die Mongo Shell kann konsolenbasiert auf die Datenbank zugegriffen werden [20]. Es gibt drei Datenbanken: *admin*, *local* und *meteor*. Jede Datenbank besteht aus verschiedenen Collections. In der *Meteor*-Datenbank liegen alle entscheidenden BBB-Collections, z.B. *bbb_chat*, *bbb_cursor*, *bbb_poll* und *bbb_user*. Eine beispielhafte Datenbankabfrage über Mongo Shell ist in den Anlagen zu sehen.

In Abbildung 18 ist ein Eintrag aus der Collection *bbb_poll* dargestellt. Alle markierten Daten werden benötigt, um über den Redis Channel an einer Umfrage teilzunehmen (vgl. Unterunterabschnitt 6.6.2). Für jede Antwortmöglichkeit wird für die Nutzerabstimmung ein Button in der *umfrage.php* erstellt (siehe Abbildung 13).

Abbildung 18: MongoDB Eintrag – Umfrage

```
{ "_id" : "k5HW2ncsTtjEBQDty", "poll_info" : { "meetingId" : "
   183f0bf3a0982a127bdb8161e0c44eb696b3e75c-1473527226874", "poll" : { "
   id" : "
   d2d9a672040fbde2a47a10bf6c37b6a4b5ae187f-1473527226875/1/1473528147700",
    "answers" : [ { "id" : 0, "key" : "True", "num_votes" : 0 }, {
   "id" : 1, "key" : "False", "num_votes" : 0 } ], "num_responders"
    : -1, "num_respondents" : -1 }, "requester" : "jqqzk7cj5pny_2
   ", "users" : [ "jqqzk7cj5pny_2" ] } }
```

Legende: interne Meeting-ID Umfragen-ID Antwort-ID Antwort

Um die MongoDB-Daten über die PHP-Seiten abrufen zu können, muss die in Listing 5 dargestellte Funktion – unter Angabe der gewünschten Collection – genutzt werden. Aus dem zurückgelieferten Array können im Anschluss alle Informationen gelesen werden.

```
87 function Mongo ($collection) {
88 $m = new Mongo("mongodb://".$GLOBALS["mongo_host"].":".$GLOBALS["
      mongo_port"]);
89 $batch = $m->$GLOBALS["mongoDB_name"]->$collection->find();
90 $m->close();
91 $array = iterator_to_array($batch);
92 return $array;
93 }
```

Listing 5: Funktion – Verbindung mit MongoDB

6.7 Fehlerbehandlung und Sicherheit

Um die Nutzerfreundlichkeit zu erhöhen ist es angemessen, bei groben Eingabefehlern, die ein weiteres Bedienen beeinträchtigen, den Nutzern eine Fehlerbeschreibung zu übergeben. In der entwickelten Anwendung gibt es auf der Startseite die Möglichkeit, einen Fehlertext zu übergeben, der in einem farblich hervorgehobenen Feld angezeigt wird. Alle Eingabefelder auf der Startseite sind als Pflichtfelder markiert. Da nicht alle Browser die HTML-Funktion *„Required"* verstehen, wird zusätzlich geprüft ob die *POST*-Variablen gesetzt sind. Wenn das nicht der Fall ist, findet ein Redirect auf die Startseite statt und eine Fehlermeldung wird ausgegeben. In dieser ist ersichtlich, welches Feld nicht ordnungsgemäß ausgefüllt wurde.

Nach dem gleichen Prinzip findet eine Validierung des Raumpasswortes statt.

Damit bei der Eingabe von Chatnachrichten keine Sicherheitslücken ausgenutzt werden (vgl. [23, S. 4]), gibt es eine Ersetzungsbibliothek, die alle Sonderzeichen escaped bzw. ersetzt. Gerade Datenbankänderungen und das Ausführen von JavaScript müssen verhindert werden, damit Angreifer keinen Zugriff (und somit Kontrolle) auf das System bekommen. Um in Erfahrung zu bringen, gegen welche Strings die Zeichen ersetzt werden müssen, war ein Abhören des Redis Channels notwendig (vgl. Unterunterabschnitt 6.6.2).

Während der Entwicklung ist aufgefallen, dass der originale Flash-Client der BBB-Entwickler Schwachstellen aufweist. So ist es beispielsweise nicht möglich, einen Backslash (\) im Chat zu schreiben. Erst bei Eingabe eines doppelten Backslashes wird ein einfacher Backslash in den Chat geschrieben.

7 Abschließender Lasttest

7.1 Testziele

Im abschließendem Systemtest soll die entwickelte Software mit dem Ausgangszustand (vgl. Abschnitt 5) in Hinsicht auf Nutzeranzahl, benötigte Bandbreite und Serverauslastung verglichen werden. Es soll insbesondere untersucht werden, ob die vermuteten Ergebnisse aus Unterabschnitt 5.5 erreicht werden.

7.2 Testumgebung

Damit die Testergebnisse vergleichbar sind, wurde auch der abschließende Lasttest mit gleicher Server-Hardware im Teletutoring-Raum der HfTL durchgeführt. Zwei Nutzer sind über den Flash-Client in die Sitzung eingetreten. Diese simulieren die Dozenten- und Übertragungsrechner (siehe Abbildung 19).

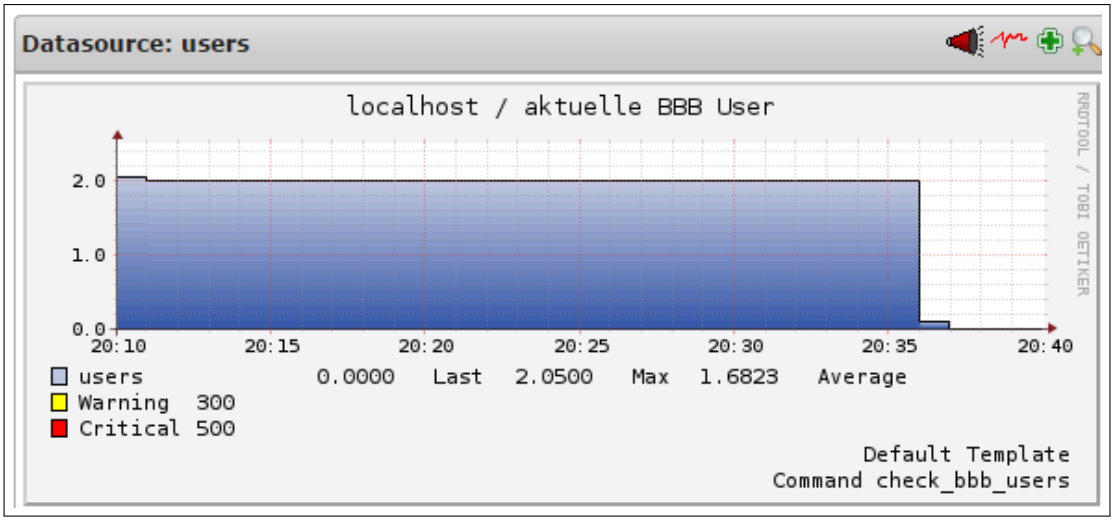

Abbildung 19: abschließender Lasttest – Verlauf der eingeloggten BBB-Nutzer während des Lasttests

Da es bei der entwickelten Anwendung in Hinsicht auf die Serverlast unerheblich ist, ob 60 Nutzer in einer Minute jeweils eine Nachricht versenden, oder ein Nutzer in einer Minute 60 Nachrichten verschickt, mussten nicht 80 Clients die Anwendung öffnen. Diese Verallgemeinerung kann getroffen werden, da die Teilnehmer durch die entwickelte Anwendung nicht mehr logisch in die Sitzung eintreten. Es gibt keine etablierte Verbindung, die über den gesamten Sitzungsverlauf aufrechterhalten werden muss. Jegliche Kommunikation wird nur über API-Aufrufe, Datenbankabfragen und Redis Channel Nachrichten realisiert (vgl. Unterabschnitt 6.6). Die ausgetauschten Informationen unterscheiden sich lediglich in ihren Parametern (z.B. Nutzer-ID), wenn sie von mehreren Clients gesendet werden. Für die Datenverarbeitung sind die Parameterwerte uninteressant. Alle Nachrichten des gleichen Typs werden unabhängig vom Inhalt gleich behandelt.

Aufgrund dieser Verallgemeinerung war es ausreichend, bei zwei Clients die Anwendung zu öffnen. Die Clients erscheinen nicht in Abbildung 19, da in Nagios über API-Aufrufe die Clientanzahl ermittelt wird. Die Clients der entwickelten Anwendung sind nicht logisch der Sitzung zugeordnet und können somit auch nicht über API-Aufrufe abgefragt werden.

7.3 Ergebnisse

Der Test dauerte 30 Minuten. In Abbildung 20 ist der ein- und ausgehende Netzwerkverkehr grafisch dargestellt. Im Vergleich mit Abbildung 19 ist gut zu erkennen, dass der eingehende Netzwerkverkehr hauptsächlich von den Nutzern des Flash-Client abhängt. Beim ausgehenden Netzwerkverkehr zeichnen sich Spitzen ab. Das Maximum liegt bei $667,56 \frac{kbit}{s}$.

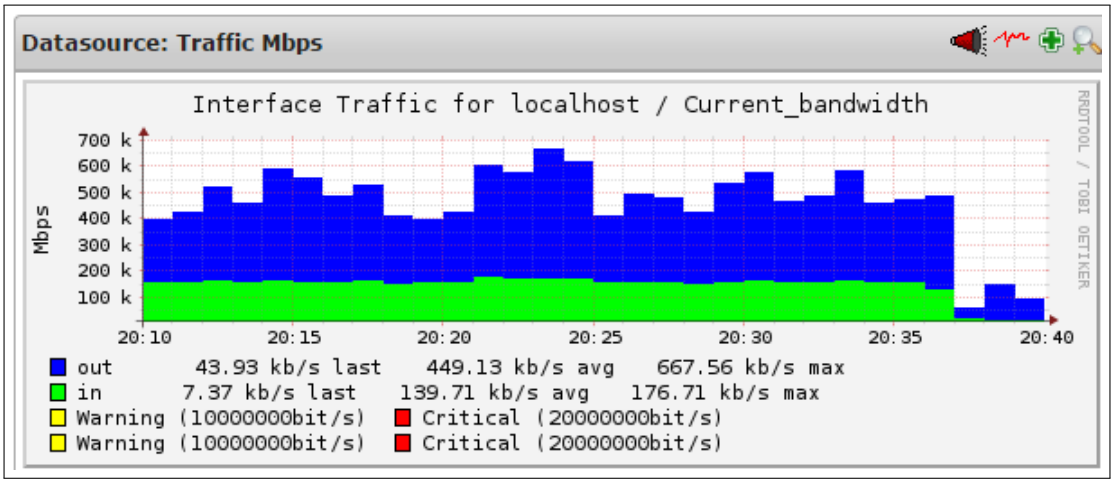

Abbildung 20: abschließender Lasttest – Bedarf an Bandbreite während des Lasttests

Diese Spitzen passen zeitlich mit den Schwankungen der Serverauslastung, die in Abbildung 21 dargestellt ist, zusammen. Zu diesen Zeiten wurden die für diesen Test ausschlaggebenden Event-Nachrichten erzeugt. Beim höchsten Ausschlag ca. 20:13 Uhr wurden Umfragen beantwortet. Hier konnte eine Last von 0,54 erreicht werden. Bei den anderen Spitzen (ca. 20:19 Uhr, 20:28 Uhr und 20:33 Uhr) wurden für eine Minute im Sekundentakt Chatnachrichten geschrieben. Pro geschriebener Textnachricht wurden die MongoDB abgefragt – da im Menü „Chat schreiben" die letzten Nachrichten angezeigt werden – und zudem zwei Channel-Nachrichten verarbeitet (Publish und Broadcast). Insgesamt gab es eine durchschnittliche Last von 0,24.

Eine Audioverbindung war jederzeit ohne Einschränkungen möglich.

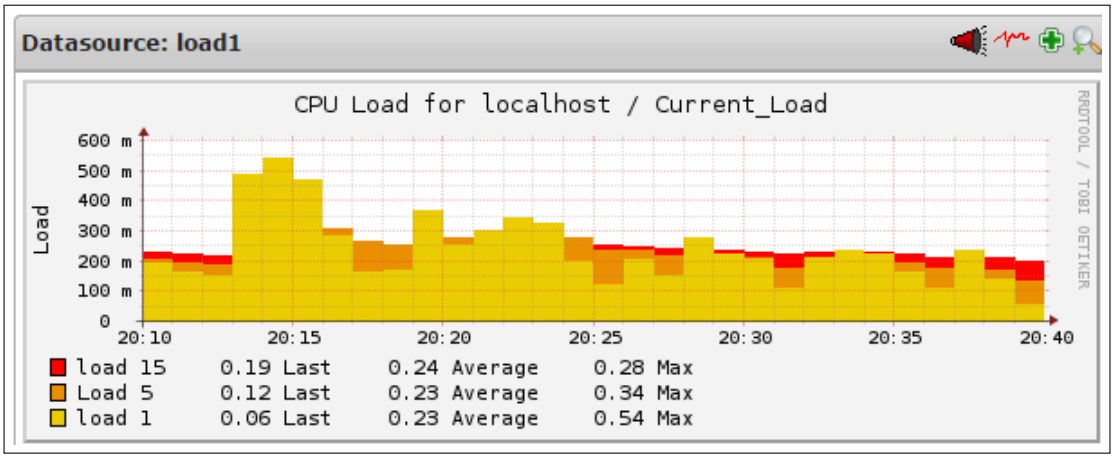

Abbildung 21: abschließender Lasttest – Verlauf der Serverauslastung während des Lasttests

7.4 Bewertung

Da keine komplexen HTML5-Clients an die Benutzer ausgeliefert werden müssen, ist der ausgehende Datenverkehr im Endzustand viel niedriger.

Im Vergleich zum initialen Lasttest, wo eine durchschnittliche Last von 0,29 erreicht wurde, konnte das Abschlussergebnis von 0,24 eine leichte Verbesserung erzielen. Zudem wurde auch der Maximalausschlag von 0,63 im abschließenden Test nicht erreicht.

Insgesamt lässt sich festhalten, dass die benötigte Bandbreite abgenommen hat. Auch die Grundlast des Servers hat sich verringert. Lediglich die Lastspitzen sind erhalten geblieben. Das kann vor allem damit begründet werden, dass sowohl vor als auch nach der Entwicklung Event-Nachrichten in den Redis Channel geschrieben und verarbeitet werden. Die Nachrichten unterscheiden sich nicht voneinander.

Der Test konnte den Server nicht annähernd an seine Lastgrenzen bringen. Die Funktionstüchtigkeit im Rahmen eines Nutzereinsatzes kann angenommen werden.

Selbstkritisch ist feststellbar, dass der Einfluss dritter Programme oder des VMWare-Clusters selbst nicht ausgeschlossen werden kann. Da der Fokus im Rahmen dieser Arbeit auf der Entwicklung eines neuartigen Lehrszenarios liegen soll, wird der durchgeführte Test als ausreichend betrachtet.

8 Nutzertest

8.1 Testszenario

Um die Usability und die Funktionen zu testen, wurde ein Nutzertest durchgeführt. Ca. 20 Teilnehmer – sowohl Studierende, Dozenten, Mitarbeiter als auch Auszubildende – wurden in einem Hörsaal und einem Seminarraum aufgeteilt. Ein weiterer Dozent hielt seine Vorlesung im Seminarraum. Diese wurde über BBB in den Hörsaal übertragen. Der Dozent bereitete sich bewusst vor die Funktionen der Anwendung zu nutzen. Dafür war eine Anpassung seines Lehrkonzepts notwendig.

Die 20 Zuhörer wurden aufgefordert, jegliche Kommunikation zum Dozenten ausschließlich über ihr mobiles Endgerät durchzuführen. Für den Zugang wurden Quick Response (QR)-Codes im Publikum ausgeteilt. Weitere Informationen wurden absichtlich nicht gegeben, damit die Testpersonen möglichst unvoreingenommen die Anwendung testen können. Nach der 25 minütigen Veranstaltung hatten die Testpersonen ausreichend Zeit, die Evaluationsbögen (siehe Anlagen, vgl. [10, S. 8]) auszufüllen.

8.2 Auswertung des Feedbacks

8.2.1 Dozent

Für den Dozenten war die Veranstaltung sehr anstrengend, da es – seiner Einschätzung nach – im Vergleich zu klassischen Vorlesungen technische Hürden gab. Ebenfalls war ein anderer didaktischer Aufbau der Vorlesung erforderlich. Zudem hatte der Dozent den Eindruck, Fachinhalte nicht vollständig vermitteln zu können und das Gefühl von Überforderung entstand. Zusätzlich fehlte ihm die Kenntnis über die bestehende Aufmerksamkeit der Studierenden im entfernten Hörsaal.

Auch wenn diese Faktoren zu einigen Unsicherheiten führen, kann sich der Dozent abschließend betrachtet durchaus vorstellen Vorlesungen dieser Art öfter zu halten. Erste Erfolgserlebnisse sind erkennbar, denn das Umfragetool wurde in Hinsicht auf Bedienung und Feedbackquantität äußerst positiv bewertet. Es ist davon auszugehen, dass mit weiteren übertragenen Vorlesungen die Bedienung immer leichter wird und folglich durch die eintretende Routine die Anfangsschwierigkeiten überwunden werden können [7, S. 5].

8.2.2 Studierende

Eine Aufschlüsselung aller Bewertungen der Evaluationsbögen ist in Abbildung 22 dargestellt. Die Fragen zur Usability der Anwendung fielen durchweg positiv aus. Nach Einschätzung der Testpersonen ist die Anwendung ist einfach, intuitiv und läuft stabil. Das Design wurde von der Mehrheit der Befragten als zweckmäßig bewertet.

Auch die Beurteilungen des Umfragetools sind weitestgehend positiv. Nur drei Befragte gaben an, dass der Zeitraum zur Beantwortung einer Frage zu kurz sei. Hier kann man die Dozierenden besser einweisen, einen ausreichenden Zeitrahmen zum Beantworten der Fragen einzuräumen.

Das meiste negative Feedback bekam der Textchat, obwohl es von technisch Seite keine Probleme gab. Jederzeit konnten Fragen gestellt werden. Ein Teilnehmer hat angegeben, dass seine Frage nicht richtig verstanden wurde. Durch die angestrebte Kurzfassung im Textchat können wichtige Informationen zum Verständnis der Frage verloren gehen. Eine fehlende Übermittlung von Mimik und Gestik erschwert zudem das Erfassen der gestellten Frage. Wenige Zuhörer bemängelten, dass der Dozent auf die gestellte Frage nicht angemessen eingegangen sei. Zum Einen könnte es zu einer zeitlichen Verzögerung bei der Beantwortung gekommen sein, zum Anderen könnte aber auch der Umfang der Antwort nicht ausreichend gewesen sein. Bei der ersten Vermutung kann es durchaus sein, dass der Dozent die Frage zu diesem Zeitpunkt nicht beantworten wollte, weil sie nicht thematisch passte oder er sein Kapitel erst zu Ende lesen wollte. Auch in klassischen Vorlesungen wird dies zum Teil so praktiziert. Jedoch bekommen die Fragenden ein Gefühl dafür, ob der Dozent die eigene Handmeldung wahrgenommen hat oder nicht. Dieses Gefühl fehlt bei der Hörsaalübertragung. In diesem Zusammenhang sollte der Dozent bereits bei der Vorlesungsplanung Zeitslots einbauen, in denen der Chat verfolgt wird. Auch hat der Dozent kein Gefühl dafür, wie ausführlich der Fragende die Antwort braucht, weil er zu diesem keinen Blickkontakt hat. Nur mit Rückfragen, wie z.B. „Ist Ihre Frage damit beantwortet?", erhält der Dozent ein Feedback. Am wenigsten konnten die Fragen schon innerhalb des Chats beantwortet werden. Während der abschließenden Feedbackrunde kam die Rückmeldung, dass einfache Fragen bereits verbal innerhalb der Gruppe geklärt wurden, ohne dabei den Textchat zu nutzen. In klassischen Veranstaltungen kann dieses Verhalten auch beobachtet werden und ist somit nicht negativ zu werten.

In Tabelle 4 ist eine Zusammenstellung aller Antworten aus dem Freitextfeld der Evaluationsbögen dargestellt. In der Spalte „Kommentar der Autorin" sind weitere Informationen, die in einer mündlichen Auswertung zusammengetragen wurden, sowie die entsprechenden Handlungsempfehlungen aufgeschlüsselt.

Abbildung 22: Antwortenverteilung studentisches Feedback

Tabelle 4: Auswertung studentisches Feedback

Nr.	Aussage	Vor-kommen	Kommentar der Autorin
1	Chat auf dem Handy aktualisiert sich nicht automatisch	9	Eine Dauernutzung der entwickelten Anwendung während der gesamten Vorlesung ist kein Teil des Nutzungsszenarios. Zu Vorlesungsbeginn ist ein Sitzungslogin ratsam. Doch nur im Bedarfsfall ist der Handybildschirm zu aktivieren. Eine dauerhafte Verwendung beansprucht den Smartphone-Akku so sehr, dass die Nutzung über einen kompletten Vorlesungstag unmöglich wird. Eine Unterweisung der Teilnehmer im nächsten Durchlauf ist ratsam: • Der Chat sollte live am Beamer gelesen werden. So sind gleichzeitig Vorlesungsinhalte verfolgbar. • Inhaltlicher Austausch mit räumlich entfernten Kommilitonen erfolgt im Bedarfsfall durch Schreiben einer Chatnachricht. • Inhaltlicher Austausch mit der eigenen Lerngruppe vor Ort erfolgt wie in normalen Vorlesungen verbal und leise. Dennoch ist die Erwartungshaltung gegenüber dieser Funktion beeindruckend hoch. Um die Akzeptanz des Gesamtszenarios zu erhöhen, ist über eine Implementation nachzudenken. Eine schnelle Lösung könnte durch die regelmäßige Aktualisierung des Textchat-Frames mit jQuery und Asynchronous JavaScript and XML (AJAX) geschehen, z.B. aller 5 Sekunden.
2	Chat sollte wie bei gängigen Messengern funktionieren (Lesen und Schreiben in einem Fenster)	4	Die Erwartungshaltung gegenüber diesem Feature ist beeindruckend hoch. Um die Akzeptanz des Gesamtszenarios zu erhöhen, ist über eine Implementation nachzudenken. Dafür kann auf der Startseite der „Chat lesen" Button entfernt werden. Im Untermenü „Chat schreiben" müssen dann alle Nachrichten angezeigt werden und nicht nur die letzten drei.

Tabelle 4: Auswertung studentisches Feedback

Nr.	Aussage	Vor-kommen	Kommentar der Autorin
3	Der Chat des BBB-Flash Clients am Beamer konnte im Gegensatz zur HTML-Anwendung kein 8-Bit UCS Transformation Format (UTF8) (Nutzung für Emoticons)	4	Eine Ursache könnte die Inkompatibilität zu UTF8 im verwendeten Zeichensatz des Flash-Chats sein. Andere serverseitige Ursachen bedürfen einer genaueren Betrachtung der Zeichenverarbeitung im Server. Eine schnelle Lösung, die das Problem umgeht, stellt die Implementierung einer Übersetzungstabelle von UTF8 Emoticons in American Standard Code for Information Interchange (ASCII) Zeichen dar. Dennoch scheint die Forderung nach Emoticons im Chat am Beamerbild eher prinzipieller Natur. Es gab im Laufe der Testvorlesung keine Situation, in der sich das Fehlen von Emoticons im Beamerbild nachteilig auswirkte. Das Feedback kann auch mit vorhandenen Zeichensätzen ausgedrückt werden.
4	Qualität der Veranstaltung ist grundlegend von Qualität des Mikrofons und der Lautsprecher abhängig	3	Über den kompletten Testzeitraum war ein Rauschen im Hörsaal zu hören. Ein Lautsprecher ist zeitweise komplett ausgefallen. Die Hörsäle der HfTL sind audiotechnisch zu modernisieren bzw. zu warten.
5	Audiorückkanal zum Fragen stellen fehlt	3	Das Nutzungsszenario von BBB sowie der entwickelten Anwendung kann um Handmikrofone in jedem Raum erweitert und getestet werden: Eine Signalisierung von Wortmeldungen kann über das „Handheben Symbol“ im BBB-Flash-Client oder durch automatische, sich regelmäßig wiederholende Nachrichten im Chat realisiert werden. Es können Rückkoppelungen durch die örtliche Nähe von Mikrofon und Lautsprechern beim Dozenten entstehen. Eventuell muss ein Assistent das Dozenten-Mikrofon per Software im BBB-Flash-Client öffnen oder schließen. Dabei kommt es zu einem Mehraufwand der Techniker. Außerdem sind hohe Weitergabezeiten des Mikrofons zu erwarten, was zu längeren Unterbrechungen der Vorlesung führt, während die Teilnehmer anderer Räume den Grund dafür nicht sehen. Zuvor sollte eine Notwendigkeit der Audiokomponente genauer untersucht werden.

Tabelle 4: Auswertung studentisches Feedback

Nr.	Aussage	Vor-kommen	Kommentar der Autorin
6	Fragen werden oft schon innerhalb der Gruppe (außerhalb des Chats) geklärt	2	siehe Kommentar zu Aussage 1
7	Umfrage mit Timer hinterlegen	2	Gerade die Studierenden, die bei den Umfragen mehr Zeit benötigen, wünschen sich eine Zeitanzeige, die bis zum Ende der Umfrage herunter zählt. Technisch ist diese Forderung nicht ohne Weiteres umzusetzen, da der Dozent die Umfragen manuell schließt und somit den Zeitpunkt selbst bestimmt. Man könnte die Dozierenden aber darauf hinweisen, dass für Umfragen mehr Zeit eingeräumt wird, und ein Schließen der Frage im Vorfeld angekündigt wird, z.B. 10 Sekunden vorher.
8	Dozent war durch die Übertragung innerhalb des Seminarraums 3x zu sehen – das irritiert	2	Im Seminarraum wurde das Sitzungsbild des Dozenten auf ein Smartboard und zusätzlich über den Beamer angezeigt. Darin war unter anderem auch das Videofenster zu sehen. Bei weiteren Tests kann man das Videobild minimieren, so dass es nicht stört.
9	Bei der Umfrage komplette Antwort anzeigen, nicht nur A,B,C,D,...	2	Technisch nicht ohne großen Aufwand umsetzbar, da serverseitig die Antworttexte unbekannt sind. Sie stehen lediglich in den Portable Document Format (PDF) Folien des Dozenten. Es findet serverseitig keine Verbindung zur Umfrage statt.
10	Steckdosen im Hörsaal notwendig, da angeschalteter Bildschirm Akku verbraucht	1	siehe Kommentar zu Aussage 1
11	Schwierigkeiten beim Folgen von Vorlesung und Textchat	1	siehe Kommentar zu Aussage 1
12	Dozent achtet nicht immer auf den Chat → Fragen werden nicht unmittelbar geklärt	1	Nach Rücksprache mit Dozenten ist es manchmal nicht gewollt, den eigenen Redefluss zu unterbrechen (wie in klassischen Vorlesungen).
13	wenig Gelegenheiten Fragen zu stellen	1	

Tabelle 4: Auswertung studentisches Feedback

Nr.	Aussage	Vor-kommen	Kommentar der Autorin
14	viel Ablenkung im dozentenlosen Raum bei Umfragen und Chats	1	
15	Gefahr von Ablenkung durch die Nutzung von anderen Apps	1	
16	Umfragenhistorie	1	Die Umfrageergebnisse werden im Whiteboard gespeichert. Findet eine Aufzeichnung der Sitzung statt, können die Ergebnisse im Nachgang angesehen werden.
17	virtuelles Melden bei Fragen	1	Die nächste Version der Anwendung könnte in regelmäßigen Abständen eine automatisierte Meldung im Chat platzieren, solange die „Hand“ eines Teilnehmers „gehoben“ ist. Das Heben und Senken der Hand kann über Buttons in der Anwendung erfolgen.
18	Web-Links als solche darstellen	1	Ist in nächster Version umsetzbar durch die Suche nach markanten Strings, wie z.B. *„http://”*, *„https://”* oder *„www.”*.
19	vereinzelt serverseitige Ausfälle der Anwendung	1	Nicht reproduzierbarer Einzelfall. Software- und Netzkonfiguration am Client sollten bei nochmaligem Auftreten geprüft werden.
20	Multiple statt Single Choice im Umfragetool	1	Ist in nächster Version umsetzbar. Buttons bei Antwortauswahl müssen gegen Checkboxen getauscht werden. Für jede aktivierte Checkbox muss beim Absenden des HTML-Formulars eine separate Redis-Channel Umfragenachricht geschickt werden.

8.3 Schlussfolgerung

Gerade das studentische Feedback hat gezeigt, dass noch einige Funktionen der Anwendung verbessert werden müssen, bevor das System endgültig für verteilte Vorlesungen genutzt werden kann. Besonders die Einführung von Multiple Choice Antworten und die Neugestaltung der Chatmenüs sind sinnvoll. Auch sollte es möglich werden, Hyperlinks über den Chat generieren zu können. Eine automatische Aktualisierung des Chatfensters und die Notwendigkeit der Verwendung von Emoticons sind zu untersuchen. Ebenfalls ist zu prüfen, ob virtuelle Handmeldungen und ein Audiorückkanal wirklich benötigt werden. Auf die Implementierung eines

Timers sollte vorerst verzichtet werden und dafür den Studierenden mehr Zeit zur Beantwortung der Fragen zur Verfügung gestellt werden. Wenn durch eine erneute Evaluation ähnliche Forderungen gestellt werden, kann über eine Implementierung nachgedacht werden.

Dennoch kann der Test als positiv gewertet werden. Es gab keine Ablehnung gegenüber dieser neuen Art Vorlesungen anzubieten. Der Dozent kann sich vorstellen in Zukunft öfter verteilte Vorlesungen zu halten, wenn die oben genannten Optimierungen umgesetzt werden. Das entwickelte System hat sehr gut funktioniert, allerdings wünschen sich die Nutzer teilweise andere Features, wie z.B. die Optimierung des Textchats. Die marode Hörsaalaudiotechnik ist negativ aufgefallen und sollte vor einem Einsatz des Systems überprüft und evtl. erneuert werden.

Hervorzuheben ist, dass die Testgruppe nur aus ausgewählten Testpersonen bestand, die sehr motiviert und interessiert waren. Die Durchführung eines erneuten Tests innerhalb einer realen Vorlesung mit größerem Auditorium ist ratsam.

9 Zusammenfassung & Ausblick

9.1 Zusammenfassung

Abschließend lässt sich sagen, dass ein möglicher Weg aufgezeigt wurde, wie Vorlesungen mit sehr großen Studierendengruppen abgebildet werden können. Basierend auf den Schnittstellen zu BBB wurde ein Softwareprototyp entwickelt, der verteilte Vorlesungen unterstützen kann. Das BBB-System stellt dabei eine ideale Grundlage zur Weiterentwicklung dar.

Eine grundsätzliche Akzeptanz und Aufgeschlossenheit gegenüber der Lehrform war bei der Testgruppe vorhanden. Trotz dieser guten Resonanz und positiv beendeten Lasttests, ist die vorgestellte Lösung noch ein Prototyp, der verbessert werden sollte, um die Anerkennung dieser neuen Lehrform weiter zu steigern. Die Arbeit zeigt in Unterunterabschnitt 8.2.2 den konkreten Veränderungsbedarf am Prototyp bzw. dem Nutzungsszenario auf.

Zu beachten ist, dass verteilte Vorlesungen dieser Art einen anderen didaktischen Aufbau der Vorlesung verlangen. Die Dozierenden müssen somit für die Vorbereitung dieser Vorlesungsart mehr Zeit einplanen, um z.B. Umfragen zu erstellen und diese in den Foliensatz einzubauen. Während des Nutzertests war es undenkbar auf eine technische Betreuung – sowohl beim Dozierenden als auch im entfernten Hörsaal – zu verzichten. Auch beim zukünftigen Einsatz des Systems sollten technische Betreuer eingeplant werden.

Während der Entwicklung sind keine unüberwindbaren Probleme aufgetreten, und die implementierten Funktionalitäten arbeiteten wie gefordert. Das Veröffentlichen von Redis-Channel Nachrichten und die MongoDB-Abfragen funktionierten während allen Tests ohne Einschränkungen.

9.2 Ausblick

Um den Prototypen als Live-System nutzen zu können, müssen noch einige Optimierungsschritte vollzogen werden. Der Testserver muss für den Produktiveinsatz vorbereitet werden. Dafür ist es unter anderem notwendig das PSA-Verfahren der Deutschen Telekom zu durchlaufen. Des Weiteren ist eine Verbesserung des Sicherheitskonzepts der Anwendung erforderlich. Z.B. ist es kritisch zu betrachten, dass die entwickelte Anwendung auf dem gleichen Webserver wie BBB läuft. So können eventuell vorhandene Sicherheitslücken ausgenutzt werden, um leichter Zugriff auf das Gesamtsystem zu bekommen. Auch ist es immer noch möglich – durch Löschung des Browser-Cookies – an Umfragen mehrmals teilzunehmen. Eine Absicherung könnte erzeugt werden, indem die Benutzer- und Umfragen-ID serverseitig gespeichert und abgefragt werden. Erneute Nutzertests mit dem weiterentwickelten Prototyp und ein Stresstest sind empfehlenswert.

Ebenfalls könnte über die Anbindung einer Lightweight Directory Access Protocol (LDAP)-Authentifizierung nachgedacht werden. Damit ist es Nutzern möglich, sich über die Hochschulauthentifizierung anzumelden, und die hinterlegten persönlichen Daten können automatisch der entwickelten Anwendung übergeben werden. So kann sichergestellt werden, dass nur die richtigen Namen der Teilnehmer verwendet werden und Chatnachrichten eindeutig zugeordnet werden können. Dadurch können den Vorlesungsverlauf störende Nachrichten im Chat minimiert werden. Die verlorene Anonymität kann sich jedoch auch nachteilig auf die Teilnah-

mebereitschaft auswirken. Was den besseren Weg darstellt, sollte vor einer endgültigen Entscheidung genauer untersucht werden.

Einige gewünschte Änderungen aus Tabelle 4 können sofort umgesetzt werden. Z.B. sollte die Möglichkeit gegeben werden, Multiple Choice Fragen zu stellen und Hyperlinks als solche im Chat darzustellen. Darüber hinaus ist es für die nächste Version des Prototyps empfehlenswert, das Chatfenster regelmäßig automatisiert zu aktualisieren und die Chatmenüs zusammenzuführen. Bei anderen Vorschlägen, z.B. dem Audiorückkanal sollten eine Überprüfung der Notwendigkeit stattfinden sowie die Auswirkungen auf den Vorlesungsverlauf beobachtet werden.

Damit die Benutzerfreundlichkeit der entwickelten Anwendung weiter erhöht werden kann, ist es sinnvoll, ein ordentliches Exception-Handling zu implementieren. Außerdem ist es gerade für ausländische Studierende von großem Vorteil, wenn die angezeigten Texte über eingebundene Sprachpakete geändert werden können.

Schlussendlich steht und fällt der Erfolg der aufgezeigten Lösung mit der Qualität der Hörsaalaudiotechnik. Sie muss zuverlässig funktionieren und darf keine Störgeräusche produzieren, denn Fehlfunktionen – gleich welcher Art – wirken sich unmittelbar negativ auf die Akzeptanz der Lösung aus. [7, S. 7]

Literatur

[1] *Anzahl der Studierenden an Hochschulen in Deutschland vom Wintersemester 2002/2003 bis 2015/2016*. URL: http://de.statista.com/statistik/daten/studie/221/umfrage/anzahl-der-studenten-an-deutschen-hochschulen/.

[2] Jutta Arrenberg und Susann Kowalski. *Lernen Frauen und Männer unterschiedlich? Eine Studie über das Lernverhalten von Studierenden*. 2007.

[3] *BigBlueButton Bugreport*. ILIAS, FHL-Master. URL: https://ilias.hft-leipzig.de/ilias/goto.php?target=dcl_48155&client_id=FHL-Master.

[4] *BigBlueButton Docs*. URL: http://docs.bigbluebutton.org/.

[5] *BigBlueButton Entwickler Forum*. URL: https://groups.google.com/forum/#!forum/bigbluebutton-dev.

[6] *BigBlueButton Homepage*. URL: http://bigbluebutton.org/.

[7] Ludger Bischofs u. a. »Erste Erfahrungen mit dem Virtuellen Softwareprojekt«. In: (2003).

[8] *Deutsche Telekom - PSA Verfahren*. URL: https://www.telekom.com/psa.

[9] *Gartner*. 2016. URL: http://www.gartner.com/newsroom/id/3323017.

[10] Kristof Haaser, Meinald T Thielsch und Robert Moeck. »Studentische Lehrveranstaltungsevaluation online: Erfahrungen, Empfehlungen und Standards der Prozessgestaltung«. In: *Psychologiedidaktik und Evaluation VI* (2007), S. 337–346.

[11] Paul Hudson. *PHP in a Nutshell*. O'Reilly Germany, 2006.

[12] Luise Kaufmann und Tobias Welz. »BigBlueButton Dokumentation«.

[13] Luise Kaufmann und Tobias Welz. »BigBlueButton Lasttestergebnisse 2014«.

[14] Michael Kerres und Annabell Preußler. »Zum didaktischen Potenzial der Vorlesung: Auslaufmodell oder Zukunftsformat?« In: *Hochschuldidaktik im Zeichen von Heterogenität und Vielfalt: Doppelfestschrift für Peter Baumgartner und Rolf Schulmeister* (2013), S. 87–97.

[15] Steffen Keßler. *Anpassung von Open-Source-Software in Anwenderunternehmen*. Springer Fachmedien Wiesbaden, 2013. DOI: 10.1007/978-3-658-01955-6.

[16] Wiebke Köhlmann, Nils Dressel und Dustin Wegner. »Erweiterung eines virtuellen Klassenzimmers zur Verbesserung der Zugänglichkeitfür Blinde«. In: (2015).

[17] Sächsischen Staatsministeriums für Wissenschaft und Kunst. »Art und Umfang der Aufgaben an staatlichen Hochschulen im Freistaat Sachsen (Sächsische Dienstaufgabenverordnung an Hochschulen – DAVOHS)«. In: (2011).

[18] Jörg Linder. *Social Semantic Web*. Springer Berlin Heidelberg, 2009. DOI: 10.1007/978-3-540-72216-8_5.

[19] Robert Mertens u. a. »Einsatz von Vorlesungsaufzeichnungen im regulären Universitätsbetrieb.« In: *GI Jahrestagung (1)*. 2004, S. 429–433.

[20] *MongoDB Docs*. URL: https://docs.mongodb.com/manual/mongo/.

[21] *Nagios Check Interface Bandwidth Script*. URL: https://exchange.nagios.org/directory/Plugins/Network-Connections,-Stats-and-Bandwidth/Check-interface-bandwidth/details.

[22] *PHP Marktanteil auf 81,1 Prozent geklettert*. 2013. URL: https://entwickler.de/online/php-marktanteil-auf-811-prozent-geklettert-137114.html.

[23] Aviv Ron, Alexandra Shulman-Peleg und Emanuel Bronshtein. »No SQL, No Injection? Examining NoSQL Security«. In: *arXiv preprint arXiv:1506.04082* (2015).

[24] Lars Schlenker und Susann Beyer. »Online in der Vorlesung–Potentiale digitaler Medien für aktives Lernen«. In: *11. Workshop on e-Learning (WeL2013)*. 2013.

[25] Udo Siebert. »Untersuchung der Integration des Konferenzsystems BigBlueButton in die Infrastruktur der Hochschule für Telekommunikation Leipzig«. Bachelorarbeit. Hochschule für Telekommunikation Leipzig (FH), 2014.

[26] »Sächsische Bauordnung«. In: (2004).

[27] Timo van Treeck, Klaus Himpsl-Gutermann, Jochen Robes u. a. *Offene und partizipative Lernkonzepte. E-Portfolios, MOOCs und Flipped Classrooms*. 2013.

[28] Tobias Welz. »Studienjahresablaufplanung, detaillierte Version«. 2016. URL: https://www.hft-leipzig.de/no_cache/de/zielgruppen/studierende.html?cid=1582&did=9174&sechash=fe0ec02e.

Selbständigkeitserklärung

Hiermit erkläre ich, dass die von mir an der Hochschule für Telekommunikation Leipzig (FH) eingereichte Abschlussarbeit zum Thema

"Entwicklung einer BigBlueButton-basierten Anwendung zur Interaktion zwischen Dozent und Studierenden innerhalb dezentral gehaltener Vorlesungen"

vollkommen selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Abbildungen in dieser Arbeit sind von mir selbst erstellt oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Hochschule oder Universität eingereicht worden.

Leipzig, den 30. September 2016

Luise Kaufmann

Anlagen

Anlagenverzeichnis

Eintrag BBB-Entwicklerforum [5], Nutzung von Redis Channels

```
Autor: Luise Kaufmann
Datum: 04.04.2016

Hello everyone,

I have the following idea:
A meeting is running (flash, including one moderator) and I want
   to ''create'' other users by using the BBB-API (join, redirect=
   false). With the received information (meetingID, userID and
   AuthToken) I want to do something in the running meeting
   without participating the meeting as an own client. The HTML5-
   Client is too excessive for my use case. I only need the
   following Client features: Chat, Emojis and Polling (
   participate).
In the HTML5 Client Overview I read something about 'Redis
   Channels'. Can I use these Channels ''manually'' to send e.g.
   Chat messages and how does it work?

In the future I want to build a web interface which generates API-
   Calls, validate the API-Responses and communicate with the
   Redis Channels. Everything without starting a ''real client'' (
   neither flash nor html).

Thanks in advance
Luise
```

Autor: Anton Georgiev
Datum: 04.04.2016

Hey Luise,

You can use the redis channels manually via redis-cli (lots of guides online). This is the current list of redis-channels in use for the BBB-apps module: https://github.com/bigbluebutton/bigbluebutton/blob/master/bbb-common-message/src/main/java/org/bigbluebutton/common/messages/MessagingConstants.java#L23-L51

You can send a json string with a structure like this: https://github.com/bigbluebutton/bigbluebutton/blob/master/bbb-common-message/src/main/java/org/bigbluebutton/common/messages/SendPublicChatMessage.java

However, you will notice that the messages for communication through redis typically include meetingId, userId and other dynamic data from the running meetings. That makes the manual insertion of messages unreliable. Also I think you will need to make some changes to the backend of BBB if you want to have a user which is not in the meeting to interact via chat/poll/emojis.

If I were you I would look again towards simplifying the html5 client as it already has the redis communication set up. You can easily strip down the functionality or UI which are not needed in your use case. It also has the backend for dealing with emojis and polls. And if you insist on having your user invisible you could perhaps hardcode a username and add some logic to omit it from the userlists.

Regards,
Anton
BigBlueButton developer

Nagios Script, Anzahl der BBB-Nutzer

Listing 6: Nagiosscript – Anzahl der BBB-Nutzer

```
#!/usr/bin/perl
# Perl Programm zum ermitteln laufender BBB-Sitzungen
# Luise Kaufmann, 2015

use strict;
use Term::ReadKey;
use LWP::Simple;
use Digest::SHA1  qw(sha1 sha1_hex sha1_base64);

### Security Salt und IP-Adresse definieren ###
  my $ip=$ARGV[0];
  my $salt=$ARGV[1];

### getMeetings API Call ###
  my $anzahlmeetings=0;
  my $xml= get("http://$ip/bigbluebutton/api/getMeetings?checksum=
     ".checksum("getMeetings".$salt));  #API Call getMeetings
  if ($xml=~/returncode>SUCCESS<\/returncode/){} else {print "
     UNKNOWN\n"; exit 3;}  #Check ob API Call erfolgreich
  if ($xml=~/messageKey>noMeetings<\/messageKey/) {$anzahlmeetings
     =0; print "OK␣-␣0␣User␣sind␣currently␣bei␣BBB␣eingeloggt␣|␣
     users=0;$ARGV[2];$ARGV[3]\n"; exit 0;}  #Check ob Meetings
     laufen
  my @xmltags=split(/></,$xml);  #xml file in Array speichern,
     Trennung an Tags (><)

  for (my $i=0; $i<scalar @xmltags; $i++){if ($xmltags[$i]=~ /^
     meeting$/) {$anzahlmeetings++}}  #Anzahl der laufenden
     Meetings bestimmen, unter <meeting> wird ein Raum beschrieben
      (Tags wurden vorher bei UEberfuehrung in Array entfernt)

  my @meetingids;
  for (my $i=0; $i<scalar @xmltags;$i++) {
    if ($xmltags[$i]=~ /meetingID/){push (@meetingids, $xmltags[$i
       ]);}  #<meetingID>[Meeting ID]</meetingID>, fuer
       getMeetingInfo API Call gebraucht
  }
  foreach (@meetingids) {$_=~ s/^.*>(.*)<.*$/$1/}  #nur [Meeting
     ID] wird gespeichert
  foreach (@meetingids) {$_=~ s/ /%20/g}  #Leerzeichen gegen %20
     ersetzen, fuer URL-Aufruf


  my @meetingnames;
  for (my $i=0; $i<scalar @xmltags;$i++) {
```

```
33      if ($xmltags[$i]=~ /meetingName/){push (@meetingnames,
           $xmltags[$i]);}  #<meetingName>[Name]</meetingName>, fuer
           getMeetingInfo API Call gebraucht
34    }
35    foreach (@meetingnames) {$_=~ s/^.*>(.*)<.*$/$1/}  #nur [Name]
         wird gespeichert
36
37
38    my @moderatorpws;
39    for (my $i=0; $i<scalar @xmltags;$i++) {
40      if ($xmltags[$i]=~ /moderatorPW/){push (@moderatorpws,
           $xmltags[$i]);}  #<moeratorPW>[Passwort]</moderatorPW>,
           fuer getMeetingInfo API Call gebraucht
41    }
42    foreach (@moderatorpws) {$_=~ s/^.*>(.*)<.*$/$1/}  #nur [
         Passwort] wird gespeichert
43
44  #System der UEbergabe von Id, Name und ModeratorPW beruht darauf,
       dass alle zusammengehoerenden Werte nacheinander im xml File
       sind (Id1, Name1, modPW1, Id2, Name2, modPW2 usw.)
45  #der Zusammenhang der Werte wird ueber den gleichen Index der
       Arrays vorgenommen
46
47  ### getMeetingInfo API Call ###
48    my $anzahluser=0;
49    for (my $i=0; $i<scalar @meetingids; $i++){
50      $xml=get ("http://$ip/bigbluebutton/api/getMeetingInfo?
           meetingID=$meetingids[$i]&password=$moderatorpws[$i]&
           checksum=".checksum("getMeetingInfomeetingID=$meetingids[$i
           ]&password=$moderatorpws[$i]".$salt));  #API Call
           getMeetingInfo, wird einmal pro geoeffnetem Raum
           ausgefuehrt
51      if ($xml=~/returncode>SUCCESS<\/returncode/){} else {print "
           UNKNOWN\n"; exit 3;}  #Check ob API Call erfolgreich
52      @xmltags=split(/></,$xml);  #xml file in Array speichern,
           Trennung an Tags (><)
53      my @user=();
54      for (my $i=0; $i<scalar @xmltags;$i++){
55        if ($xmltags[$i]=~/fullName/) {push (@user, $xmltags[$i])}
             #Namen der Teilnehmer werden rausgefiltert, Angabe unter
             <fullName>[Name]</fullName>
56      }
57      foreach (@user) {$_=~s/^.*>(.*)<.*$/$1/;}  #nur [Name] wird
           gespeichert
58      $anzahluser=$anzahluser + scalar @user;
59    }  #Anzahl der Teilnehmer wird aufaddiert (Summe der Namen)
60
61    my $status;
62    my $exit;
```

```
63   if ($anzahluser <$ARGV[2]){$status = "OK"; $exit=0;};
64   if ($anzahluser >=$ARGV[2] && $anzahluser <$ARGV[3]){$status="
        WARNING"; $exit=1;};
65   if ($anzahluser >=$ARGV[3]){$status="CRITICAL"; $exit=2;};
66   print "$status - $anzahluser User sind currently bei BBB
        eingeloggt | users=$anzahluser;$ARGV[2];$ARGV[3]\n";
67   exit $exit;
68
69 ### Subs ###
70   sub checksum{  #Checksumme aus API Call berechnen
71     #use Digest::SHA1  qw(sha1 sha1_hex sha1_base64);
72     my $checksum=sha1_hex($_[0]);
73     return "$checksum\n";
74   };
75
76 __END__
```

Nginx Konfiguration

```
1  server {
2  root /var/www/bigbluebutton-default;
3  listen   80;
4  server_name  212.184.75.131;
5
6  access_log  /var/log/nginx/bigbluebutton.access.log;
7  location ~ \.php$ {
8  fastcgi_split_path_info ^(.+\.php)(/.+)$;
9  #       # NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
10 #
11 #       # With php5-cgi alone:
12 #       fastcgi_pass 127.0.0.1:9000;
13 #       # With php5-fpm:
14 fastcgi_pass unix:/var/run/php5-fpm.sock;
15 fastcgi_index index.php;
16 include fastcgi_params;
17 }
18 ...
55 ...
56 # BigBlueButton landing page.
57 location / {
58 root   /var/www/bigbluebutton-default;
59 index  index.php index.html index.htm;
60 expires 1m;
61 }
```

Listing 7: /etc/nginx/sites-available/bigbluebutton

PHP Extensions für MongoDB und Redis

```
843 ;;;;;;;;;;;;;;;;;;;;;
844 ; Dynamic Extensions ;
845 ;;;;;;;;;;;;;;;;;;;;;
846
847 ; If you wish to have an extension loaded automatically, use the
    following
848 ; syntax:
849 ;
850 ;   extension=modulename.extension
851 ;
852 extension = mongodb.so
853 extension = redis.so
```

Listing 8: PHP Extensions

Funktion für API-Aufruf in der Anwendung

```
37 $getmeetings = getmeetings("2");
38
39
40
41 function getMeetings($wert) {
42 $xml = getAPIurl("getMeetings", "");
43 $err = $xml->returncode[0];
44 $meetingIDs = array();
45 $meetingNames = array();
46 if ($err == "SUCCESS") {
47 foreach ($xml->meetings->meeting as $meeting) {
48 $meetingIDs[] = $meeting->meetingID;
49 $meetingNames[] = $meeting->meetingName;
50 }
51 } else {echo "Keine Sitzungsdaten gefunden!";}
52
53 if ($wert == "0") {return $meetingNames;}
54 if ($wert == "1") {return $meetingIDs;}
55 if ($wert == "2") {return $xml;}
56
57 }
58
59 function getAPIurl ($action, $string) {
60 $checksum = sha1($action.$string.$GLOBALS["salt"]);
61 if ($string == "") {
62 $url = $GLOBALS["api"].$action."?checksum=".$checksum;
63 } else {
64 $url = $GLOBALS["api"].$action."?".$string."&checksum=".$checksum;
65 }
66 $xml = simplexml_load_file(rawurlencode($url));
67 return $xml;
68 }
```

Listing 9: API-Aufruf „getMeetings”

MongoDB Abfrage über Mongo Shell

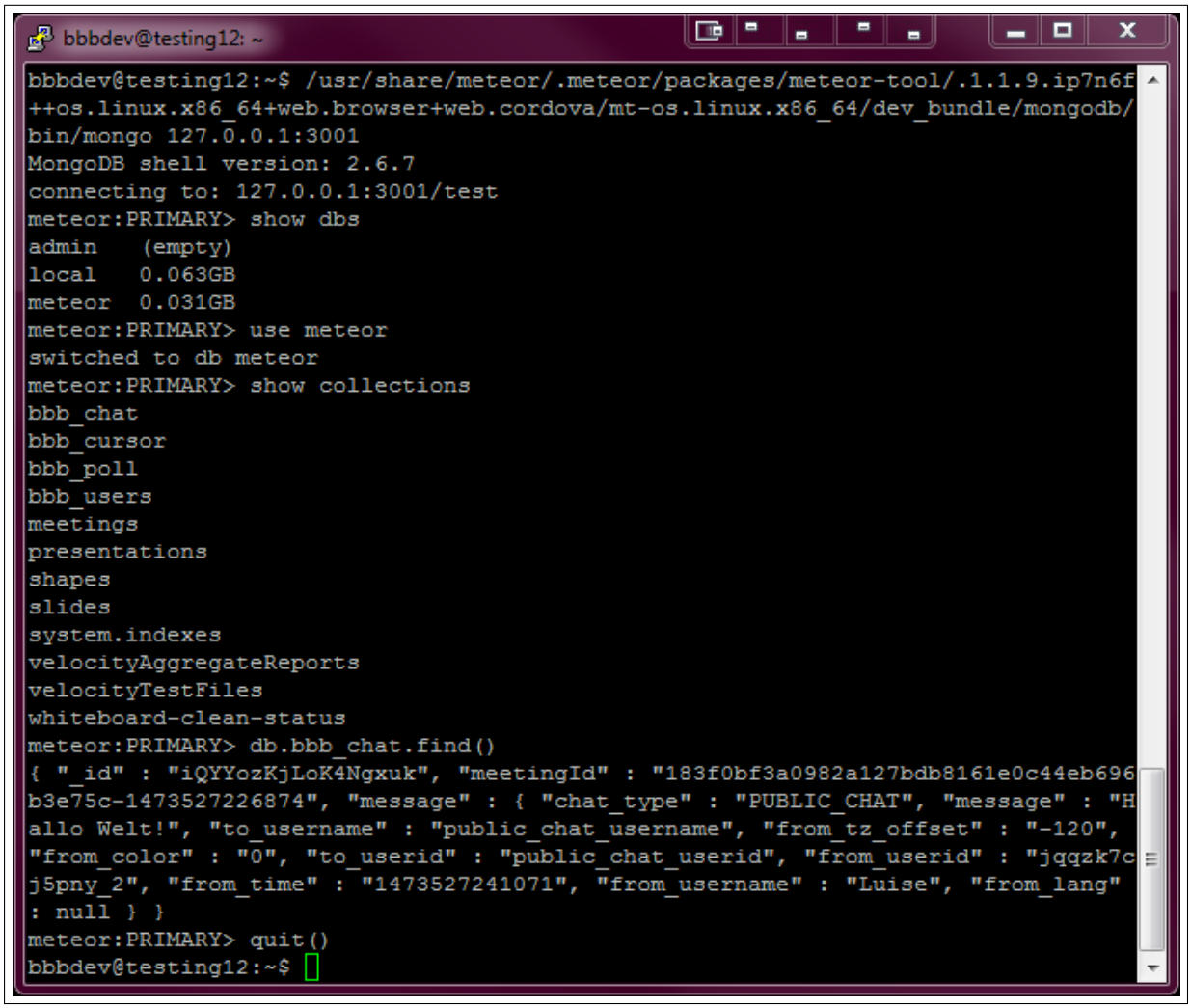

Abbildung 23: Anzeige aller vorhandenen Chatnachrichten

Evaluationsbogen Studierende

HfTL

Bewertungsbogen Studierende

Arbeitsanleitung:

Bitte folgen Sie der gezeigten Veranstaltung und arbeiten Sie aktiv mit. Nutzen Sie für die Mitarbeit ausschließlich das zu testende Tool. Bewerten Sie anschließend die untenstehenden Aussagen.
Die Bewertung kann in 4 Abstufungen angegeben werden. Von „Stimme zu"(100%) bis „Stimme **nicht** zu"(0%).
Wenn Sie die Aussage nicht bewerten können, dann lassen Sie bitte alle Auswahlfelder in der entsprechenden Zeile frei.

Bewertungsskala:

100%	67%	33%	0%
Stimme zu	Stimme überwiegend zu	Stimme überwiegend **nicht** zu	Stimme **nicht** zu

	Aussage	**Bewertung**			
Usability	Der Zugang zur Anwendung war einfach.	a) 100%	67%	33%	0%
	Die Bedienung der Anwendung ist intuitiv.	b) 100%	67%	33%	0%
	Die Anwendung läuft stabil.	c) 100%	67%	33%	0%
	Das Design ist zweckmäßig.	d) 100%	67%	33%	0%
Textchat	Ich konnte jederzeit Fragen stellen (über den Textchat).	e) 100%	67%	33%	0%
	Der Dozent hat meine Frage richtig verstanden.	f) 100%	67%	33%	0%
	Auf meine Fragen wurde angemessen eingegangen.	g) 100%	67%	33%	0%
	Durch den Textchat konnten Fragen bereits innerhalb der Gruppe geklärt werden.	h) 100%	67%	33%	0%
Umfragen	Ich konnte an den Umfragen teilnehmen.	i) 100%	67%	33%	0%
	Ich war motiviert, an den Umfragen teilzunehmen.	j) 100%	67%	33%	0%
	Ich hatte ausreichend Zeit, die Umfragen zu beantworten.	k) 100%	67%	33%	0%

Bitte beachten Sie, dass es Vorder- und Rückseite gibt

HfTL

Bewertungsbogen Studierende

Welches Betriebssystem nutzen Sie?
Android iOS Anderes:

Version:

Welchen Browser nutzen Sie?

Kommentar:

Bitte beachten Sie, dass es Vorder- und Rückseite gibt

Evaluationsbogen Dozierende

HfTL

Bewertungsbogen Dozierende

Arbeitsanleitung:

Bitte halten Sie Ihre komplette Vorlesung inklusive Präsentationsfolien über das Videokonferenzsystem BigBlueButton unter Verwendung des Flash Clients. Halten Sie dabei den Textchat im Auge und beantworten Sie die aufkommenden Fragen zu angemessener Zeit. Stellen Sie regelmäßig Verständnisfragen mithilfe des internen Umfragetools. Bewerten Sie anschließend die untenstehenden Aussagen.
Die Bewertung kann in 4 Abstufungen angegeben werden. Von „Stimme zu"(100%) bis „Stimme **nicht** zu"(0%).
Wenn Sie die Aussage nicht bewerten können, dann lassen Sie bitte alle Auswahlfelder in der entsprechenden Zeile frei.

Bewertungsskala:

100%	67%	33%	0%
Stimme zu	Stimme überwiegend zu	Stimme überwiegend **nicht** zu	Stimme **nicht** zu

	Aussage	**Bewertung**			
Textchat	Ich habe im Textchat die Fragen der Studierenden rechtzeitig gesehen.	a) 100%	67%	33%	0%
	Ich konnte die Probleme der Studierenden aus der formulierten Frage erkennen.	b) 100%	67%	33%	0%
Umfragen	Die Bedienung des Umfragetools war einfach.	c) 100%	67%	33%	0%
	Ich habe ausreichend studentisches Feedback erhalten.	d) 100%	67%	33%	0%
Usability	Es gab während der Vorlesung keine technischen Hürden.	e) 100%	67%	33%	0%
	Ich habe das Gefühl, dass ich Studierenden Fachinhalte vermitteln konnte.	f) 100%	67%	33%	0%
	Insgesamt bin ich zufrieden mit der gehaltenen Vorlesung.	g) 100%	67%	33%	0%
	Ich kann mir vorstellen, in Zukunft mehr übertragene Vorlesungen zu halten.	h) 100%	67%	33%	0%

Kommentar:

Für weitere Anmerkungen nutzen Sie bitte die Rückseite...