HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Internet Performance
und
End User Experience

Studienarbeit

Herbstsemester 2010

Abteilung Informatik

Autoren:	Florian Hungerbühler <fhungerb@hsr.ch> Raphael Neumann <rneumann@hsr.ch>
Betreuer:	Prof. Dr. P. Heinzmann <pheinzma@hsr.ch>
Themengebiet:	Internet-Technologien und -Anwendungen
Industriepartner:	cnlab AG, Rapperswil
Arbeitsdauer:	20.09.2010 - 23.12.2010

Kurzbeschreibung (Abstract)

Abteilung:	Informatik
Name der Studierenden:	Florian Hungerbühler, Raphael Neumann
Studienjahr:	Herbstsemester 2010
Titel der Studienarbeit:	Internet Performance und End User Experience
Examinator:	Prof. Dr. P. Heinzmann
Themengebiet:	Internet-Technologien und -Anwendungen
Projektpartner:	cnlab AG, Rapperswil
Institut:	Institut für Internet-Technologien und -Anwendungen

Kurzfassung

Mit dem cnlab Performance Test können Endkunden die Leistungsparameter ihrer Internet-Verbindung überprüfen. Dabei werden nur Up-/Download-Datenraten und Antwortzeiten der Internet-Verbindung erfasst. Es existieren bisher kaum Daten darüber, wie diese Leistungsparameter das effektive Endkundenerlebnis beim Surfen beeinflussen. Diese Auswirkungen und allfällige Unterschiede aufgrund des benutzten Browsers bzw. Betriebssystems sollen aufgezeigt werden.

Das Endkundenerlebnis ist von vielen Faktoren abhängig und nur schwer objektiv messbar. In dieser Arbeit wurden vier verschiedene Zeitpunkte des Webseitenaufrufs als Indikatoren des Endkundenerlebnisses definiert, welche automatisch (mittels HttpWatch) und teilweise manuell (mittels Stoppuhr) aufgezeichnet wurden.

Um die Auswirkungen der verschiedenen Leistungsparameter zu zeigen, wurden diese mittels eines Netzwerk-Emulators nachgebildet. Der bestehende Emulator bedurfte einiger Verbesserungen, wobei auch die genaue Funktionsweise des Traffic Shapings dokumentiert wurde. Für das Verständnis der Auswirkungen der Leistungsparameter sind einige Punkte des TCP Protokolls genauer erarbeitet worden.

Die Messungen des Endkundenerlebnisses wurden mit dem Internet Explorer 8, Firefox 3.6 und Chrome 7 unter Windows XP sowie Windows 7 beim Aufruf der Webseiten www.20min.ch und www.sbb.ch durchgeführt. Nachfolgend die wichtigsten Ergebnisse der Untersuchungen.

Bei einer Internet-Verbindung mit hohem Bandbreite-Delay-Produkt brachte das TCP Auto-Tuning von Windows 7 nur einen geringen Geschwindigkeitsvorteil im Vergleich zu Windows XP. Allgemein konnte festgestellt werden, dass die Antwortzeit den grösseren Einfluss auf das Endkundenerlebnis hat als die verfügbare Bandbreite. Dies kommt daher, dass die meisten TCP-Verbindungen beim Aufruf einer Webseite einen grossen Teil ihrer kurzen Lebenszeit in der Slow Start-Phase bei geringer Datenrate verbringen. Wie lange diese Phase dauert wird massgeblich durch die Antwortzeit bestimmt. Ferner führt eine hohe Antwortzeit bei hoher Bandbreite zu einem hohen Bandbreite-Delay-Produkt, mit welchem beide getesteten Betriebssysteme nur bedingt umgehen können.

Erstaunlicherweise war der sonst als langsam eingestufte Internet Explorer bei der Darstellung des ersten Webseiten-Inhalts eher schneller als seine Kontrahenten.

Für die untersuchten Webseiten und Leistungsparameter sind keine gravierenden Unterschiede zwischen den Browsern und Betriebssystemen zu beobachten. Erst bei tiefer Bandbreite und hoher Antwortzeit sind Unterschiede in der Rendering-Reihenfolge sichtbar, welche das Endkundenerlebnis stärker beeinflussten als erwartet.

Inhaltsverzeichnis

1. Einleitung

Mit Performance-Messungen wie beispielsweise dem cnlab Performance Test [1] werden diverse Leistungsparameter der Internet-Verbindung auf Client- sowie Serverseite erfasst. Dabei werden meistens die Up-/Download-Datenraten und Antwortzeiten der Internet-Verbindung der Benutzer erfasst, womit aber zur effektiven End User Experience (EUE) beim Verwenden dieser Internet-Verbindung keine Aussage gemacht werden kann.

In dieser Arbeit geht es darum, den Einfluss der Leistungsparameter auf das Verhalten von Internet-Anwendungen zu untersuchen und dabei aufzuzeigen, wie die verschiedenen Parameter die EUE von einzelnen Applikationen beeinflusst. Ein spezieller Augenmerk wird dabei auf die Auswirkungen beim Besuch einer Webseite mit aktuellen Browsern gelegt. Die Leistungs- bzw. Netzwerkparameter werden mit Hilfe eines Netzwerk-Emulators nachgebildet, welcher das Resultat einer früheren Bachelor Arbeit ist (siehe Kapitel 1).

1.1. Anwendungen

Die wichtigsten und verbreitetesten Anwendungen, welche das Internet nutzen, sind im untenstehenden Baum-Diagramm aufgezeigt. Dazu wurden die Hauptverwendungsarten sowie wichtige Aspekte ergänzt.

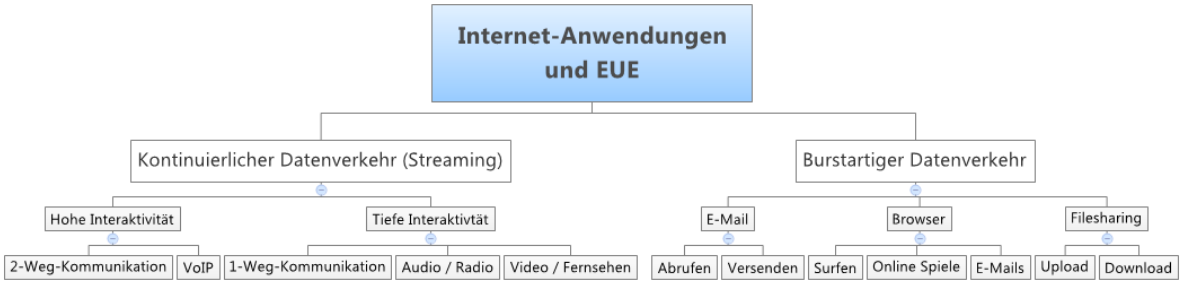


Abbildung 1: Baum-Diagramm Internet-Anwendungen

Wird der verursachte Datenverkehr dieser Anwendungen betrachtet, so können diese grob in zwei Kategorien unterteilt werden:

- **Kontinuierlicher Datenverkehr (Streaming)**
  Für die Anwendung wird ein kontinuierlicher Datenstrom erzeugt, wobei der Verlust einzelner Pakete verkraftbar ist. Meistens wird UDP (User Datagram Protocol) verwendet, in seltenen Fällen auch TCP (Transmission Control Protocol). Typische Streaming-Anwendungen sind: Skype, YouTube, Wilmaa, Zattoo, Internet-Radio.

- **Burstartiger Datenverkehr**
  Der erzeugte Datenverkehr ist nicht kontinuierlich, sondern burstartig. Kurzfristig werden mit hoher Datenrate (höher als im Durchschnitt) Daten übertragen und daraufhin erfolgt eine Übertragungspause, welche normalerweise um einiges grösser ist als die vorherige Übertragungszeit. Dabei kommt pro Anfrage ein Burst zustande, z.B. durch den Aufruf einer Webseite. Bei diesen Anwendungen dürfen keine Pakete verloren gehen, weshalb TCP verwendet wird.

In der folgenden Tabelle werden die verschiedenen Anwendungen genauer charakterisiert.

Anwendung	**Gewünschte erste Reaktion**	**Zu übertragende Datenmenge**	**Erforderliche Datenrate**	**Verkehrstyp**	**Beispielprogramme**
Aufruf von Webseiten E-Mail	1-3s	320 kB [2, S. 17] [3]	0.8-2.5 Mbit/s	Anfrage und Antwort, asymmetrischer Verkehr	Internet Explorer Firefox Chrome
Betrachtung von Videos	bis zu 10s	je nach Streaming-Dauer	200-900 kbit/s[1] [4]	Streaming unidirektional	Browser mit Flash-Plugin (YouTube)
Anhören von Musik	bis zu 10s	je nach Streaming-Dauer	64-128 kbit/s	Streaming unidirektional	Internet Radio
VoIP	< 300ms	je nach Gesprächsdauer	64-128 kbit/s	Streaming, bidirektional	Skype
Filesharing	kaum relevant	je nach Download	beliebig	Up- & Download ausgelastet	BitTorrent

Tabelle 1: Datenflusscharakter von versch. Internetanwendungen

Eine grobe qualitative Analyse der Verkehrstypen bzgl. der Leitungsparameter ergibt folgendes Bild:

Kontinuierlicher Datenverkehr (Streaming)

Ist die Datenquelle nicht immer verfügbar, so fällt dies bei den Streaming-Anwendungen sehr schnell auf, da dort kontinuierlich Daten fliessen. Durch Caching (Zwischenspeicherung) und verzögertes Abspielen des Streams kann dieses Problem etwas abgefangen werden.

Falls UDP verwendet wird, ist nicht sichergestellt, dass alle Datenpakete in der richtigen Reihenfolge ankommen. Die Applikation muss sich selber um die Reihenfolge kümmern, wobei wiederum der Cache ins Spiel kommt. Die Umsortierung der Pakete kann jedoch nur erfolgen, wenn die Pakete nicht allzu spät ankommen. Kommt das Paket nach seinem Abspielzeitpunkt an, so muss es verworfen werden.

[1] Falls Video ohne Caching abgespielt werden soll (Übertragungszeit = Videodauer)

Beim Audio-Streaming sind durchschnittliche Datenraten von 64 - 128kbit/s ausreichend. Auch können relativ grosse (einige 100ms) Round Trip Time (RTT) verkraftet werden. Temporäre Reduktionen der Datenraten oder sogar Unterbrüche können relativ gut durch einen grossen Cache abgefangen werden. Je grösser der Cache, desto grössere Unterbrüche sind verkraftbar, ohne dass der Hörer etwas bemerkt. Um grosse Caches zu füllen, muss man bei Beginn der Übertragung jedoch lange warten, bis man den Stream anschauen/hören kann.

Einzelne Paketverluste werden vom Benutzer nicht bemerkt, da nur ein kleiner Bildfehler oder ein Sprung in der Tonspur entsteht. Doch führen bei Live-Videostreaming Paketverluste von 1% und mehr bereits zu einem nicht mehr geniessbaren Bild [5].

VoIP fällt ebenfalls in die Streaming-Kategorie, ist jedoch aufgrund seiner hohen Interaktivität sehr zeitkritisch. Die RTT darf maximal 300ms betragen, ansonsten leidet die Interaktivität und man fällt sich gegenseitig ins Wort, da die Antworten des Partners zu spät ankommen. Ein zu grosser Cache ist darum kontraproduktiv. Ein kleiner Cache ist jedoch nötig um den Jitter abzufangen und ebenfalls für die Umsortierung der Pakete.

Je nach Audiocodec und Audioqualität wird eine Bandbreite von 64 – 128 kbit/s benötigt. Bei VoIP sind Gespräche mit 10% Paketverlust noch knapp möglich [6]. Dabei sind die Störungen gut wahrnehmbar, die Gesprächsqualität ist jedoch in etwa vergleichbar mit einem durch Umgebungsgeräusche gestörten Telefonat. Bei gewissen Codecs wird versucht kurzfristige Aussetzer im Datenstrom zu überbrücken, indem versucht wird aus den angekommenen Paketen das fehlende zu berechnen (Packetloss Concealment).

Burstartiger Datenverkehr

Ist eine Webseite kurzzeitig nicht erreichbar, so fällt dies beim Surfen nur bedingt auf, da oft die Interaktivität nicht so gross ist. Das heisst, oft wird eine Seite angefordert und dann in Ruhe gelesen, wobei natürlich auch wesentlich dynamischere Seiten existieren.

Die RTT ist hier auch sehr wichtig, da beim Aufbau einer TCP Verbindung aufgrund des Three-Way-Handshakes jeweils 1.5 RTT benötigt werden und pro Anfrage beim Aufbau einer Webseite ebenfalls eine RTT gewartet werden muss. Die aktuellen Browser verwenden jedoch verschiedene Techniken wie parallele Verbindungen und Pipelining (siehe Kapitel 5.1), um die Auswirkungen dieser Verzögerungen zu minimieren. Bei Paketverlusten löst TCP ein erneute Übermittelung (Retransmit) der verloren gegangenen Pakete aus. Dadurch wird jedoch auch die Verbindungsgeschwindigkeit temporär gesenkt, da TCP bei Paketverlusten von einer Überlastsituation im Netzwerk ausgeht.

1.2. Untersuchte Anwendungen / Systeme

Diese Arbeit beschränkt sich auf die genaue Analyse der EUE im Zusammenhang mit Browsern. Dies ist die Anwendung, welche im Internet am häufigsten gebraucht wird und auch am ehesten damit in Verbindung gebracht wird. Die Wahrnehmung der Leistungsparameter der Internet-Verbindung ist beim Surfen sehr ausgeprägt, da nur schon kurze Wartezeiten auffallen.

Gemäss einer Studie von Statcounter [7] (siehe Abbildung 2) sind Anfang September 2010 die Browser Internet Explorer, Firefox und Chrome die weltweiten Top 3-Browser. Aus diesem Grund werden

in dieser Arbeit diese drei Browser untersucht, wobei nur folgende Versionen betrachtet werden: Internet Explorer 8, Firefox 3 und Chrome 6. Genauere Informationen zum Testsystem sind in Kapitel 9.8.2 zu finden.

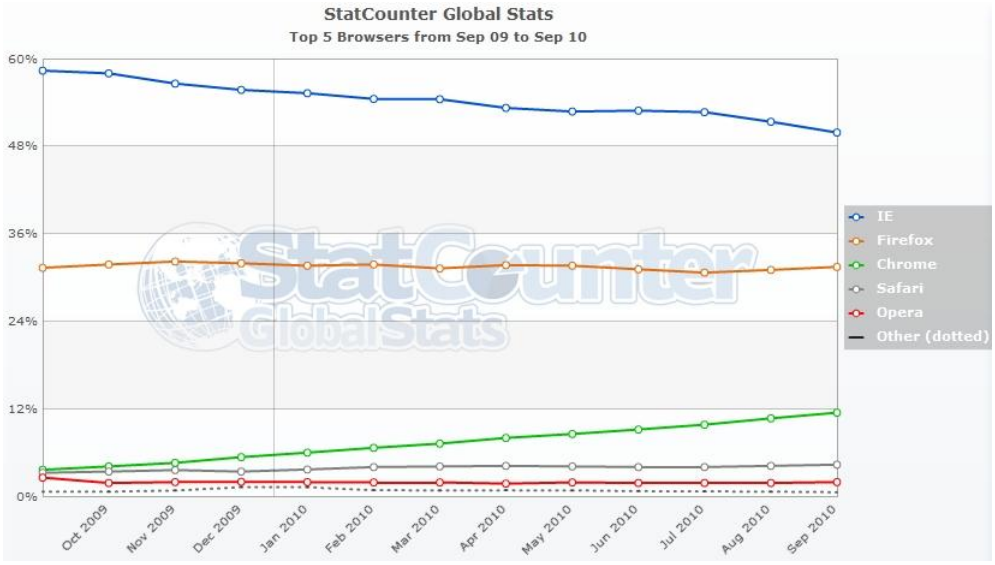


Abbildung 2: Statistik weltweite Browserverteilung von StatCounter [7]

Bei den Betriebssystemen werden nur Windows XP sowie Windows 7 betrachtet. Microsoft Windows wurde ausgewählt, da zum einen der Internet Explorer nur dort nativ läuft und es das Betriebssystem mit dem höchsten Marktanteil ist. Gemäss StatCounter [8] ist Windows XP weltweit und in Europa immer noch die Nummer eins und Windows 7 hat erst vor kurzem Windows Vista als Nummer zwei abgelöst. Dieser Trend kann in Abbildung 3 gut nachvollzogen werden.

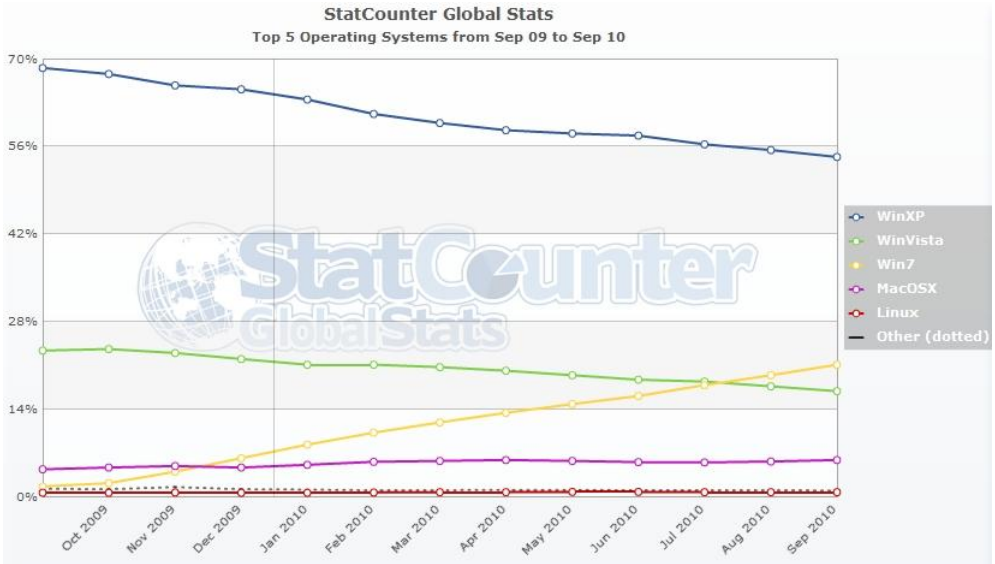


Abbildung 3: Statistik weltweite Betriebssystemverteilung von StatCounter [8]

Dasselbe spiegelt sich auch in der Client-Statistik [9] vom cnlab Performance Test wieder.

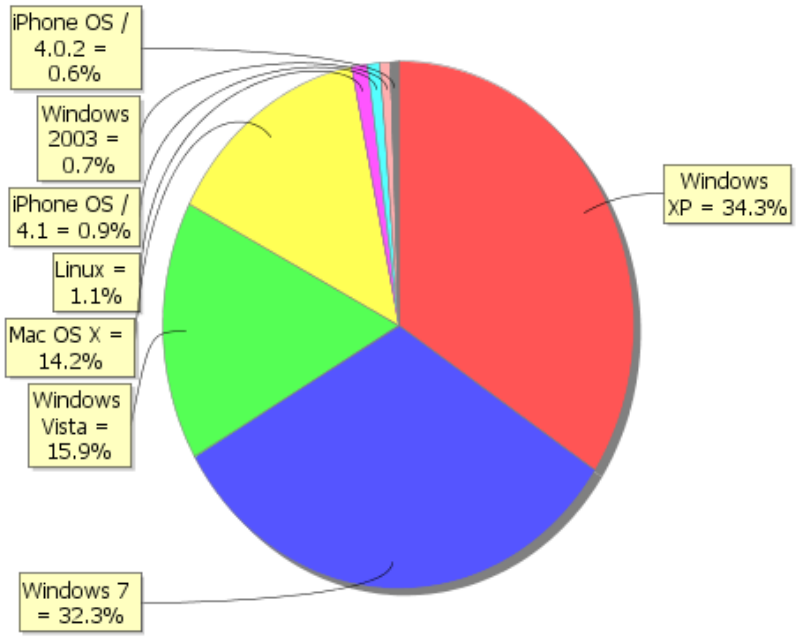


Abbildung 4: Statistik Browserverteilung, cnlab Performance Test, Stand Okt. 2010 [9]

Ein Vergleich von Windows XP und Windows 7 ist insofern interessant, da seit Windows Vista der „Next Generation TCP/IP Stack“ mit „Receive Window Auto-Tuning“ eingesetzt wird, welcher unter Windows XP noch nicht existierte.

Durch die Untersuchung der verbreitetesten Browsern und Betriebssystemen wird der grösste Teil aller Internetbenutzer abgedeckt und die gewonnenen Resultate sind praxisrelevant.

2. Definition End User Experience

2.1. Einleitung

Nicht nur im täglichen Leben wird versucht, den Kunden zufrieden zu stellen. Sondern auch im Internet wird das Ziel eines zufriedenen Kunden angestrebt. Dies trifft primär auf kommerzielle Webseiten wie Online-Shops zu, aber auch für andere Seiten. Denn nur wenn die Kunden mit der Webseite zufrieden sind, wird sie erneut besucht und auch weiterempfohlen.

Um viele Seitenbesuche zu erreichen, muss also das Erlebnis für die Benutzer beim Besuch der Webseite stimmen. Je nach Anwendungsfall (Surfing, VoIP, Online Games, Streaming usw.) und Benutzertyp bzw. Zielpublikum, unterscheiden sich jedoch die Anforderungen an eine Webseite. In dieser Arbeit geht es primär um den Anwendungsfall „Surfing“ und „Browsing“. Gemeint ist damit der Aufruf von Webseiten im Browser, zum Zweck der Informationssuche. Folgende Beispiele sind denkbar: Fahrplan-Abfrage, Lesen von aktuellen Nachrichten (Tagesanzeiger, 20min, CNN), Recherche zu einem bestimmten Thema usw.

Selbst bei Fokussierung auf diesen einen Anwendungsfall gibt es verschiedene Kriterien anhand deren ein Benutzer die Webseite beurteilt:

- Inhalt: Informationsqualität und -tiefe
- Gestaltung: Übersichtlichkeit, Attraktivität, Aufbereitung der Informationen
- Anzeige: Geschwindigkeit und Ablauf des Webseitenaufbaus

Bei der Beurteilung dieser Aspekte spielt hauptsächlich die subjektive Wahrnehmung des Benutzers hinein. Bei den ersten beiden Punkten noch stärker als beim letzten. Für die Untersuchung der End User Experience beim Surfen beschränkt sich diese Arbeit auf den letzten Aspekt, da nur dort die Netzwerkparameter einen Einfluss haben.

Für den Rest der Arbeit wird der Begriff EUE synonym für die Aspekte Geschwindigkeit und Auflauf des Webseitenaufbaus verwendet.

2.2. Bedeutung

Seit Juni 2008 gibt es die Informatik-Konferenz „Velocity“ [10] (Geschwindigkeit), welche sich nur dem Thema „Web Performance“ widmet. Sie findet seither jährlich stattfindet und wird gut besucht. Im Jahr 2010 gab es sogar zwei Konferenzen: Eine wie gewohnt in Kalifornien USA und eine in China. Sie wird jeweils vom internationalen O’Reilly Verlag organisiert.

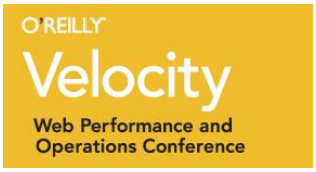


Abbildung 5: Logo Velocity Konferenz

Das Motto dieser Konferenz ist „fast by default“ (standardmässig schnell), was vermitteln soll, dass Geschwindigkeit eine sehr wichtige Funktionalität ist. Ein weiterer Standpunkt ist, dass der Erfolg einer Firma im Internet sehr stark von der wahrgenommenen Geschwindigkeit ihres Auftritts abhängig ist. Dazu gibt es Zahlen als Beweis [11]. An der Velocity wird vermittelt, dass ein ganzheitlicher Blick auf die Web Performance nötig ist, um viel zu erreichen. Es wird nicht mehr wie früher nur das Netzwerk betrachtet [12]. Dies spiegelt sich auch in der Auswahl der unterschiedlichen Referenten an

der Konferenz wieder. Dabei liegt der Fokus primär auf der Optimierung der Datenauslieferung und zwar so, dass der deren Bearbeitung beim Benutzer möglichst schnell abläuft [13].

Steve Souders [14] ist bezüglich Web Performance eine sehr erfahrende Person, da er früher bereits für Yahoo viele Arbeiten zu diesem Thema durchgeführt hat und auch zwei sehr interessante Bücher dazu veröffentlich hat (siehe Kapitel 9.6.3). Er ist Co-Vorsitzender der Velocity-Konferenz und hat an diversen anderen relevanten Projekten betreffend Web Performance wie beim Browservergleich Browserscope [15] oder dem Firefox Plugin Firebug (siehe Kapitel 9.7.1) mitgearbeitet. Mittlerweile arbeitet er für Google und bezeichnet sich selber als Performance Evangelist.

Seit 2008 hat auch Google sehr viel zu einem schnelleren Internet-Erlebnis beigetragen [17]. Sie entwickelten Geschwindigkeits-optimierte Programme wie den neuen Browser Chrome inkl. das dazugehörige Mess-Plugin Speed Tracer (siehe Kapitel 0) oder das Tool Page Speed [16], das Webentwicklern erlaubt ihre Webseite auf Performance-Optimierungen zu untersuchen. Google hat sogar ein neues Protokoll namens SPDY [19] („SPeeDY“ (schnell) ausgesprochen) entwickelt, welches als Ersatz von HTTP für die Auslieferung von Webinhalten dienen soll. Durch verschiedene Verbesserungen soll die Ladezeit einer Webseite verkürzt werden und somit das Surf-Erlebnis verbessert werden [17]. Weiterhin gibt es seit November 2010 ein neues Modul (mod_pagespeed) für den weit verbreiteten Apache Webserver. Es beruht auf dem zuvor erwähnten Page Speed und ist in der Lage, in Echtzeit diverse Optimierungen am Webseiteninhalt vorzunehmen und damit für eine schnellere Auslieferung der Daten an den Benutzer zu sorgen.

Seit April 2010 wird die Ladegeschwindigkeit einer Webseite auch als weiteres Kriterium für die Platzierung der Suchresultate in der Google-Suche aufgenommen [20]. Da Google mit Abstand die am meisten genutzte Suchmaschine ist, wird auch der Anreiz für Webentwickler grösser, ihre Seiten diesbezüglich zu optimieren.

2.3. Einflüsse auf die EUE / Umfeldanalyse

Es gibt viele verschiedene Parameter, welche die EUE beeinflussen, da zwischen Client und Server viele verschiedene Komponenten bei der Datenübertragung involviert sind. Die Abbildung 6 zeigt eine ganzheitliche Sicht auf die wichtigsten Einflüsse und enthüllt, wo diese anzutreffen sind.

Meistens wird nur an die Internetanbindung des Benutzers gedacht bzw. primär an die Download-Geschwindigkeit des Internetabos. Verfügt der Benutzer jedoch über einen schnellen Internetanschlüsse mit 100 Mbit/s Bandbreite im Download und ist via einen WLAN-Router mit dem 802.11g Standard angeschlossen, so beschränkt diese Funkverbindung den Datendurchsatz. Die Beschränkung tritt meistens jedoch schon bei 50 Mbit/s Anschlüssen auf, da die 54 Mbit/s Datendurchsatz bei 802.11g ein theoretischer Maximalwert sind, der nur bei Idealbedingungen erreicht wird. Auch noch tiefere Geschwindigkeiten können betroffen sein, wenn die WLAN-Bandbreite durch massive Funk-Störungen oder viele Kollisionen stark sinkt. Ebenso spielt auch die Rechenleistung des Clients einen nicht zu unterschätzenden Einfluss, da dieser die empfangenen Daten zuerst einmal für die Darstellung der Webseite verarbeiten muss und dabei beispielsweise von einem Virenscanner gebremst werden kann.

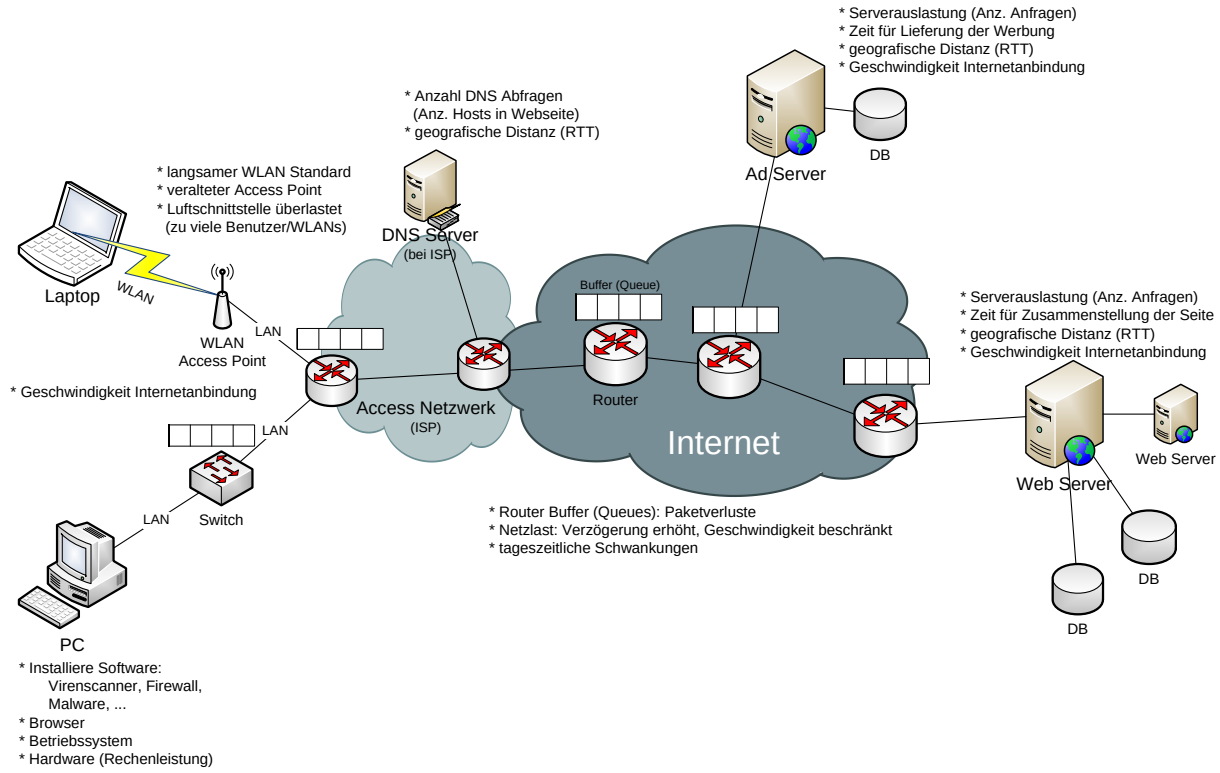


Abbildung 6: Einflüsse auf die Netzwerkperformance

Auch die Serverseite hat einen entsprechenden Einfluss. Ist der Server aufgrund hoher Benutzerzahlen belastet, so braucht er länger für die Auslieferung der angefragten Ressourcen an den Client, da die Bandbreite und die CPU über alle Benutzer bzw. Anfragen aufgeteilt werden muss. Oft ist Werbung oder anderer externer Inhalt in einer Webseite eingebunden, der dann von anderen Servern geladen werden muss. Diese sind eventuell stärker als der eigentliche Webserver der Webseite belastet oder weiter entfernt, was eine höhere RTT und damit grössere Verzögerungen bei der Datenübertragung mit sich zieht. Diese externen Server kann man als Webentwickler jedoch nicht kontrollieren. Ein Ausfall davon kann je nach Wichtigkeit und Einbettung gravierende Folgen für die EUE haben (siehe Kapitel 5.3), da ein wichtiger Teil des Inhalts fehlt oder stark verzögert wird. Ebenfalls ist pro Host eine weitere DNS Abfrage nötig, deren Bearbeitung stark davon abhängt, ob sich die Antwort im Cache befindet oder nicht. Beispielsweise dauert eine DNS-Abfrage von www.sandiego.com 645ms, jedoch sobald diese zwischengespeichert ist nur noch 3ms. In diesem Zusammenhang ist das „DNS Prefetching“ interessant (siehe Kapitel 7.3.5).

In Abbildung 7 wurden einige der wichtigsten Punkte von der Umfeldanalyse dargestellt sowie weitere Punkte ergänzt, welche bei der Messung der Browser EUE eine Rolle spielen.

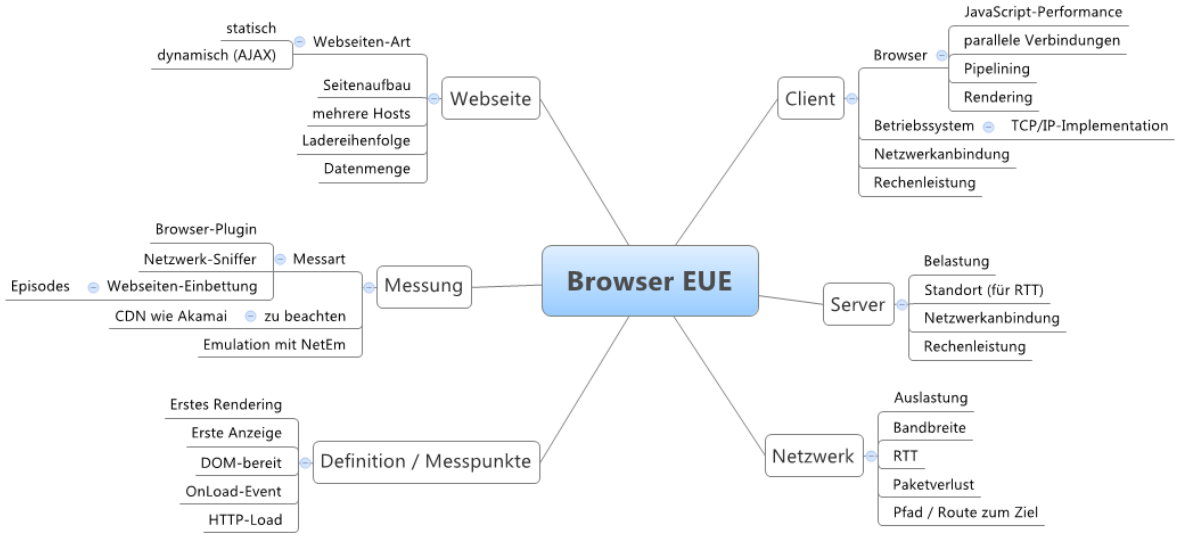


Abbildung 7: Mind-Map Browser EUE

In dieser Arbeit können jedoch aufgrund des Umfangs nicht alle Punkte behandelt werden. Einige Punkte entziehen sich unseres Einflusses und unserer Beobachtungsmöglichkeiten.

2.4. Emulation

Mit der NetEm-Box (siehe Kapitel 1), welcher das Resultat einer früheren Bachelor Arbeit ist, werden die verschiedenen Netzwerkparameter emuliert. Damit wird die Variation im Netzwerk nachgestellt, welcher in der untenstehenden Grafik eingerahmt ist.

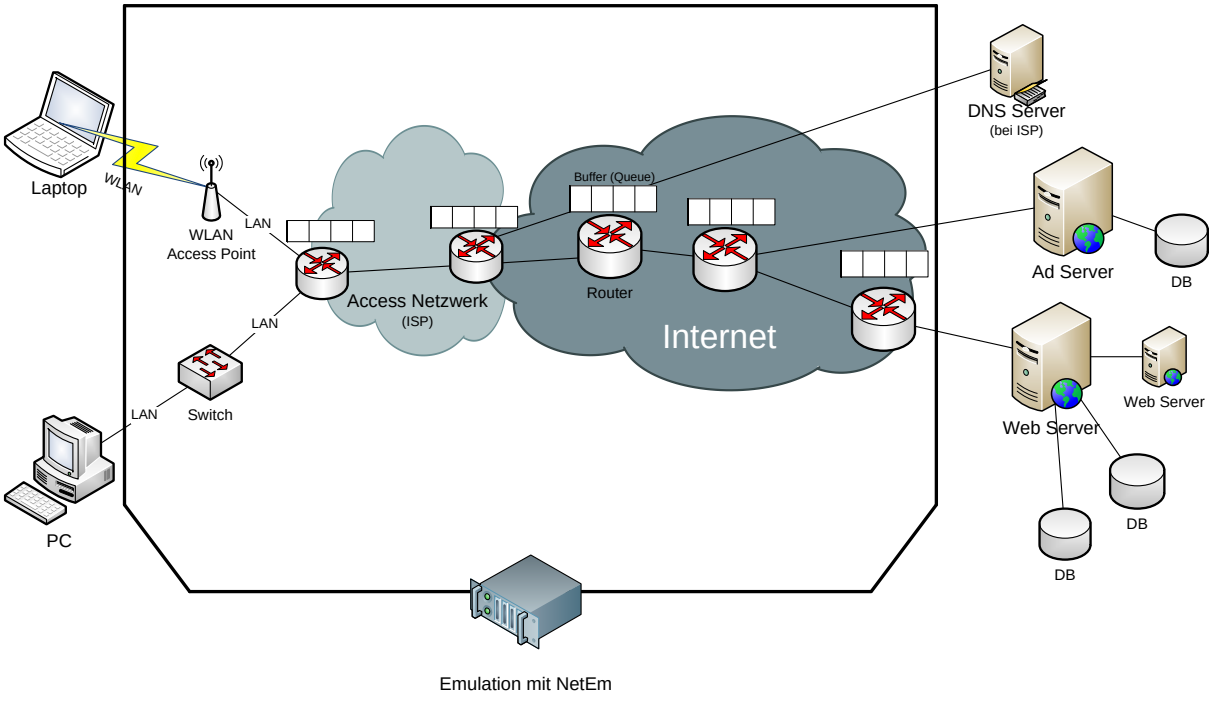


Abbildung 8: Durch die NetEm-Box emulierter Teil des Netzwerks

Vereinfacht dargestellt, sieht der Aufbau wie folgt aus:

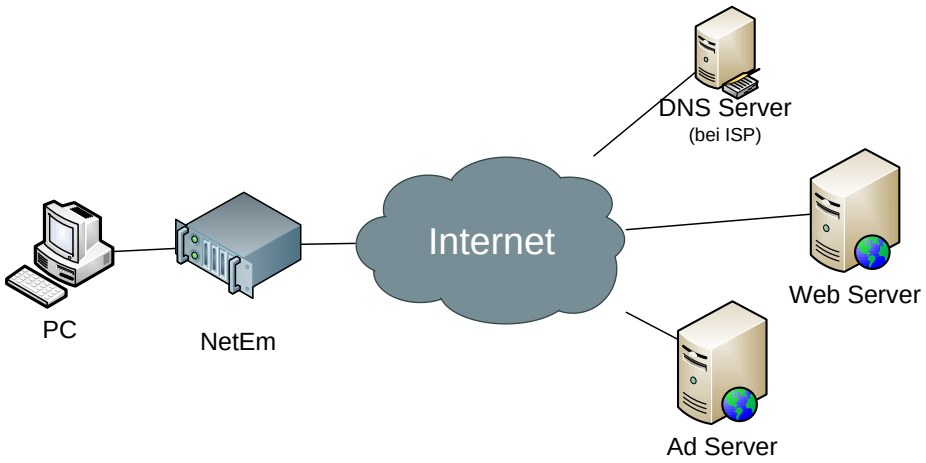


Abbildung 9: Vereinfachte Abbildung der NetEm-Box Emulationsanordnung

So wie die NetEm-Box in dieser Arbeit eingesetzt wird, ist die durch den Emulator eingeführte Variation zu allen Zielservern gleich. Dies entspricht jedoch nicht genau der Realität. Oft sind die Netzwerkbedingungen unterschiedlich zu den Servern, wobei in dieser Anordnung die Variation des Internets immer noch vorhanden ist.

Um die Variationen zu den Servern einzeln bestimmen zu können, muss eine andere Anordnung gewählt werden. Oft ist die Verzögerung zum DNS-Server des ISP (Internet Service Provider) viel kleiner als jene zum Webserver oder es wird ein anderer DNS-Server verwendet, der weniger schnell antwortet, da er nicht im schnellen ISP-Netzwerk ist. Es könnte, wie in Abbildung 10 dargestellt, pro Server ein separater Netzwerk-Emulator verwendet werden. Eine andere Möglichkeit wäre, dass man im Emulator verschiedene Filter pro Server konfiguriert, was jedoch über die Web-Oberfläche der NetEm-Box nicht unterstützt wird.

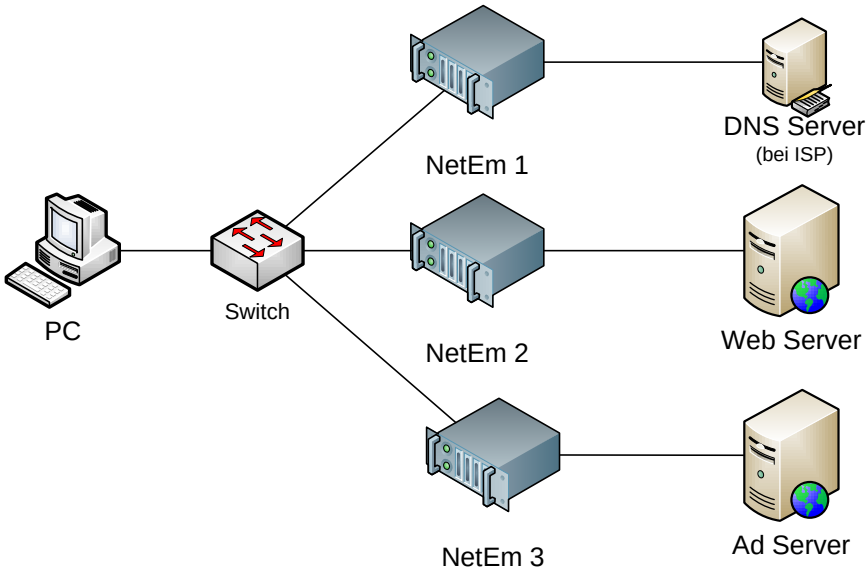


Abbildung 10: Realitätsgetreuere Netzwerkemulation

2.5. Messbare Kriterien

Wie im Kapitel 2.1 beschrieben, spielt beim Surfen auch die subjektive Wahrnehmung des Benutzers eine Rolle. Je nach aktuellen Gemütszustand und Erfahrungslevel des Benutzers ist er mehr oder weniger geduldig beim Laden einer Webseite und beurteilt die Ladegeschwindigkeit unterschiedlich. Für die Messung gilt es also möglichst objektiv messbare Kriterien über die Schnelligkeit der Webseitendarstellung. Dieses Resultat wird jedoch durch das Zusammenspiel vom Aufbau der Webseite, der Leistungsfähigkeit des Übertragungssystems (Server, Netzwerk) und Leistungsfähigkeit des Darstellungssystems (Client-Rechner, Betriebssystem, Browser) beeinflusst.

Es genügt nicht, einfach nur die Zeit von der Eingabe der URL bis zur kompletten Darstellung der Webseite im Browser zu messen. Unter Umständen ist dies für den Benutzer wenig interessant, denn er ist bereits zufrieden, wenn ihm ein Teil der Webseite angezeigt wird und er z.B. auf einer News-Webseite die erste Schlagzeile lesen kann. Wird bei einer langen Seite zuerst der obere Inhalt geladen und dargestellt, so hat der Benutzer bereits das Gefühl, die Webseite sei komplett geladen worden. Dass im Hintergrund der nicht sichtbare Teil der Webseite fertig geladen wird, bemerkt der unerfahrene Benutzer kaum. Meistens wird der Benutzer nicht auf die Statusleiste des Browsers achten, wo der Fortschritt der Ladevorgangs dargestellt wird.

Aus diesem Grund müssen verschiedene Werte beim Laden der Webseite gemessen werden [21]. Die EUE kann nicht über einen einzigen Wert abgebildet werden. Es ist auch von der Gestaltung und des Aufbaus der Webseite abhängig, wann die Webseite für den Besucher benutzbar wird.

Bei der Recherche im Internet bzgl. des Seitenladevorgangs wurden verschiedene Zeitpunkte bzw. Events gefunden, welche unterschiedlichen Fortschritte signalisieren. Im Folgenden werden diese Events aufgelistet und bezüglich der Eignung für die nachfolgenden Messungen der EUE beurteilt. Im Kapitel 7.2 können zu einigen Events Screenshots gefunden werden.

Erstes Rendering

Zu diesem Zeitpunkt beginnt der Browser mit dem Rendering. Dies sagt jedoch nichts dazu aus, wie viel der User bereits von der kompletten Webseite sieht. Dies variiert je nach Aufbau der Seite. Erste Testmessungen haben gezeigt, dass zum Zeitpunkt des „Ersten Renderings“ im Browser oft noch kein Element der Webseite anzeigt wird, da es sich z.B. um einen weissen Hintergrund handelt

Dieser Event kann mit einem Tool gemessen werden.

Erste Anzeige

Wenn der Benutzer zum ersten Mal etwas von der aufgerufenen Webseite im Browser sieht, wird die Zeit gemessen. Dabei wird ein allfälliger einfarbiger Hintergrund nicht beachtet. Es muss sich um ein Bild oder um Text der Webseite handeln.

Dieser Event muss von Hand gestoppt werden. Von den meisten Tools wird er nicht unterstützt oder es ist unklar, was der Benutzer zu diesem Zeitpunkt genau sieht.

Seite benutzbar

Sobald der Benutzer den ersten Nutzen von der Webseite hat, wird diese Zeit gestoppt. Dies ist je nach Webseite anders und hat auch viel mit dem Layout und dem Aufbau der Seite zu tun. Dieser Zeitpunkt muss pro Webseite festgelegt werden. Für die später untersuchten Webseiten ist im Kapitel 7.2 festgelegt, was die erste mögliche Benutzung ausmacht. Dabei kann es sich um die erste Schlagzeile einer Online-Zeitung handeln oder auch nur um Suchfeld bei einer Suchmaschine.

Da dieser Event je nach Webseite stark schwankt, ist eine automatische Auswertung nicht möglich und die Zeit muss von Hand gestoppt werden. Da die aktuellen Browser schon bevor der ganze HTML Code geladen wurde, einen Teil der Webseite darstellen können, liegt dieser Event oft noch vor dem DOM-bereit.

DOM-bereit

Dieser Event (auch DOMContentLoaded genannt) wird ausgelöst, sobald die DOM-Struktur[2] der Webseite gelesen wurde. Der DOM ist fertig aufgebaut, doch referenzierte CSS Stylesheets oder Bilder wurden noch nicht geladen. Ab diesem Zeitpunkt können Scripts mit Befehlen wie getElementsByIDs() oder getElementByName() auf den DOM zugreifen, da aller HTML Code geladen wurde [22][23].

Von Firefox und Opera wird dieser Event nativ unterstützt, jedoch noch nicht vom Internet Explorer und von Safari [24]. Um die Funktionalität in diesen Browsern ebenfalls nutzen zu können, gibt es diverse JavaScript Frameworks. Die Webseite ist jedoch vor dem DOM-bereit bereits rudimentär benutzbar. Es fehlen auf jeden Fall noch einige Bilder, da erst beim OnLoad alle geladen sind.

OnLoad-Event

Der JavaScript OnLoad-Event [23] findet statt, nachdem alle eingebetteten Objekte einer Webseite geladen wurden. Dazu gehören auch alle externen Ressourcen wie Bilder und CSS Stylesheets. Mit dem dort registrierten Code können nach dem Laden aller Objekte der Seite noch Änderungen vorgenommen oder weitere Elemente nachgeladen werden. JavaScript Code kann jedoch schon vor diesem Event ausgeführt werden.

Gemäss Steve Souders [25] ist der OnLoad-Event für Web 1.0-Webseite (statisch) ein guter Indikator dafür, dass die Webseite fertig geladen wurde. Für Web 2.0-Webseiten (dynamisch) hingegen reiche dies nicht aus, da diese intensiv mit JavaScripts arbeiten, welche nach dem OnLoad-Event heruntergeladen und ausgeführt werden. Da jedoch bereits vor dem OnLoad-Event JavaScripts ausgeführt werden, kann dies nicht so pauschal gesagt werden. Es kommt stark auf den Aufbau der Webseite an, welche Operationen noch nach dem OnLoad ausgeführt werden. Es wird versucht dem Benutzer möglichst früh eine benutzerbare Webseite anzuzeigen, weshalb gegen Schluss geladene Elemente nicht direkt zum wichtigen Inhalt der Webseite gehören dürften, wie beispielsweise Werbung.

Von Vorteil ist, dass der Zeitpunkt dieses Events von vielen Browser-Plugins angezeigt wird. Oft ist die Webseite jedoch schon weit vorher benutzbar.

[2] Beim DOM handelt es sich um eine standardisierte Schnittstelle um auf HTML Dokumente zuzugreifen und diese zu manipulieren. Dies geschieht of mittels JavaScript. Der DOM repräsentiert ein HTML Dokument als Baumstruktur, wobei die HTML Tags die Baumelemente darstellen.

HTTP-Load

Zu diesem Zeitpunkt wurde die letzte HTTP-Anfrage erfolgreich abgeschlossen. Falls nach dem On-Load-Event durch JavaScripts noch weitere Daten angefordert werden, so kommt der HTTP-Load nach dem OnLoad-Event. Ansonsten sollten die beiden Events in etwa gleichzeitig stattfinden. Die Webseite wurde komplett geladen.

Ablauf der Events

In Abbildung 11 ist anhand einer Zeitmessung beim Aufruf der Webseite 20min.ch unter Firefox auf Windows 7 der grobe Ablauf der Events dargestellt. In Abbildung 12 ist das Verhältnis der Zeiten besser sichtbar. Vom „Ersten Rendering“ bis zur „Ersten Anzeige“ vergeht nur wenig Zeit, verglichen mit der Differenz zwischen „Erste Anzeige“ und „DOM-bereit“ bzw. „OnLoad-Event“. Hier fallen die Events „Erste Anzeige“ und „Seite benutzbar“ sogar zusammen.

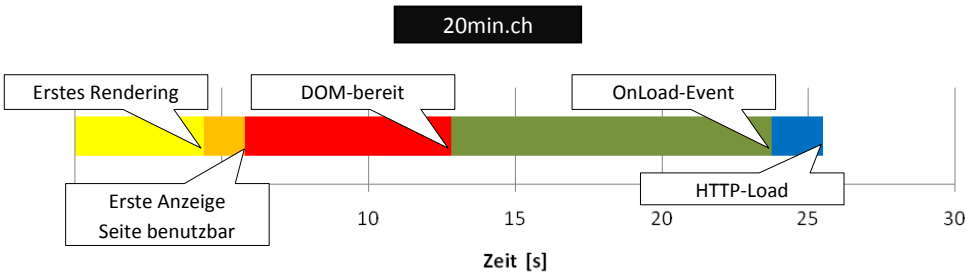


Abbildung 11: Ablauf der Events beim Webseitenaufruf, 20min.ch

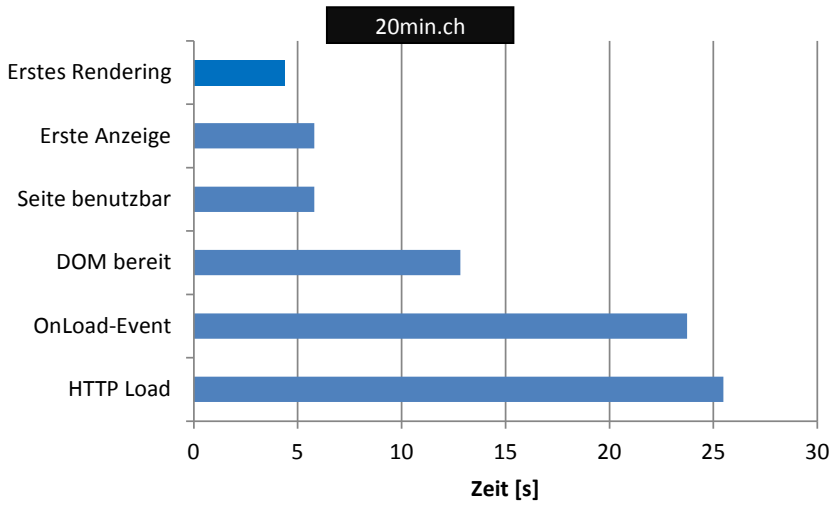


Abbildung 12: Verhältnis der Events beim Webseitenaufruf, 20min.ch

Der Aufbau der Webseite beeinflusst diese Verhältnisse jedoch stark, wie die zwei folgenden analogen Grafiken vom Aufruf der sbb.ch Webseite zeigen. Zudem hat der Browser je nach Webseite auch einen grossen Einfluss auf deren Darstellung (siehe Kapitel 7.3.4). Bei sbb.ch erscheint z.B. der „Seite benutzbar“-Event erst zum Schluss.

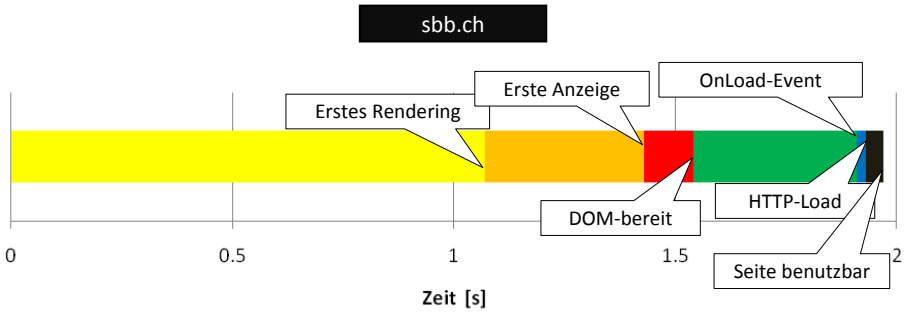


Abbildung 13: Ablauf der Events beim Webseitenaufruf, sbb.ch

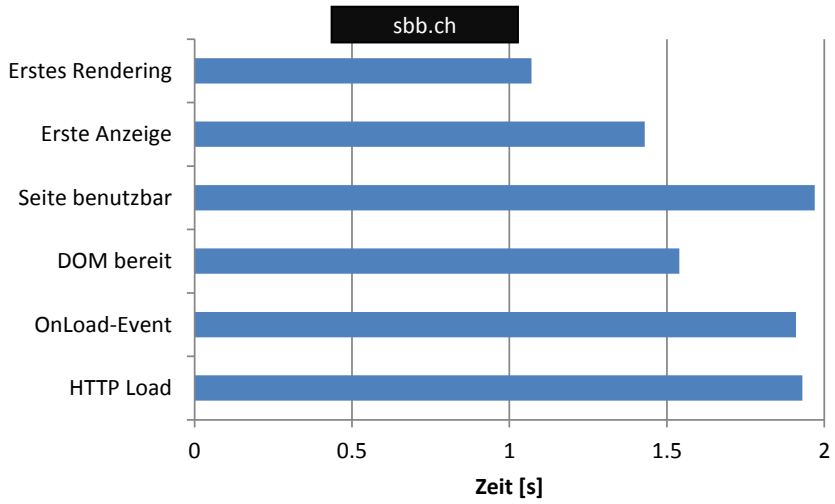


Abbildung 14: Verhältnis der Events beim Webseitenaufruf, sbb.ch

Zusammenfassung

Die EUE kann nicht über einen einzigen Wert abgebildet bzw. gemessen werden. Es gibt einige messbare Werte, welche alle nur bedingt mit der EUE zu tun haben.

Der „Erstes Rendering“-Event ist ein guter Hinweis, allerdings zu wenig aussagekräftig, da nicht klar ist, was dort genau bereits vom Browser dargestellt wird. Um dies erfassen zu können, muss der Ladevorgang manuell überwacht und die Zeit für die „Erste Anzeige“ und „Seite benutzbar“ von Hand gestoppt werden.

Der grösste Teil der Webseite ist bereits vor dem DOM-bereit und dem OnLoad-Event geladen. Es kann also meistens bereits mit der Seite interagiert werden, da auch schon ein Teil der Bilder beim DOM-bereit angezeigt wird. Da dieser jedoch nicht in allen Browsern einfach messbar ist, sollte auf den OnLoad-Event ausgewichen werden.

Der Wert „Seite benutzbar“ kommt der EUE am nächsten, kann allerdings nur grob und manuell gemessen werden. Zudem hängt er stark vom Aufbau der Webseite selber ab bzw. was ein Benutzer als benutzbar bezeichnet. Der OnLoad-Event wird zusätzlich aufgezeichnet, da dann die Webseite komplett geladen wurde und für Benutzer, welche die ganze Seite anschauen möchten, von Bedeutung ist.

Eine Übersicht mit welchen Tools welche Events aufgezeichnet werden können ist im Anhang im Kapitel 9.7 zu finden. Der genaue Ablauf der Messungen ist im Kapitel 0 und die effektiv eingesetzte Software in 9.8.2 dargestellt.

2.6. Verschiedene Messmethoden

Im Blog [26] der Firma dynaTrace, welche unter anderem im Bereich Web Performance tätig ist, werden einige Messmethoden angesprochen, welche hier kurz aufgelistet und kommentiert werden sollen.

Synthetische Transaktionen

Bei diesem Konzept werden Benutzer simuliert und spezifische nur lesende Transaktionen von verschiedenen Clients automatisiert aufgerufen. Werden wirkliche Browser simuliert, dann ist die Aussage relevanter, als wenn nur spezielle HTTP Anfragen simuliert werden. Der Schwerpunkt der Auswertung liegt hier auf der Serverseite und es geht primär darum Änderungen im Antwortverhalten des Servers sowie Netzwerkprobleme frühzeitig zu erkennen.

In dieser Arbeit sind jedoch der Browser und das Rendering ein wichtiger Teil der EUE und darum ist dieser Ansatz nur bedingt nützlich.

Netzwerk Sniffer

Hierbei wird ein Netzwerk Sniffer zwischen die Clients und die Webserver geschaltet, welcher den ganzen Datenverkehr aufzeichnet. Im Gegensatz zu den synthetischen Transaktionen wird realer Datenverkehr von echten Benutzern aufgezeichnet.

Der Sniffer kann auch auf dem PC des Surfers laufen, wäre also sehr nahe beim Endbenutzer. Jedoch kann browserspezifisches Verhalten wie JavaScript-Ausführung oder Browser-Events nicht aufgezeichnet bzw. erfasst werden. Es können jedoch TCP Verbindungsdetails analysiert und auch DNS-Abfragen erfasst werden.

Code Injection in die Webseite

Es wird zusätzlicher JavaScript Code in die Webseite eingebaut, welcher direkt vom Browser ausgeführt wird und deshalb auch browserspezifisches Verhalten aufzeichnen kann. So kann z.B. kurzer Code in Kopf- und Fussteil der Webseite eingefügt werden, welcher Zeit-Informationen des Browsers sammelt wie den Zeitpunkt des OnLoad-Events. Auch die von Steve Souders entworfene Methode „Episodes“ [25] fällt in diesen Bereich. Um diese Variante nutzen zu können, muss jedoch die Webseite angepasst werden, was für diese Arbeit nicht in Frage kommt.

Ein weiterer Nachteil ist, dass für jede zu ladende Ressource, bei der das Timing genauer untersucht werden soll, zusätzlicher Code eingefügt werden muss. Weiterhin liefert diese Methode keine Informationen zum Rendering, da dies nicht per JavaScript abfragbar ist.

Browser Plugins

Für den Browser ist ein Plugin zu installieren, mit welchem der Benutzer sehr detaillierte Informationen zum Seitenaufbau erhält. Die Informationen werden direkt beim Browser gewonnen und eignen sich somit auch für den Vergleich von verschiedenen Browsern, wenn dann auch mehrere Browser unterstützt werden. Nur damit kann man Informationen über das Renderingverhalten und die JavaScript-Ausführung erhalten.

Problematisch ist hier, dass es kein Plugin gibt, das in allen Browsern funktioniert.

Browser selber

Aktuell liefern die Browser selber keine Detailinformationen über den Ladevorgang. Diese sind nur via Plugins verfügbar. Es gibt aber Bestrebungen dies zu ändern, so dass diese Daten über eine einheitliche API abgefragt werden können [27]. So könnte auch ein Tool entwickelt werden, das mit allen Browsern funktioniert.

Stoppuhr-Tools

Bei diesem Typ Tool läuft lokal ein JavaScript, das die Zeit misst zw. Ladebeginn der Webseite und wenn der Browser meldet, dass die Seite fertig geladen wurde („Fertig“ wird in der Statusleiste angezeigt). Dieser Zeitpunkt entspricht also in etwa dem OnLoad-Event.

Die Erfassung dieses Events ist mit anderen Tools jedoch um einiges einfacher und es dabei auch noch weitere Daten aufgezeichnet.

Manuelle Stoppuhr

Es wird die Zeit gemessen, welche vom Drücken des Enter-Buttons nach Eingabe der URL bis zur Darstellung der Webseite im Browser vergeht. Diese Messmethode ist natürlich nicht so genau, wie beim Einsatz von Tools, da von Hand gemessen wird. Für die Events „Erste Anzeige“ sowie „Seite benutzbar“ ist dies die einzige Methode ohne allzu grossen Aufwand.

Die Genauigkeit könnte durch Filmen des Bildschirminhalts und Auswerten des Videos verfeinert werden, doch wächst damit der Aufwand stark an.

Zusammenfassung

Es gibt einige Methoden, um die Geschwindigkeit des Seitendownloads und deren Anzeige festzuhalten. Jedoch nur wenige davon beziehen dabei den Browser mit ein und zeigen auch relevante Werte bzgl. der EUE.

Die Wahl fällt auf die Browser-Plugins, da nur dort das browserspezifische Verhalten (Rendering und JavaScript-Ausführung) miterfasst werden kann. Die Events „Erste Anzeige“ sowie „Seite benutzbar“ können so jedoch nicht gemessen werden und müssen von Hand gestoppt werden. Für eine Detailanalyse der TCP-Verbindungen kann bei Bedarf ein Netzwerk Sniffer eingesetzt werden.

Ein Vergleich der möglichen Tools ist im Kapitel 9.7 zu finden und der genaue Ablauf der Messungen wird im Kapitel 0 beschrieben.

3. Transmission Control Protocol (TCP)

Der Browser verwendet HTTP (Hypertext Transfer Protocol), um Webseiten via Internet zu laden. HTTP basiert wiederum auf dem zuverlässigen Transportlayerprotokoll TCP (Transmission Control Protocol), da Paketverluste dadurch behandelt werden und sich nicht der Browser damit beschäftigen muss.

In diesem Kapitel werden die wichtigsten Eigenschaften von TCP analysiert, welche für die Analyse und Interpretation der Messergebnisse relevant sind. Weitere grundsätzliche Eigenschaften sind im deutsch- [28] sowie englischsprachigen [29] Wikipedia Artikel sehr gut beschrieben.

3.1. Verbindungsaufbau

Bevor mit TCP Daten gesendet werden können, muss zuerst eine TCP-Verbindung aufgebaut werden, da es sich um ein verbindungsorientiertes Protokoll handelt. Zum Aufbau der Verbindung wird der sogenannte Three-Way-Handshake (Drei-Wege-Handschlag) verwendet.

In der untenstehenden Grafik ist der Ablauf schematisch aufgezeichnet. Es vergeht eine RTT (Round Trip Time, Paketumlaufzeit) bis der Client die ersten Daten - seine Anfrage für die Webseite - schicken kann. Die RTT bezeichnet die Zeit, welche zwischen dem Absenden der Anfrage und dem Empfangen der Antwort verstreicht.

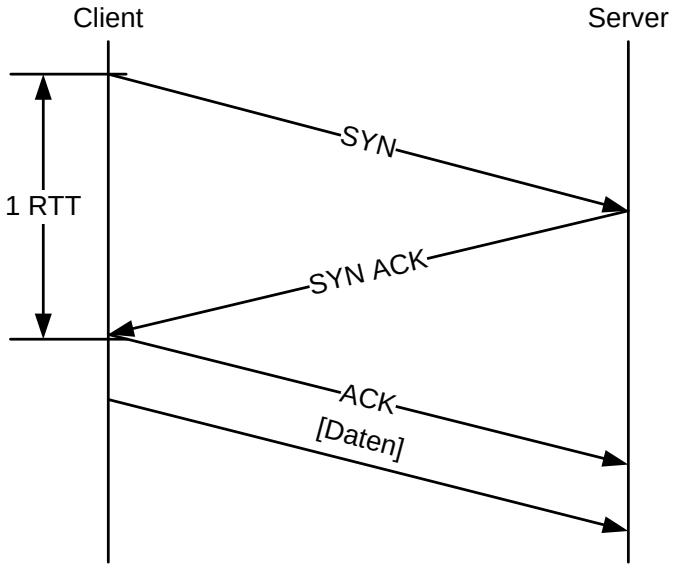


Abbildung 15: TCP Handshake, schematischer Ablauf

Zusätzliche Verzögerung für die EUE zeigt sich also direkt in der benötigten Zeit für den Aufbau einer TCP Verbindung. Da pro Webseitenaufruf mehrere Verbindungen geöffnet werden, kommt der Wert mehrfach zum Tragen. Im Kapitel 5.1 wird genauer auf die Anzahl der Verbindungen der verschiedenen Browser eingegangen.

3.2. Maximum Segment Size

Beim Verbindungsaufbau teilen die Hosts einander zudem ihre MSS (Maximum Segment Size) mit (siehe Abbildung 16). Dies ist die Anzahl Bytes, welche ein Host als Payload (Nutzlast) in ein TCP-Segment packen kann, ohne dass das IP-Paket einen Layer tiefer unten für den Versand fragmentiert werden muss. Der Austausch ist nötig, damit sie wissen, welche MSS für Segmente an den anderen Host zu verwenden ist. Dieser Wert ist von der MTU (Maximum Transmission Unit) - der L2 Payload - des verwendete Layer 2 Protokolls abhängig, wie in der folgenden Grafik gut zu erkennen ist.

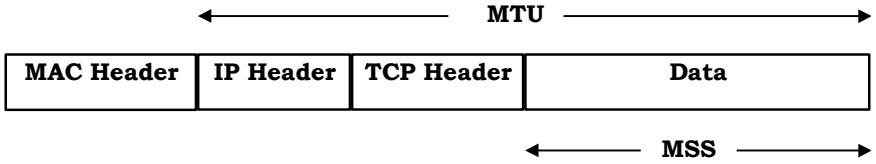


Abbildung 16: Zusammenhang MTU und MSS [30]

IP-Pakete können bis 64 kB gross werden, müssen jedoch in Layer 2 Frames verpackt werden. Bei Ethernet ist die maximale Payload eines Frames auf 1500 Bytes festgelegt, wobei von den 1500 Bytes je 20 Bytes für den IP Header sowie 20 Bytes für den TCP Header wegfallen. Es bleiben also noch 1460 Bytes als Payload für TCP, was der MSS entspricht. Bei DSL muss man davon aufgrund des PPP-Headers noch 8 Bytes abziehen.

Früher wurde für die Bestimmung der MTU nur das lokale Netz betrachtet. Dieser Wert lässt sich direkt vom Netzwerkkartentreiber auslesen. Wichtig ist jedoch die minimale Ende-zu-Ende-MTU. Denn nur wenn IP-Pakete nicht grösser sind als diese, werden sie während der ganzen Übermittlung nicht fragmentiert. Um die MTU des entsprechenden Pfades durch das Internet zu bestimmen, gibt es die sogenannte Path MTU Discovery (Bestimmung der MTU für einen bestimmten Pfad, siehe Kapitel 3.2).

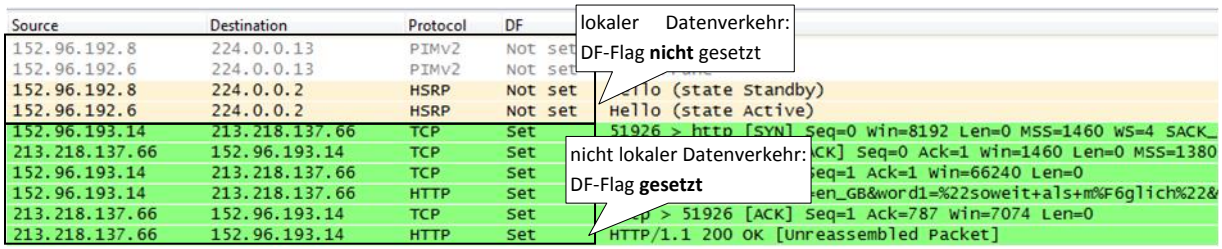
Jeder Host bestimmt für sich die MSS und übermittelt sie dem anderen im SYN Paket beim Verbindungsaufbau. Der kleinere der beiden Werte wird für anschliessend für die Kommunikation verwendet [31]. Wird diese Grösse eingehalten, so sollten im Wireshark keine fragmentierten IP-Pakete zu sehen sein. Wenn vom anderen Host die MSS Option nicht empfangen wird, so wird der Standardwert von 536 Bytes gewählt.

Nun enthält jedoch nicht jedes TCP-Paket 1460 Bytes bei Ethernet als Layer 2-Technologie, da dies nur den maximalen Wert darstellt. Der wirklich verwendete Wert hängt von der Receive Window Grösse (RWin) des Empfängers ab [32] (siehe Kapitel 3.4).

3.3. Path MTU Discovery

Die Path MTU Discovery dient der Bestimmung der minimalen MTU für einen bestimmten Pfad im Internet. Damit soll die Fragmentierung von IP-Paketen verhindert werden. Denn die Fragmentierung und Reassemblierung kostet Rechenzeit und erhöht dadurch die RTT. Weiterhin bedeutet es mehr Kommunikationsoverhead (mehr Header) und einige Firewalls verwerfen fragmentierte Pakete aus Sicherheitsgründen.

Die Bestimmung der Pfad-MTU funktioniert wie folgt: Sobald Pakete für nicht lokale Netzwerke bestimmt sind, wird das DF-Flag (Don't Fragment) im IP-Header gesetzt (siehe Abbildung 17)



Source	Destination	Protocol	DF	
152.96.192.8	224.0.0.13	PIMv2	Not set	
152.96.192.6	224.0.0.13	PIMv2	Not set	
152.96.192.8	224.0.0.2	HSRP	Not set	Hello (state Standby)
152.96.192.6	224.0.0.2	HSRP	Not set	Hello (state Active)
152.96.193.14	213.218.137.66	TCP	Set	51926 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_
213.218.137.66	152.96.193.14	TCP	Set	CK] Seq=0 Ack=1 Win=1460 Len=0 MSS=1380
152.96.193.14	213.218.137.66	TCP	Set	eq=1 Ack=1 Win=66240 Len=0
152.96.193.14	213.218.137.66	HTTP	Set	en_GB&word1=%22soweit+als+m%F6glich%22&
213.218.137.66	152.96.193.14	TCP	Set	> 51926 [ACK] Seq=1 Ack=787 Win=7074 Len=0
213.218.137.66	152.96.193.14	HTTP	Set	HTTP/1.1 200 OK [Unreassembled Packet]

Abbildung 17: DF-Flag bei versch. Datenverkehr

Falls das Paket von einem Router auf dem Pfad für die Weiterleitung fragmentiert werden müsste, so verwirft er das Paket und sollte anschliessend eine ICMP-Fehlermeldung an den Absender schicken (Destination Unreachable - Fragmentation Needed, DF Set). Der Sender verringert dann seine MTU solange, bis die Daten beim Empfänger ankommen ohne eine ICMP Fehlermeldung zu erhalten. Eine so bestimmte MTU ist natürlich nur für einen bestimmten Pfad gültig. Verkleinert sich die MTU aufgrund eines anderen Pfades durch das Internet, so wird der Absender wieder durch eine ICMP-Fehlermeldung benachrichtigt. Ist auf dem neuen Pfad jedoch eine grössere MTU als zuvor möglich, so merkt dies der Client nicht automatisch. Er muss dazu die Path MTU Discovery erneut durchführen. Damit diese Untersuchungen jedoch das Netzwerk nicht zu stark belasten, darf dieser Test erst fünf Minuten nach der letzten MTU-Verringerung stattfinden [33].

3.4. Receive Window

Beim Verbindungsaufbau tauschen die Hosts die Grösse ihres RWin (Receive Window) aus. Dieser Wert wird in jedem TCP-Segment mitgeschickt, auch wenn die Verbindung bereits steht (siehe Abbildung 18). Beim RWin handelt es sich um die maximale Anzahl Bytes, welche der Absender des Wertes aktuell empfangen möchte und ist somit eine Angabe zum freien Platz im Empfangsbuffer des Hosts. Das RWin steht aber auch für die maximale Anzahl unbestätigter Bytes, welche ein Sender versenden kann. ohne ein ACK zu erhalten.

Beim RWin handelt sich um den Teil der TCP-Flusskontrolle, welchen der Empfänger beeinflussen kann. Für die Beeinflussung des Senders, siehe Kapitel 3.7.

Mit Hilfe der Window Scaling Option ist ein maximales Window von 1 GB möglich. Ohne die Scaling Option beträgt das Maximum nur 64 kB. Die Scaling Option kann nur beim Verbindungsaufbau aktiviert resp. gesetzt werden. Anschliessend lässt sich das Scaling für die entsprechende TCP-Verbindung nicht mehr ändern.

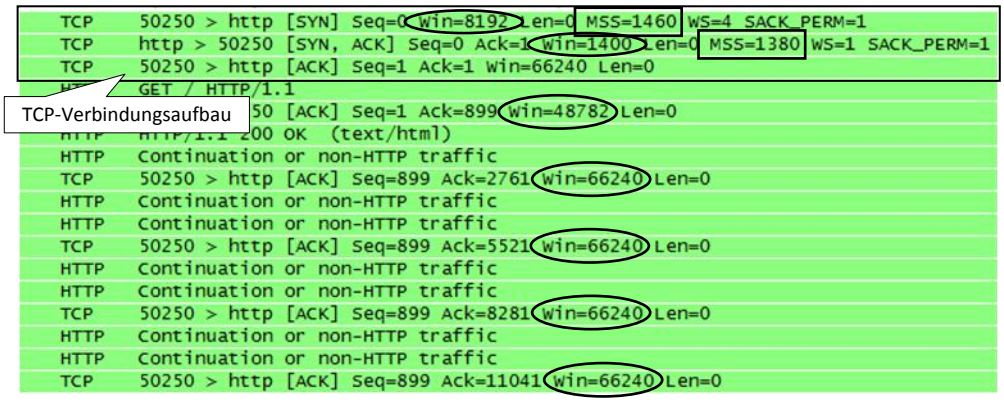


Abbildung 18: TCP-Verbindungsaufbau und weitere Datenübertragung

Will oder kann ein Host für eine gewisse Zeit keine Daten mehr empfangen, da sein Empfangsbuffer voll ist, so setzt er seine RWin auf null. Er teilt seine RWin bei jedem ACK seinem Gegenüber mit, kann aber auch ein separates Paket schicken, nur um ihm seine neue RWin mitzuteilen. Dann spricht man von einem Window Update.

3.5. RTT-Messung und RTO

Wenn TCP nach einer bestimmten Zeit kein ACK für ein versandtes Segment erhält, so wird dieses erneut geschickt. Dieser Timeout – in der Fachsprache RTO (Retransmission Timeout) genannt – ist zu Beginn auf 2.5s bis 3s gesetzt, wird jedoch anschliessend von TCP überwacht und in Abhängigkeit von der RTT laufend neu berechnet.

Dabei wird pro RWin mindestens eine RTT-Messung anhand der Zeitabstände zwischen versendetem Segment und empfangenem ACK vorgenommen [34]. Erneut gesendete Segmente dürfen für die Messung nicht verwendet werden, da unklar ist, ob das ACK auf das erste oder das erneut gesendete Segment kam (Karn's Algorithmus). Wird ein sehr grosses cwnd (Congestion Window, siehe Kapitel 3.7) verwendet, so sollten mehrere Messungen pro Window vorgenommen werden, um eine einigermassen genaue Schätzung zu erhalten. Ansonsten ist die „Abtastrate" der RTT zu tief ist [34][35]. Mit diesen Messungen wird die RTT geschätzt und daraus der RTO berechnet, wobei über die Varianz der RTT ebenfalls Buch geführt wird und diese mit einfliesst. Die Werte werden jeweils geglättet.

Eine weitere Variante der RTT-Messung, welche auch bei grossem cwnd eine gute Schätzung erlaubt, ist die Verwendung der TCP Timestamps (Zeitstempel), welche Bestandteil der TCP-Optionen sind. Um diese einsetzen zu können, muss beim Verbindungsaufbau diese Option mit gesendet werden [36]. Damit könnte bei jedem ACK die RTT gemessen werden [35].

Es existiert ein Minimalwert für den RTO von 1s [28][37], was vor allem mit dem Delayed ACK (verzögertes ACK) zu tun hat. Der Timeout dort beträgt gemäss RFC 1122 [38] bis zu 500ms.

3.6. Bandbreite-Delay-Produkt

Die RTT und die RWin-Grösse beeinflussen die verwendbare Bandbreite des Senders. Wenn das Window kleiner ist als das Bandbreite-Delay-Produkt (BDP, siehe Formel 1), so wird der Datendurchsatz limitiert, da lange auf ACKs gewartet werden muss. Während einer RTT können maximal nur so viele Bytes unterwegs sein, wie das RWin gross ist. Denn so lange dauert es, bis die Daten vom Sender zum Empfänger gehen und das ACK zurückkommt.

$$erreichte\ Bandbreite \leq \frac{RWin}{RTT}$$

$$RWin \geq erreichte\ Bandbreite * RTT$$

Formel 1: Bandbreite-Delay-Produkt

So kann bei einer RTT von 200ms und einem Window von 65‘535 Bytes nur eine maximale Bandbreite von 2.62 Mbit/s genutzt werden (65‘535 Bytes * 8 Bit pro Byte / 0.2s = 2.62 Mbit/s). Wird nun die RTT um 50ms erhöht, so sind nur noch 2.1 Mbit/s nutzbar. So spiegelt sich eine Erhöhung der RTT bei fixer Window-Grösse umgekehrt proportional in der nutzbaren Bandbreite wieder.

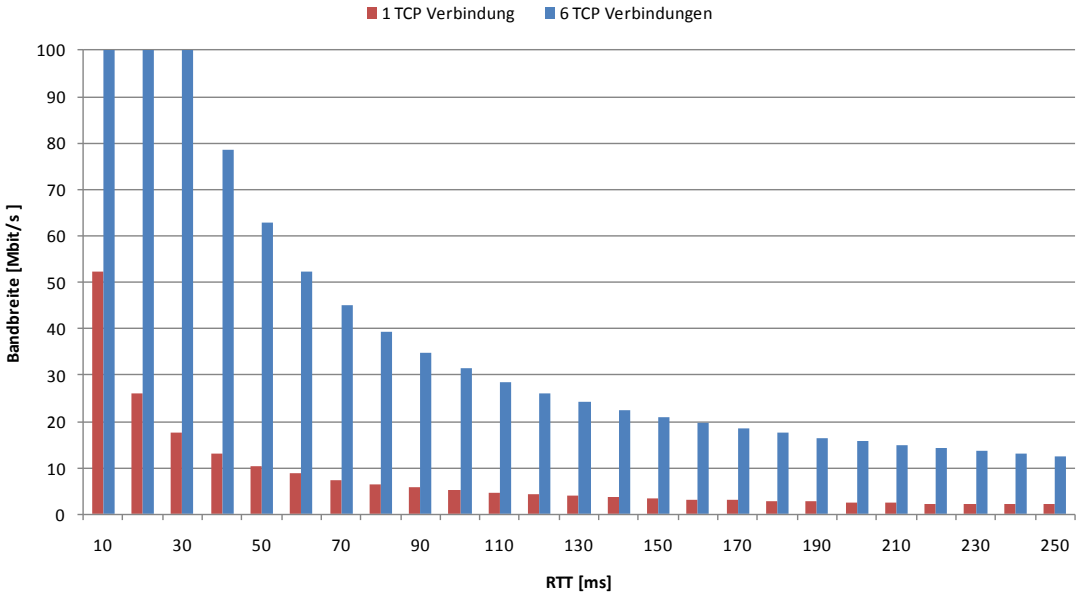


Abbildung 19: Bandbreite in Abhängigkeit der RTT, RWin 64 kB

Nimmt man eine RTT von 30ms, was in Europa schnell einmal erreicht ist, so ist er Datendurchsatz bei einem RWin von 64 kB auf ca. 17.5 Mbit/s pro TCP-Verbindung begrenzt. Bei den heutigen Internet-Abos von 20 Mbit/s und mehr bekommt diese Beschränkung immer mehr an Bedeutung.

Aus diesem Grund unterstützen einige Applikationen wie Browser oder Downloadmanager mehrere parallele TCP-Verbindungen, um auch bei begrenztem RWin noch die volle Bandbreite ausnutzen zu können. Im Kapitel 5.1 gibt es weitere Details bzgl. paralleler Verbindungen bei den Browsern.

Doch auch trotz Einsatzes von bis zu 6 parallelen TCP-Verbindungen, wie bei den aktuellsten Browsern möglich ist, wird bei einem Download mit einem RWin von 64kb und einer RTT von 110ms (Ostküste USA) nicht mehr als 29 Mbit/s erreicht (siehe Abbildung 19). Das Window Scaling wird immer wichtiger, denn ein grosser Anteil an der RTT wird durch die Signalgeschwindigkeit im Kabel festgelegt, sprich durch die unveränderliche Lichtgeschwindigkeit.

Die Hosts sollten also bei der Festlegung der RWin-Grösse die RTT bzw. das BDP berücksichtigen. Die RWin-Grösse des Servers ist dabei im Gegensatz zu jener des Clients kaum relevant, da die meisten Daten vom Server zum Client hin fliessen.

Primär wird die Window-Grösse jedoch aufgrund des freien Platzes im Empfangsbuffer des Hosts gewählt. Dies spielt vor allem eine Rolle für die Wahl des aktuellen RWin-Wertes, da eine optimale RWin bzgl. des BDP keinen Vorteil bringt, wenn die Applikation die Daten nicht schnell genug aus ihrem Empfangsbuffer abholen kann. Mit dem „Next Generation TCP/IP Stack“ wurde dieses Verhalten angepasst (siehe Kapitel 4.2.1).

3.7. TCP Slow Start

Zweck des Slow Starts ist, dass der Sender nicht zu viele Daten auf einmal schickt und dadurch das Netzwerk überlastet. Er soll sich an die optimale Senderate herantasten. Deshalb wurde das cwnd (Congestion Window) eingeführt. Der kleinere Wert von cwnd und RWin bestimmt, wie viel Bytes der Host schicken darf.

Beim cwnd handelt sich um den Teil der TCP-Flusskontrolle, welchen der Sender beeinflussen kann. Für die Beeinflussung des Empfängers, siehe Kapitel 3.4.

Normales ACK

Da im heutigen Internet sehr viele kurzlebige Datenflüsse anzutreffen sind [45], welche den grössten Teil ihrer Lebenszeit im Slow Start verbringen, ist es wichtig diese Phase genauer anzuschauen. Bei der Einführung des Slow Starts wurde das cwnd bei Verbindungsbeginn auf 1 MSS gesetzt, also auf den Wert welcher beim Verbindungsaufbau ausgehandelt wird. Pro empfangenes ACK wird das cwnd um eine weitere MSS erhöht, wobei die ACKs des Verbindungsaufbaus nicht dazu zählen (RFC 3390). Durch den Slow Start ist gewährleistet, dass die Sendegeschwindigkeit des Hosts Stück für Stück erhöht wird.

Zu Beginn des Slow Starts wird die Variable ssthresh (Slow Start Threshold, Slow Start Schwellwert) gesetzt. Anhand dieses Wertes wird entschieden, ob sich TCP in der Slow Start (cwnd < ssthresh) oder in der Congestion Avoidance Phase (cwnd ≥ ssthresh, siehe Kapitel 0) befindet. Aus diesem Grund kann der anfängliche Wert beliebig hoch sein und wird meistens auf die maximale RWin-Grösse gesetzt, damit der Slow Start nicht vorzeitig abgebrochen wird [39, S. 57].

Beim normalen ACK-Verhalten, wo jedes Paket direkt bestätigt wird, erhöht sich das cwnd stark exponentiell, wie in Abbildung 20 zu sehen ist. Pro RTT wird das cwnd verdoppelt.

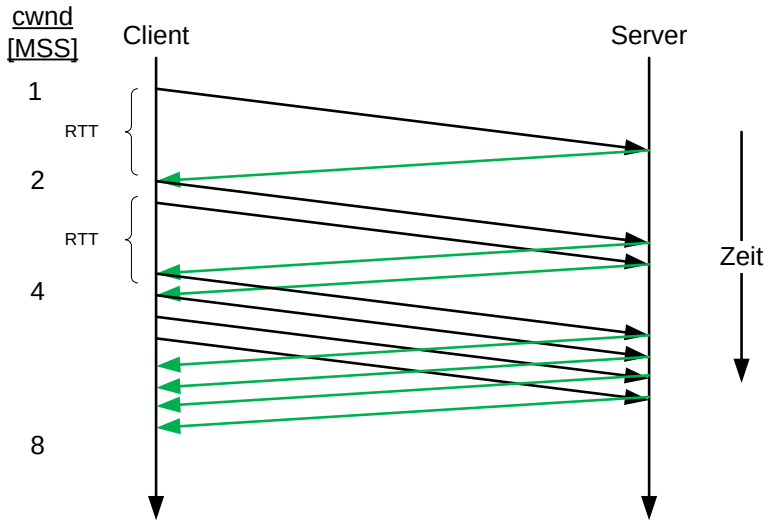


Abbildung 20: TCP Slow Start mit direktem ACK, Zeit für Paketverarbeitung nicht berücksichtigt

Delayed ACK

Mit dem RFC 1122 wurde das Delayed ACK (verzögertes ACK) eingeführt um die Netzwerklast und den Overhead (Mehraufwand) zu verringern. Es wird nicht jedes Segment sofort bestätigt, sondern bis zu maximal 500ms (typischerweise 100-200ms) gewartet. Ziel ist es das ACK gerade mit der Antwort mitzuschicken (Huckepack/piggybacked) bzw. für mehrere Segmente aufs Mal. Gemäss RFC sollte jedoch im Minimum für jedes zweite Segment ein ACK verschickt werden [38][39, S. 46][40].

In der Realität wächst das cwnd also nicht so stark exponentiell, sondern langsamer [39, S. 54]. TCP muss Delayed ACK nicht implementieren, es handelt sich lediglich um eine Empfehlung im RFC 1122 [38]. Wobei heutzutage vermutlich die meisten TCP-Stacks dieses Feature implementiert und aktiviert haben. In Abbildung 21 ist ein Vergleich des cwnd-Wachstumsverhalten der verschiedenen ACK-Varianten im Slow Start dargestellt. Es ist klar ersichtlich, dass das neue Verfahren das Netzwerk wesentlich weniger stark belastet in der Startphase. Es ist ein feineres Herantasten an das Netzwerklimit möglich.

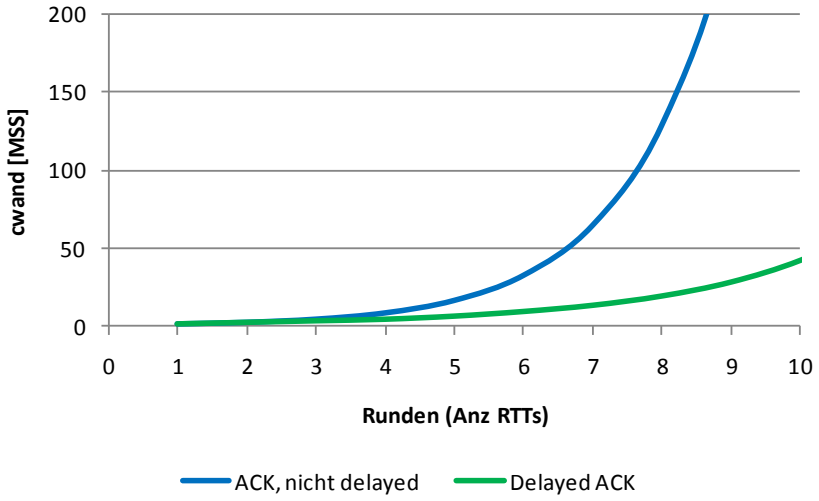


Abbildung 21: Vergleich cwnd Wachstumsverhalten im Slow Start bei versch. ACK-Varianten

Bei Einsatz des Delayed ACK können zwei Hauptfälle unterschieden werden:

(1) $RTT \leq T_{DelayedACK}$
(2) $RTT > T_{DelayedACK}$

Ist die $RTT \leq T_{DelayedACK}$, so kommt es abgesehen vom Anfang nie vor, dass das Timeout des Delayed ACK abläuft und ein ACK für ein einzelnes Segment verschickt wird (siehe Abbildung 22). Die Segmente kommen schnell genug an, dass immer ein ACK für zwei Segmente versendet werden kann.
Dieser Fall entspricht heute den meisten Situationen, zumindest die meisten Internetbenutzer in der Schweiz nur selten eine RTT von über 200ms antreffen.

Für die folgende Modellierung wurde angenommen, dass keine Paketverluste auftreten, der Datenfluss nicht durch das RWin limitiert wird und keine Zeit für den Empfang/Verarbeitung/Antwort auf die Pakete benötigt wird.

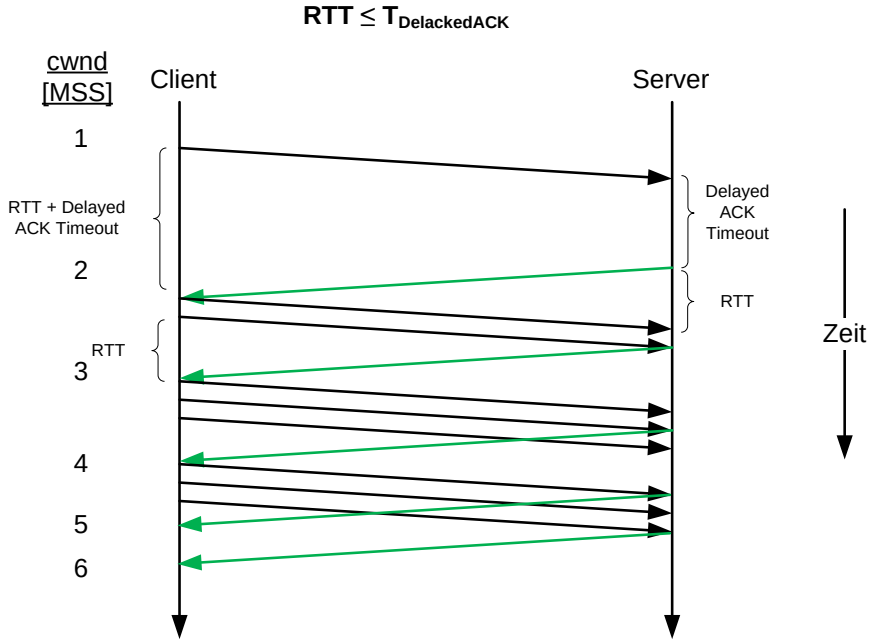


Abbildung 22: TCP Slow Start mit Delayed ACK wobei $RTT \leq T_{DelayedACK}$

Ist die $RTT > T_{DelayedACK}$, so kommt immer wieder mal vor, dass das Timeout des Delayed ACK abläuft und ein ACK für ein einzelnes Segment verschickt wird (siehe Abbildung 23). Dies ist vor allem bei Verbindungen mit einem hohen Bandbreiten-Delay-Produkt der Fall.

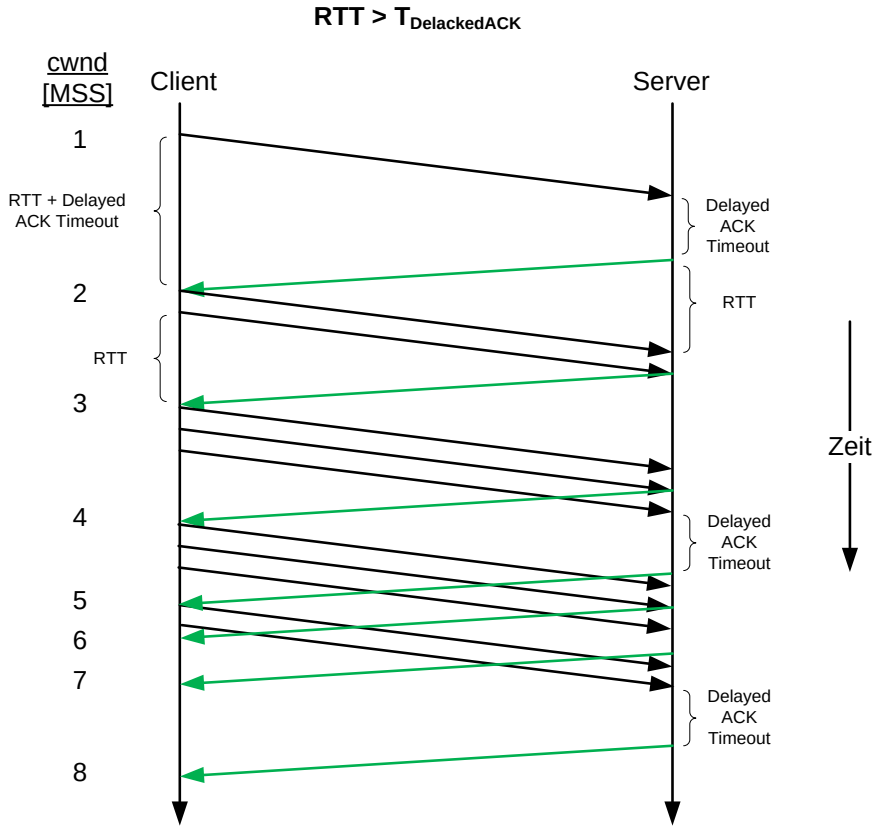


Abbildung 23: TCP Slow Start mit Delayed ACK wobei RTT > $T_{DelayedACK}$

Höheres cwnd beim Start

Beim ersten Segment handelt es sich um einen Spezialfall, der bei beiden Fällen auftritt. Denn der Client darf vorerst aufgrund des cwnd nur ein Paket versenden und der Server wartet anschliessend auf ein weiteres Paket, das aber nie eintrifft. Erst nach dem typischen Timeout von 200ms geht es weiter.

Um dieses Timeout zu umgehen, wurde im RFC 5681 sowie in der Vorversion (RFC 3390) das anfängliche cwnd auf zwei bis vier MSS gesetzt, je nach grössere der MSS. Bei RFC 5681 handelt es sich jedoch noch um einen Draft Standard. Bei einer MSS von 1460 Bytes (Ethernet) wird das anfängliche cwnd auf drei festgelegt, wobei es sich dabei um eine Empfehlung handelt. Ein kleinerer Wert ist weiterhin erlaubt. Es wird aktuell diskutiert, das anfängliche cwnd noch weiter auf bis zu 10 MSS zu erhöhen [50]. Denn während die Netzwerke über die letzten 10 Jahre immer schneller geworden sind, ist das anfängliche cwnd gleich geblieben.

Von einem grösseren anfänglichen cwnd profitieren vor allem kurzlebige Verbindungen, da so schneller mehr Daten verschickt werden können und die Übertragung somit schneller abgeschlossen ist. Dadurch wird die Interaktivität deutlich erhöht. Dies betrifft viele Web- und E-Mail-Übertragungen. Aber auch langlebigeren Verbindungen mit hohem BDP und hoher RTT profitieren davon, da einige RTTs gespart werden können (im aktuellen Falls bis zu 3 RTTs).

RTT beeinflusst Anstiegsverhalten

Interessant ist der Befund im Paper „The Impact of RTT and Delayed ACK Timeout Ratio on the Initial Slow Start Phase“ [45], dass das Anstiegsverhalten des cwnd je nach RTT leicht anders ausfällt. Dabei konnten vier verschiedene Fälle der RTT unterschieden werden, welche für diese Arbeit relevant sind.

Fall	Zeitintervall relativ	Zeitintervall absolut [ms]	Empfangene ACKs
a	$]0, T_{DelayedACK}]$	]0, 200]	immer für 2 Segmente (nie $T_{DelayedACK}$)
b	$]T_{DelayedACK}, 2\,T_{DelayedACK}[$	]200, 400]	ab und zu ACKs für einzelne Segmente sonst für 2 (gelegentlich $T_{DelayedACK}$)
c	$2\,T_{DelayedACK}$	400	
d	$]2\,T_{DelayedACK}, 3\,T_{DelayedACK}]$	]400,600]	

Tabelle 2: Definition von verschiedenen RTT-Intervallen, $T_{DelayedACK}$ = 200ms

Bei RTTs, die höher als das Delayed ACK Timeout ($T_{DelayedACK}$) von 200ms sind (Fälle b-d), ist gemäss Tabelle 3 ab Runde 4 das Anstiegsverhalten des cwnd schneller als bei a. Ein schnelleres Anstiegsverhalten hat zur Folge, dass ein Host schneller mehr Daten versenden kann. In diesem Zusammenhang wird der Einfluss der RTT auf die effektive Datenübertragung noch nicht beachtet.

Bei der Auswertung, welche in Tabelle 3 dargestellt ist, wurde mit einer MSS von 1460 Bytes gerechnet. Dies stellt die am häufigsten anzutreffende MSS im Internet dar, da viele Clients via Ethernet angeschlossen sind. Eine Runde beginnt, wenn der Sender das erste Segment eines Windows verschickt und endet, wenn er ein ACK (für eines oder mehrerer dieser Segmente) empfängt [45]. Eine Runde entspricht also einer RTT, wobei diese natürlich gemäss Tabelle 2 unterschiedlich sind.

Runde	Fall a	Fall b	Fall c	Fall d	Abweichung b von a	Abweichung c von a	Abweichung d von a
1	1.43	1.43	1.43	1.43	0%	0%	0%
2	4.28	4.28	4.28	4.28	0%	0%	0%
3	8.55	8.55	8.55	8.55	0%	0%	0%
4	12.83	15.68	15.68	15.68	22%	22%	22%
5	21.39	24.24	24.24	27.09	13%	13%	27%
6	34.22	37.07	39.92	44.20	8%	17%	29%
7	51.33	58.46	61.31	69.86	14%	19%	36%
8	76.99	88.40	95.53	108.36	15%	24%	41%
9	115.49	135.45	146.86	163.96	17%	27%	42%
10	175.37	203.89	223.85	249.51	16%	28%	42%
11	265.20	306.54	339.34	377.83	16%	28%	42%
12	397.79	460.53	510.43	570.31	16%	28%	43%
Schnitt					+ 15%	+ 23%	+ 36%

Tabelle 3: Vergleich der kumulierten Datenmenge [kB] bei gleicher Runde resp. RTT

Dieses Ergebnis ist unerwartet, da mit zunehmender RTT aufgrund des mehrfachen Auftretens des Timeouts ($T_{DelayedACK}$) mit einem langsameren Wachstum des cwnd und damit der kumulierten Datenmenge gerechnet werden könnte. Der Effekt kommt dadurch zustande, dass gelegentlich auch ACKs für einzelne Segmente versendet werden. Das heisst, dass es bei den Fällen b-d total mehr ACKs gibt als beim Fall a, wo konsequent immer ein ACK pro zwei Segmente verschickt wird.

Da sich die RTTs bei den einzelnen Fällen jedoch markant unterscheiden, ist ein direkter Vergleich ohne Einbezug der RTT nicht aussagekräftig. Die Wachstumskurve mit dem obersten Punkt (siehe Abbildung 24, rot eingekreist) ist die mit dem effektiven Verhalten der entsprechenden RTT. Die unterste Linie steht für das Verhalten vom Fall a, der für höhere RTTs zu Vergleichszwecken hochgerechnet wurde.

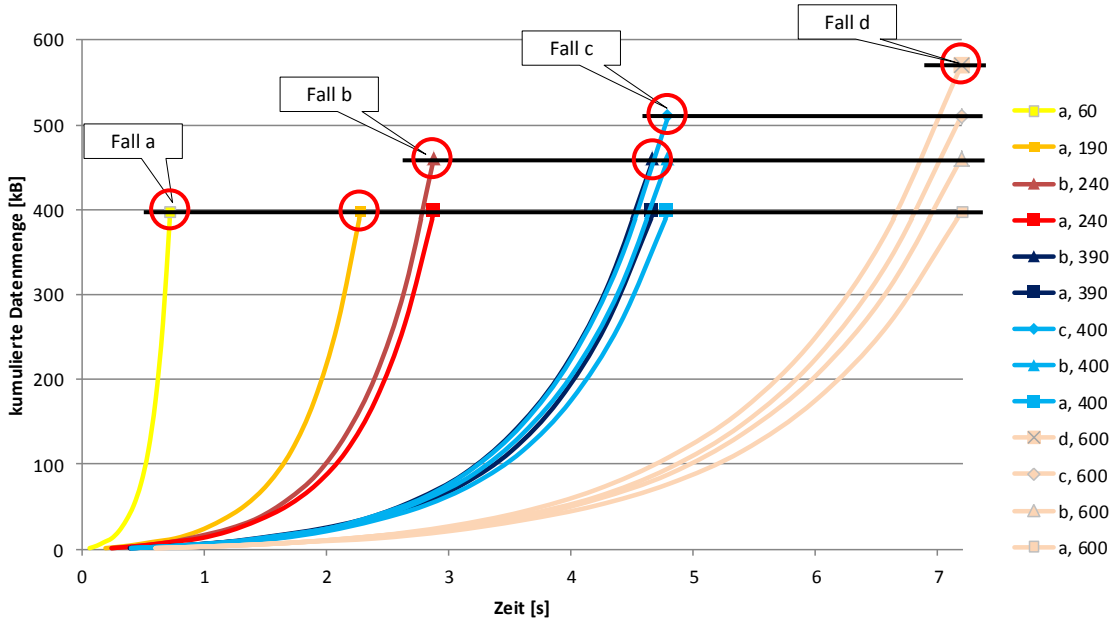


Abbildung 24: Vergleich der cwnd Anstiegsverhalten

In Tabelle 4 sieht man wie lange es dauert, bis eine bestimmte Datenmenge übertragen wurde, wenn die Verbindung frisch gestartet wurde. Die Übertragung von 60 kB ist bei einer MSS von 1460 Bytes in den folgenden markierten Runden abgeschlossen:

Runde	normales ACK	Delayed ACK (Fall a)
1	1.43	1.43
2	4.28	4.28
3	9.98	8.55
4	21.39	12.83
5	44.20	21.39
6	89.82	34.22
7	181.07	51.33
8	363.57	76.99
9	728.57	115.49
10	1'458.57	175.37

Tabelle 4: Vergleich TCP Slow Start und kumulierte Datenmenge [kB]

Dabei ist bis zum Ende der jeweils markierten Runde mehr Daten als die 60 kB übertragen worden. Für 69 kB bekommt man also die gleichen Runden-Werte wie für 60 kB. Dies bedeutet, dass es für Webentwickler kaum Sinn macht die erste Seite auf 52 kB zu optimieren, wenn die Übermittlung bei der Delayed ACK Variante bis zu 75 kB praktisch gleich lange dauert.

Damit der Benutzer möglichst schnell etwas von der Webseite im Browser sieht, sollte innerhalb der ersten RTTs der wichtigste Inhalt komplett geschickt werden können. Es sollte ebenso auf die Reihenfolge der Elemente geachtet werden, in welcher sie abgerufen werden. Die kleinen sollten zuerst an der Reihe sein, wenn das cwnd noch klein ist, und erst später, wenn das Window offen ist, die grossen. Diese Optimierungen haben direkt nichts mit dem Netzwerk zu tun, sondern mit dem Aufbau der Webseite.

Beispiel sandiego.com

In Abbildung 24 wurde der Beginn eines Downloads auf der Client-Seite mit Wireshark aufgezeichnet und daraus ein tcptrace (Time/Sequence Graph) [42] generiert. Die mit ping gemessene RTT zur Webseite www.sandiego.com betrug ca. 160ms. Ganz links unten gibt es ein sehr kleines erstes Paket, welches bei dieser Skalierung jedoch nicht zu erkennen ist. Wobei es sich dort gemäss Wireshark um das SYN/ACK Paket des Servers handelt. Anschliessend vergehen ca. 275ms bis die nächsten Segmente eintreffen. Es dauert also länger als eine RTT. Dies hat vermutlich damit zu tun, dass der Server zuerst die Inhalte der Seite zusammenstellen muss, bevor er mit deren Auslieferung beginnen kann. Das weiter oben beschriebene Delayed ACK Timeout beim ersten Datenpaket kann nicht der Grund sein, da zu Beginn das cwnd bei Windows 7 kaum kleiner ist wie bei Windows XP, wo es auf zwei MSS gesetzt ist (siehe Kapitel 0).

Wie in der Abbildung eingetragen ist, wird pro gesendetes ACK des Clients beim Server das cwnd um eins erhöht. Bei ca. 610ms könnte der Server bereits 4 Segmente schicken, aufgrund des Delayed ACK ist jedoch noch ein ACK ausstehend, weshalb er nur weitere 3 Segmente senden kann. Total sind dann jedoch 4 Segmente ausstehend. Nach einer RTT erhält der Client ein weiteres Segment und kann somit ein weiteres ACK verschicken (immer pro 2 Segmente ein ACK). Das Timeout des Delayed ACK wird nicht überschritten, da hier die RTT kleiner als das Timeout ist.

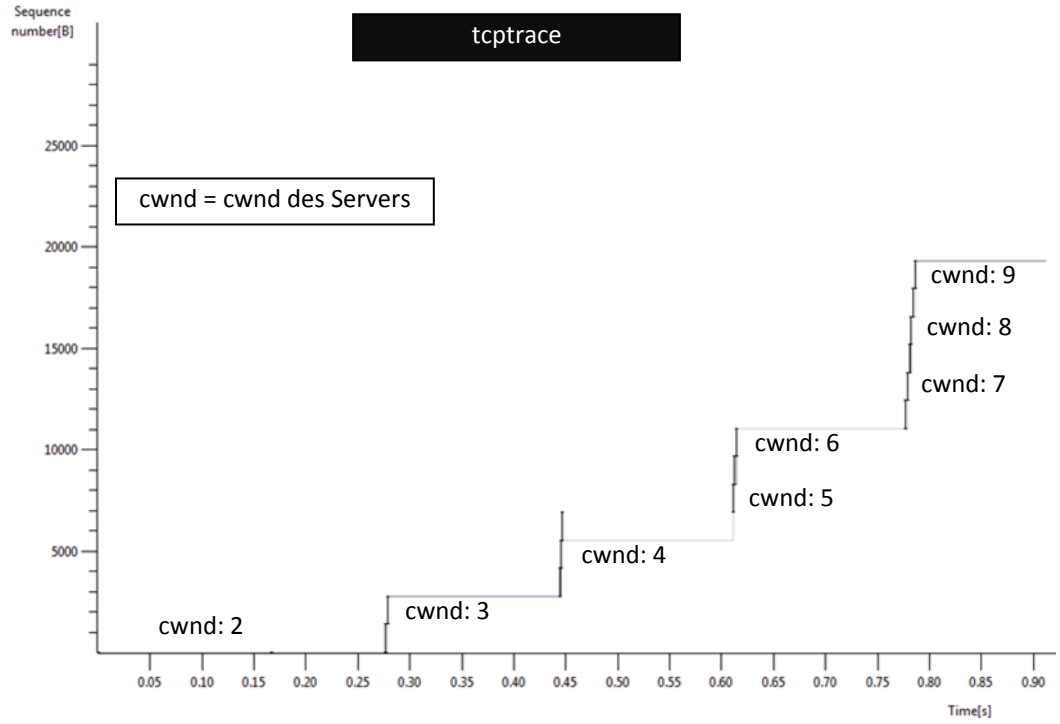


Abbildung 25: tcptrace von TCP Slow Start mit Delayed ACK, Aufzeichnung der empfangenen Datenpakete auf der Clientseite, Download von www.sandiego.com, Windows 7

Der Slow Start ist für den Aufruf von Webseiten relevant, da zu Beginn pro Host mehrere TCP Verbindungen aufgebaut werden. Am Anfang ist die verfügbare Bandbreite auf diesen Verbindungen jedoch aufgrund des Slow Starts relativ tief, wodurch das Laden der ersten Ressourcen länger dauert, als man annehmen würde, nämlich nicht mit voller Geschwindigkeit. Zu Beginn limitiert eine hohe RTT den Datendurchsatz noch stärker als später.

3.8. Congestion Avoidance

Der Slow Start-Prozess dauert so lange, bis ein bestimmter Grenzwert (Variable ssthresh oder oft die RWin des anderen Hosts) erreicht wird oder bis Congestion (Verstopfung) auftritt. Dies wird durch Auftreten des RTO oder durch den Empfang von drei duplizierten ACKs erkannt.

Für die Congestion Avoidance (Verstopfungsvermeidung) gibt es verschiedene Algorithmen. Die zwei Verbreitesten sind TCP Tahoe und der etwas neuere TCP Reno. Die wichtigsten Eigenschaften lauten wie folgt [44]:

(1) TCP Tahoe: Duplizierte ACKs und ein Timeout werden gleich behandelt: cwnd wird auf 1 zurückgesetzt und es wird mit Slow Start begonnen bis ssthresh erreicht wird. Anschliessend wird cwnd nur noch linear erhöht.
(2) TCP Reno: Bei duplizierten ACKs wird das cwnd halbiert, ein Fast Retransmit für das fehlende Paket ausgeführt und die Fast Recovery Phase wird begonnen. Bei einem Timeout reagiert Reno wie Tahoe.

Die Abbildung 26 zeigt das unterschiedliche Verhalten nach einem Timeout bzw. duplizierten ACKs.

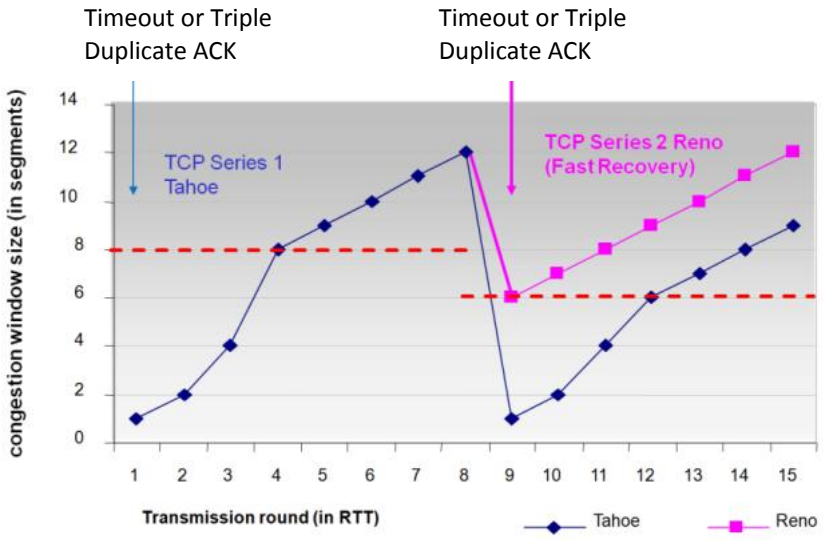


Abbildung 26: Vergleich TCP Tahoe und TCP Reno [39, S. 58]

Ein dupliziertes ACK wird vom Empfänger verschickt, wenn er ein Segment in falscher Reihenfolge erhält [41]. Weil ein vorheriges Segment noch nicht angekommen ist, kann er das aktuelle nicht bestätigen, sondern schickt nochmals ein ACK für das zuletzt richtig empfangene Segment. Der Sender darf jedoch noch nicht annehmen, dass das Segment verloren ging, da es auch andere Gründe für eine falsche Reihenfolge geben kann. Deshalb wartet der Sender auf drei duplizierte ACKs um den Paketverlust festzustellen. Bei TCP Reno findet dann ein Fast Retransmit (schnelles erneutes Senden) des Segments statt ohne das RTO abzuwarten. Bei TCP Tahoe wird dies wie ein Timeout behandelt.

Bei Erkennung der Congestion via duplizierte ACKs wird das cwnd halbiert und dieser Wert bei beiden Algorithmen in der Variable ssthresh (Slow Start Threshold Size) gespeichert. Von dort aus wird dann Congestion Avoidance betrieben bzw. bei TCP Reno beginnt die spezielle Fast Recovery Phase (siehe Quelle [39, S. 58] für weitere Details). Tritt ein Timeout auf, so handelt es sich um grösseres

Problem, d.h. es kommen keine Pakete mehr beim Empfänger an oder alle ACKs gehen verloren. Deshalb wird auch bei TCP Reno dann das cwnd auf 1 gesetzt und es wird wieder mit dem Slow Start begonnen, jedoch nur bis zur Hälfte des vorherigen Window (ssthresh). Ab dort kommt Congestion Avoidance zum Zug.

In der Phase der Congestion Avoidance wird cwnd nur noch linear erhöht bis wieder duplizierte ACKs auftreten. Cwnd sollte um ein Segment pro RTT erhöht werden, egal wie viele ACKs während dieser Zeit eintreffen.

Wie in Abbildung 26 zu sehen ist, wird im Mittel die Netzkapazität ausgenutzt. In den Spitzen(1) werden die Buffer der Router gefüllt und können geleert werden, wenn der die Sendegeschwindigkeit nach duplizierten ACKs reduziert wird (2).

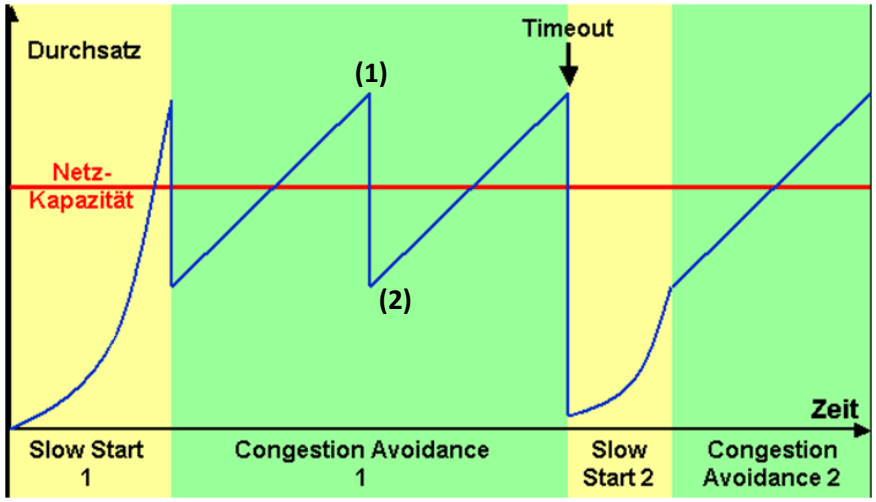


Abbildung 27: TCP-Datenrate bei Anwendung der Algorithmen zur Staukontrolle (schematisch) [43]

4. Betriebssysteme

Wie im Kapitel 1 erläutert, werden in dieser Arbeit nur die Betriebssysteme Windows XP und Windows 7 untersucht und für die Messungen verwendet.

Im Folgenden werden die für die Aufgabenstellung relevanten Unterschiede betrachtet.

4.1. Windows XP

Unter Windows XP sowie Windows Server 2003 gibt es kein Auto Tuning (automatische Einstellung) des RWins wie unter Windows 7. Im Vergleich zu älteren Windows-Versionen werden jedoch grössere RWins verwendet und es findet eine einfache Anpassung an die Schnittstelle statt: Es wird ein gerades Vielfaches der beim Verbindungsaufbau ausgehandelten MSS verwendet anstatt ein fixer Standardwert (siehe Punkt 2 in der folgenden Auflistung).

Abbildung 28: Windows XP Logo

Der Standard-Wert des RWin hängt von den folgenden Punkten ab. Hier in der Reihenfolge der Priorität aufgelistet [30]:

1. Anwendung
   Die Anwendung kann mittels der SO_RCVBUF Windows Sockets-Option die maximale RWin beim Verbindungsaufbau festlegen. Soll Window Scaling verwendet werden, so muss einfach ein Wert über 64 kB angegeben werden.
2. Registry
   Die RWin kann in der Registry manuell konfiguriert werden. Es gibt einen Wert pro Interface sowie einen globalen Wert. Der Standardwert ist im Minium 8192 Bytes oder ein gerades Vielfaches der MSS. Für via Ethernet angebundene Computer beträgt er 8760 Bytes (6 * 1460).
3. Auto-Erkennung
   Dieser Wert beruht auf der Verbindungsgeschwindigkeit der Netzwerkschnittstelle des Senders:
   - Unter 1 Mbit/s: 8 kB
   - Zwischen 1 und 100 Mbit/s: 17 kB
   - 100 Mbit/s und höher: 64 kB

Windows XP unterstützt zwar das Window Scaling, macht jedoch von sich aus davon nicht Gebrauch. Standardmässig wird es nur verwendet, wenn der Kommunikationspartner dies beim Verbindungsaufbau wünscht (TCP-Option im SYN Paket gesetzt) oder wenn der Programmierer explizit ein grösseres RWin als 64 kB wählt.

Pro Netzwerkschnittstelle existiert ein fixer maximaler RWin-Wert. Nur wenn der Programmierer die Windows Sockets-Option verwendet, kann er das RWin je nach Kontext anpassen. Dabei wird es jedoch nicht aufgrund des aktuellen Netzwerkzustandes angepasst, sondern aufgrund der Datenverarbeitungsgeschwindigkeit der empfangenden Applikation.

Bei Windows XP und Windows Server 2003 ist das anfängliche cwnd standardmässig auf zwei MSS gesetzt [30], um das im Kapitel 3.7 beschriebene Timeout nach dem Verbindungsaufbau bei Verwendung von Delayed ACK zu umgehen.

Als Congestion Avoidance-Algorithmus verwendet Windows XP die TCP Reno-Variante mit Fast Retransmit [51]. Das Delayed ACK Timeout beträgt standardmässig 200ms und es wird versucht für jedes zweite Segment ein ACK zu schicken, wie es im RFC 1122 empfohlen wird [30].

4.2. Windows 7

Der TCP/IP Stack wurde für Windows Vista und Windows Server 2008 komplett neu geschrieben. Somit erfüllt er die Anforderungen an die aktuell genutzten Netzwerkumgebungen. Er löst den veralteten, aus den 90er Jahren stammenden Stack von Windows XP und Windows Server 2003 ab. Die neue Version trägt den Namen „Next Generation TCP/IP Stack“.

Abbildung 29: Windows 7 Logo

Es wurden diverse neue Funktionen implementiert und einige Verbesserungen vorgenommen. Das wichtigste Merkmal des neuen TCP/IP Stacks bzgl. dieser Arbeit ist das Auto-Tuning. Der Stack erkennt automatisch die Netzwerkumgebung und passt die Einstellungen wie die TCP Receive Window-Grösse entsprechend an. Dadurch sollen gemäss Microsoft „TCP-Leistungsverbesserungen bei Hochgeschwindigkeitsnetzwerken, asymmetrischen Satellitenverbindungen und WLANs“ erreicht werden [46].

Aus den vielen neuen Funktionen [52] des Next Generation TCP/IP Stacks werden folgende genauer dargestellt:

- TCP Receive Window Auto-Tuning
- Compound TCP

Andere Funktionen wie z.B. Explicit Congestion Notification (ECN) werden hier nicht betrachtet.

Bei Windows 7 konnte keine Angabe zur anfänglichen cwnd gefunden werden. Der Wireshark Trace von Abbildung 25 lässt jedoch denselben Wert wie bei Windows XP vermuten (2 MSS). Das Timeout für das Delayed ACK beträgt standardmässig ebenfalls 200ms wie unter Windows XP und es wird auch versucht, für jedes zweite Segment ein ACK zu schicken [58].

4.2.1. TCP Receive Window Auto-Tuning

Die RWin-Grösse wird durch Messen der aktuellen Verbindungsqualität (Bandbreite und Verzögerung, siehe Kapitel 0) sowie der Datenabrufrate der Anwendung [47] aus dem Empfangsbuffer regelmässig automatisch angepasst. Diese Messungen und die Anpassungen des RWin geschehen nicht nur beim Verbindungsaufbau, sondern auch während der Datenübertragung. So kann direkt auf Veränderungen der Verbindungsqualität und des Applikationsverhaltens reagiert und der Datendurchsatz optimiert werden. Hierbei handelt es sich um eine Datendurchsatzoptimierung auf der Empfängerseite.

Für diese Funktion wird das Window Scaling aus RFC 1323 verwendet. Window Scaling ist standardmässig bis zu einem Maximalwert von 16 MB aktiviert. Die RWin wird im Gegensatz zu Windows XP

automatisch individuell pro Verbindung gesetzt. Der Programmierer muss sich also nicht mehr mit der optimalen Wahl des RWin auseinandersetzen und auch die manuelle Konfiguration der RWin für spezifische Computer entfällt.

Da die Bandbreite nun automatisch besser genutzt wird, steigt die Auslastung der Verbindungen an und die Verwendung von Quality of Service (QoS) kann nötig werden um z.B. die Qualität von VoIP Gesprächen sicherzustellen [49]. Deshalb wird ab Windows Vista eine QoS Konfiguration mittels Gruppenrichtlinien unterstützt.

4.2.2. Compound TCP

Bei Compound TCP (CTCP) handelt es sich um eine Datendurchsatzoptimierung auf der Senderseite, indem das Congestion Window cwnd bei Verbindungen mit hohem Bandbreiten-Delay-Produkt aggressiver angepasst wird. Denn der klassische Algorithmus erhöht das cwnd in diesem Fall nicht schnell genug um die volle Bandbreite der Verbindung nutzen zu können. Dabei wird jedoch geschaut, dass andere TCP-Verbindungen nicht negativ beeinflusst werden [48].

Standardmässig ist diese Funktion in Windows 7 deaktiviert [48], jedoch auf dem Windows Server 2008 standardmässig aktiviert. Ab Windows Server 2008 kommen die Internet Information Services 7 zum Einsatz. Wird die Webseite also auf einem solchen Server bereitgestellt, ist CTCP aktiviert, solange der Administrator es nicht manuell deaktiviert hat. Gemäss [53] gibt es jedoch einige Probleme mit CTCP und einige Punkte sollten überarbeitet werden vor einem grossflächigen Einsatz. Die Verbreitung des IIS 7 ist jedoch noch nicht weit fortgeschritten. Je nach Statistik hat er einen Marktanteil von 3-6% [54][55]. Deshalb wird CTCP in dieser Arbeit nicht genauer betrachtet.

4.3. Vergleich

Folgende Tabelle zeigt der Vergleich der TCP/IP Stacks von Windows XP und Windows 7 bzgl. der untersuchten wichtigsten Parameter.

Der Hauptunterschiede, welcher für die Messungen relevant ist, ist das TCP Receive Window Auto Tuning, welches neu ab Windows Vista dazugekommen ist. Gerade bei hoher Bandbreite und grosser Verzögerung sollten deshalb bei Windows 7 schnellere Datenübertragungen möglich sein als bei Windows XP.

	Standard RWin[3]	**Window Scaling**	**anfängliches cwnd**	**Delayed ACK Timeout**
Windows XP	65‘535 Bytes	nur wenn explizit vom Programmierer gewählt oder von anderem Host genutzt	2 MSS	200ms
Windows 7	k.A., vom Auto-Tuning gewählt	automatisch aktiviert	2 MSS	200ms

Tabelle 5: Vergleich TCP/IP Stack von Windows XP und Windows 7

[3] Bei einem 100 Mbit/s Netzwerkanschluss

5. Browser

Bei den Messungen sollen die aktuellen Varianten der Top 3-Browser bzgl. Veränderung der Netzwerkleistungsparameter verglichen werden. Wie im Kapitel 1 bereits erwähnt, werden die aktuellen Versionen von Internet Explorer, Firefox und Chrome verwendet. Genauere Informationen zum Testsystem sind Kapitel 9.8.2 der Arbeit zu finden.

5.1. Verbindungen

Eine Webseite besteht aus mehreren verschiedenen Objekten wie der HTML Seite, Bildern, CSS- und JavaScript-Dateien. Für jedes dieser Objekte ist eine Anfrage an den Webserver nötig, ein sogenannter GET-Request.

Bei der HTTP Version 1.0 (RFC 1945) wurde für jede Anfrage eine neue TCP-Verbindung zum Webserver aufgebaut, welche nach Übertragung der Daten wieder geschlossen wurde. Deshalb wurden pro Objekte zwei RTTs benötigt, bis die ersten Daten eintrafen (TCP Handshake und Versenden der Anfrage).

Persistente Verbindungen

Mit der HTTP Version 1.1 (RFC 2616, Jahr 1999) wurden persistente Verbindungen sowie Request-Pipelining eingeführt. Persistente Verbindungen waren zwar schon mit HTTP/1.0 möglich, jedoch nicht standardmässig aktiviert. Neu wird die Verbindung nach Erhalt der Daten nicht mehr geschlossen, sondern bleibt für weitere Anfragen offen, bis einer der Hosts die Verbindung explizit schliesst oder das Timeout eintritt. Somit ist die nächste Anfrage schneller, da keine neue TCP-Verbindung aufgebaut werden muss und somit das TCP Handshake entfällt. Die gleiche Verbindung kann mehrmals benutzt werden (siehe Abbildung 30, links). Dadurch wird auch die langsame Slow Start-Phase einer neuen Verbindung übersprungen, was eine schnellere Ladegeschwindigkeit und somit Zeitersparnis zur Folge hat.

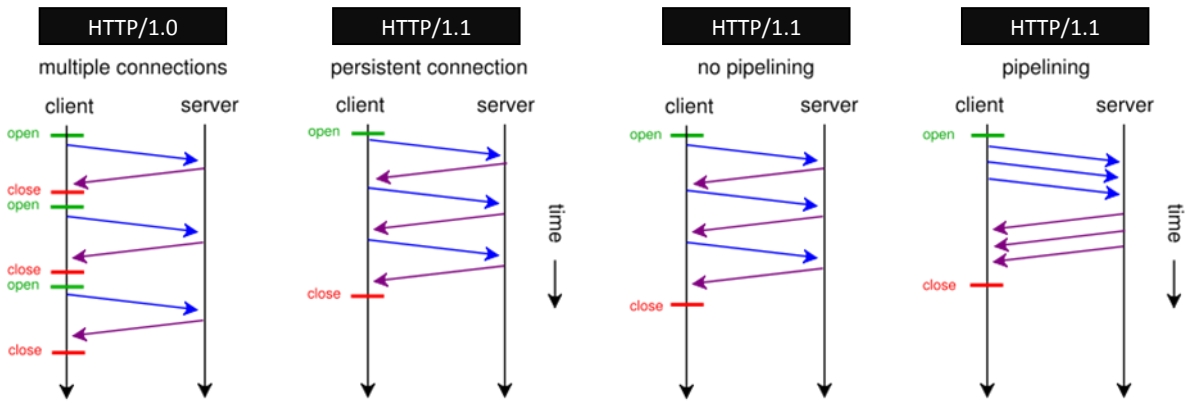


Abbildung 30: Visualisierung persistente Verbindung und Pipelining [63]

Pipelining

Die Anfragen für die Objekte werden jedoch weiterhin sequentiell abgeschickt und erst eine RTT später treffen die ersten Daten beim Client ein. Mit der neuen Funktion Pipelining, welche persistente Verbindungen voraussetzt, können mehrere Anfragen aufs Mal gesendet werden ohne dabei auf die entsprechende Antwort zu warten (siehe Abbildung 30 rechts). Da nicht auf die Antwort gewartet werden muss, wird pro Objekt-Anfrage bzw. pro Objekt eine RTT gespart. Ebenso müssen weniger Pakete verschickt werden, da meistens mehrere HTTP-Anfragen in ein TCP-Segment passen. Der Server muss die Antworten in der gleichen Reihenfolge schicken, wie die Anfragen angekommen sind. Aus diesem Grund wird der Zeitgewinn jedoch eventuell geschmälert, wenn die Bearbeitungszeit von frühen Anfragen länger ist, als jene von späteren [63]. Von den am weitesten verbreiteten Browser ist einzig bei Opera das Pipelining standardmässig aktiviert, schon seit Version 4. Bei Safari wird es überhaupt nicht unterstützt [77].

CSS Sprites

Da Pipelining nicht eingesetzt wird, kostet eine Anfrage eines Webseiten-Objekts doch eine RTT. Aus diesem Grund sollte ein Webentwickler seine Seite so optimieren, dass möglichst wenig HTTP GET-Anfragen nötig sind. So können oft mehrere CSS- oder JavaScript-Dateien zu einer Datei zusammengefasst werden. Auch mehrere Bilder können mittels CSS-Sprites [76] zusammengefasst werden, wenn es sich um viele und kleine Bilder wie Icons handelt. Dabei werden mehrere Bilder nebeneinander in ein grosses Bild gepackt (siehe Abbildung 31) und mittels CSS-Code wird nur der entsprechende Teil des grossen Bildes angezeigt. So ist nur eine HTTP-Abfrage nötig und es wird ein grosses Bild anstatt viele kleine geladen. So wird auch die Verbindung besser ausgenutzt.

Abbildung 31: Beispiel für eine CSS-Sprite Grafik

Für weitere grafische Vergleiche der verschiedenen Funktionen sei das JavaApplet „HTTP Delay Estimation“ [64] empfohlen.

Parallele Verbindungen

Um die Internetverbindung besser auszulasten und mehrere Ressourcen parallel anfordern zu können, öffnen heutige Browser mehrere TCP-Verbindungen pro Webserver. Gemäss RFC 2616 (HTTP/1.1) sollte kein Client mehr als zwei gleichzeitige Verbindungen zu einem Server öffnen, wobei es sich damit nur um eine Empfehlung handelt. Wie die Tabelle 6 zeigt, überschreiten alle genannten Browser diesen Wert. Dieses Limit gilt pro Server, wobei Server für einen DNS-Alias steht. Auch wenn verschiedene DNS-Aliase (CNAMEs) auf dieselbe IP verweisen, so werden die Aliase als verschiedene Server interpretiert [65] und ein Vielfaches des Verbindungslimits an Verbindungen auf die gleiche IP ist möglich. Mehr als 8 bis 10 parallele Verbindungen pro Server dürften jedoch kaum einen Geschwindigkeitsvorteil bringen, da spätestens dann die Bandbreite der beschränkende Faktor ist.

In der Tabelle 6 werden die untersuchten Browser auf die in diesem Zusammenhang stehenden Funktionen analysiert. Mit „Total“ ist die totale Anzahl Verbindungen gemeint, welche der Browser über alle Server hinweg gleichzeitig öffnen kann.

	HTTP/1.1, pro Server persistent	**HTTP/1.0, pro Server nicht persistent**	**Total**	**Pipelining [57]**
Internet Explorer	6	6	35[4]	nicht unterstützt
Firefox	6	15	30	deaktiviert
Chrome	6	k.A.	35[5]	nicht unterstützt (in Arbeit)

Tabelle 6: Anzahl unterstützter paralleler Verbindungen pro Browser inkl. Pipelining

Interessant ist, dass alle Browser den Download einer Datei nicht auf mehrere Verbindungen aufteilen. Bei kleinen Dateien ist dies verständlich, da dort der Mehraufwand zu gross wäre, aber gerade bei grossen Dateien und einer hohen RTT liesse sich der Download massiv beschleunigen (siehe Kapitel 7.3.2). Aktuelle Download-Manager unterstützen jedoch die Dateiübertragung mittels mehreren Verbindungen und es gibt auch Browserplugins dafür (z.B. das Plugin FireDownload [75] für Firefox).

Die Browser unterscheiden sich nicht bzgl. der Unterstützung von parallelen Verbindungen pro Server. Firefox erlaubt standardmässig leicht weniger totale Verbindungen, aber dieses Limit wird vermutlich selten erreicht. Es müssten entweder zu fünf Servers jeweils sechs Verbindungen aufgebaut sein und das alles gleichzeitig. Auch bei der getesteten Webseite 20min.ch mit mehreren Hosts und vielen geladenen Objekten war dies bei weitem nicht der Fall, da die meisten Daten nur von einem Host geladen wurden und für fast alle anderen Hosts nur eine Verbindung aufgebaut wurde, da es von dort nur wenige Daten zu übertragen galt.

5.2. Paralleles Laden von Ressourcen

Wie im vorherigen Kapitel geschildert, sind Browser dank parallelen Verbindungen in der Lage verschiedene Objekte einer Webseite parallel zu laden. Interessanterweise gibt es jedoch Unterschiede bei den Browsern, wie die verschiedenen Dateitypen behandelt werden.

Scripts wurden z.B. von früheren Browsern (IE 7 und Firefox 2) nicht parallel mit anderen Ressourcen geladen. Nach dem Anfordern eines Scripts wurden alle weiteren Anfragen geblockt, bis das Script geladen und ausgeführt worden war. Andere Ressourcen konnten erst danach parallel geladen werden. Der sequentielle Download und das direkte Ausführen garantierte die Reihenfolge der Scriptausführung. Dies ist wichtig, wenn Abhängigkeiten zwischen den Scripts bestehen [67].

Nichtsdestotrotz könnte der Browser das zweite Script und die anderen Ressourcen parallel zum ersten Script herunterladen. Er muss nur die Ausführungsreihenfolge garantieren. Wie in der Tabelle 7 zu sehen ist, sind die aktuellen Browser in der Lage fast alle Ressourcen parallel zu Scripts herun-

[4] Bzgl. der Gesamtanzahl offener Verbindungen beim IE konnte keine offizielle Angabe gefunden werden. Gemäss [56] und [65] gibt es dort kein bzw. ein sehr hohes Limit. Auf browserscope.org findet man hingegen den Wert 35.

[5] Gemäss Statistik auf browserscope.com schafft Chrome 7 total 30 Verbindungen. Lässt man jedoch den Netzwerktest mit diesem Browser laufen, so ist das Resultat 35.

terzuladen. IFrames werden jedoch immer noch erst nach dem Laden und Ausführen der vorangestellten Scripts heruntergeladen. Es existiert also noch Spielraum für weitere Optimierungen.

In der Tabelle 7 steht „Async Scripts“ für eine neue Funktion von HTML5. Es handelt sich dabei um dieselbe Funktion wie das Link Prefetching, welches im Kapitel 7.3.5 erwähnt wird. Dabei wird das Script heruntergeladen ohne direkt ausgeführt zu werden [68]. Es gibt dem Webentwickler die Möglichkeit, selber kontrollieren zu können, wann ein Script heruntergeladen und ausgeführt wird. Dies ist ein wichtiger Punkt, da der Entwickler die Webapplikation am besten versteht und genau weiss, zu welchem Zeitpunkt ein Script benötigt wird. Dies erlaubt fortgeschrittenen Webentwicklern eine genaue Performance-Optimierung der Webseite.

Die folgende Tabelle ist wie folgt zu lesen: Ein Haken bei „|| Script CSS“ bedeutet beispielsweise, dass beim Laden eines Scripts auch parallel eine CSS Datei geladen werden kann. Steht dort ein Kreuz, so blockt der Browser beim Download eines Script das Laden der CSS Datei.

	|| Script Script	**|| Script CSS**	**|| Script Bild**	**|| Script IFrame**	**Async Scripts**	**|| CSS**	**|| CSS Inline Script**
IE	✓	✓	✗	✗	✗	✓	✗
Firefox	✓	✓	✓	✗	✓	✓	✓
Chrome	✓	✓	✓	✗	✗	✓	✗

Tabelle 7: Browservergleich paralleles Laden von Ressourcen [15]

5.3. Rendering und Scripts

Überraschend ist auch, dass Scripts in der HTML-Seite das Rendering blockieren [68][69]. Alle Elemente unterhalb des <script>-Tags werden erst nach dem Laden und Ausführen des Scripts gerendert. Das HTML-Dokument ist zu diesem Zeitpunkt bereits komplett heruntergeladen, aber es kann noch nicht alles vom Browser dargestellt werden. Dies ist besonders ärgerlich, wenn ein Script im <head>-Teil der Seite bzw. sehr früh im <body>-Teil referenziert wird. Dies blockiert das Rendering der ganzen Webseite bis das Script heruntergeladen und ausgeführt wurde.

Ein extremes Beispiel ist der Ausfall von Google im Mai 2009 [72]. Da von vielen Webseiten für die Erfassung von Benutzerstatistiken die Google Analytics JavaScript-Datei eingebunden wird, waren viele Webseiten vom Ausfall betroffen. Die Auswirkungen sind jedoch je nach Platzierung des Scripts in der Webseite unterschiedlich spürbar. Bei sbb.ch sowie 20min.ch wird das Script bereits sehr früh in der HTML Datei eingebunden: Bei sbb.ch im Header und bei 20min.ch direkt danach. Wird nun die Nicht-Erreichbarkeit der Domain google-analytics.com durch einen Eintrag in der hosts-Datei auf dem Computer simuliert, so dauert es bis zu 21s bis die Webseite dargestellt wird. Zuvor sieht man nur eine leere Seite. Nur der Titel des Browserfensters ändert bereits, da der <title>-Tag im Header vor dem Script platziert ist und diese Information bereits dargestellt werden kann. Die ganze HTML sowie die CSS-Dateien wurden bereits fertig heruntergeladen, doch das Rendering der Elemente unterhalb des Google Analytics-Scripts wird blockiert. Wird jedoch das Script erst am Ende der HTML-Datei eingebunden oder asynchron geladen, so merkt der Benutzer kaum etwas vom Ausfall. Die Seite wird normal geladen, ausser dass am Schluss für ca. 21s in der Statusleiste des Browsers der Verbindungs-

versuch zu www.google-analytics.com angezeigt wird. Dies ist jedoch kein Problem und dürfte den meisten Endbenutzern nicht einmal auffallen.

Gerade beim Einbinden von externen Scripts ist daher Vorsicht geboten, da man die Verfügbarkeit dieser Ressourcen nicht selber in der Hand hat. Es gibt verschiedene Möglichkeiten um dieses Problem zu umgehen. Steve Souders gibt auf seiner Webseite [68][69] einen Überblick über die Varianten. Beispielsweise besteht die Möglichkeit, die Scripts falls möglich erst am Ende der Seite einzubinden oder asynchron zu laden. Durch asynchrones Laden ist auch ein paralleler Script-Download in älteren Browsern möglich und Probleme durch blockiertes Rendering werden gemindert. Es gibt auch spezielle JavaScript-Frameworks wie ControlJS [70] oder jQuery [73], welche den Webentwicklern mehr Kontrolle über das Laden und Ausführen von Scripts geben.

Es ist klar ersichtlich, dass der Aufbau einer Webseite einen sehr starken, wenn nicht sogar fast den grössten Einfluss auf das Endbenutzererlebnis bzgl. der Ladegeschwindigkeit der Webseite. Dazu gibt es von Steve Souders zwei Bücher (siehe Kapitel 9.6.3) mit vielen Tipps [71].

5.4. JavaScript-Performance

Die JavaScript-Performance wird bei den aktuellen Browsern immer wichtiger, da bei vielen Webseiten immer mehr auf JavaScript gesetzt wird. Kaum eine grössere Webseite kommt ohne JavaScript aus und je nach Seitenaufbau, trägt die JavaScript-Performance viel zur Geschwindigkeit des Seitenaufbaus bei. Viele RIA (Rich Internet Applications) beruhen auf AJAX, was für „Asynchronous JavaScript and XML“ steht. Aus diesem Grund und um das Browserverhalten bei JavaScript-lastigen Seiten abschätzen zu können, wurden nachfolgend die Resultate der WebKit SunSpider Benchmark Suite [61] dargestellt. Dabei handelt es sich um einen Benchmark, welcher die Leistung bei realitätsnahen Aufgaben wie Ver-/Entschlüsselung und Text Manipulationen prüft [62]. Beim verwendeten SunSpider JavaScript Test handelt es sich um die Version 0.9.1, welche seit April 2010 verfügbar ist.

Gemäss Abbildung 32 wird Chrome hier seinem Ruf als schneller Browser für moderne Webapplikationen gerecht. Firefox steht mittelmässig da und der IE 8 ist weit abgeschlagen der langsamste Browser bzgl. JavaScript. Dies wird sich jedoch mit der neuen Version 9 ändern.

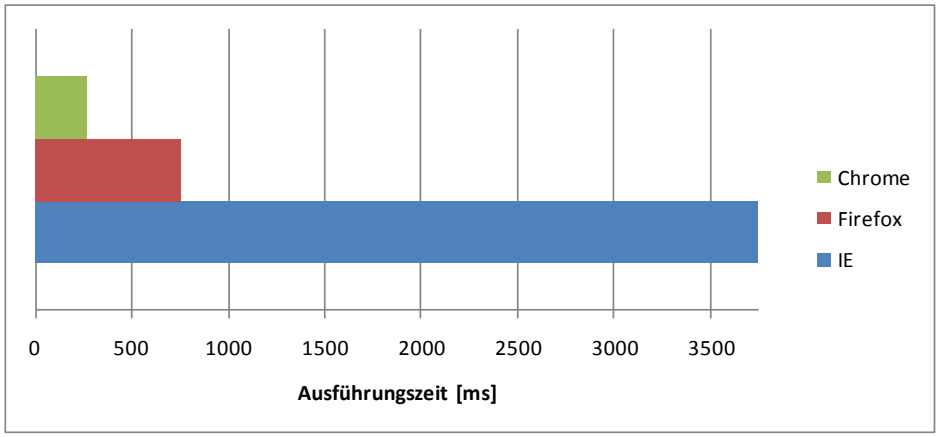


Abbildung 32: Vergleich JavaScript Performance bzgl. SunSpider Benchmark [61], kleiner ist besser

5.5. Standard-Konformität

Die Konformität der Browser mit den von W3C definierten Webstandards ist für Benutzer relevant, da diese Standards definieren, wie eine Webseite aussehen sollte. Werden nicht alle Standards von einem Browser erfüllt, so kann es dazu führen, dass eine Seite nicht korrekt angezeigt wird oder gewisse Elemente gar nicht funktionieren. Aus diesem Grund müssen oft verschiedene Versionen der Webseite erstellt werden und mit allen weit verbreiteten Browsern getestet werden. Dies ist sehr aufwändig und verlangsamt dadurch die Webentwicklung, weshalb Webentwickler Browser mit schlechter Standardkonformität nicht schätzen. Werden jedoch die speziellen Anpassungen nicht gemacht, so ärgern sich die Benutzer, wenn gewisse Teile der Webseite unbenutzbar sind. So läuft z.B. der Dromaeo JavaScript Test [66] aufgrund der schlechten Implementierung von Standards im IE nicht komplett durch.

Wie in Tabelle 8 ersichtlich ist, erfüllt der Chrome alle und der Firefox fast alle Punkte der beiden getesteten Standards Acid3 Test [60] und CSS3 Selector Test [59]. Nur beim IE sieht es diesbezüglich schlecht aus, was sich jedoch mit der Version 9 stark verbessern wird (95% bei Acid3 mit der Preview Version). Alle anderen Topbrowser, welche auf browserscope.org gelistet sind, erreichen beim Acid3 Test mittlerweile 93% und mehr.

	ACID3 browserscope	CSS3 Selectors Test
Internet Explorer 8	20%	60%
Firefox 3.6	94%	100%
Chrome 7	100%	100%

Tabelle 8: Standard-Konformität der Browser, grösser ist besser

Der Acid3 Test ist seit Anfang 2008 verfügbar und konzentriert sich auf Technologien, welche in heutigen interaktiven Webseiten genutzt werden. Ebenso werden auch die Unterstützung von SVG (Scalable Vector Graphics), XML und CSS2 untersucht. Der CSS3 Selector Test prüft die Unterstützung einer Vielzahl von CSS Selektoren [74] inkl. vieler, welche neu in CSS3 hinzugekommen sind. Es ist somit eine Aussage zur CSS-Komptabilität.

6. Netzwerk-Emulator (NetEm-Box)

Abbildung 33: NetEm-Box / Microspace MPC20

Damit die EUE unter verschiedenen Netzwerk-Einflüssen wie Bandbreite, Verzögerung usw. gemessen werden kann, müssen diese Eigenschaften emuliert werden. Das Betriebssystem Linux stellt dazu von Haus aus eine Traffic Control (Verkehrskontrolle) zur Verfügung, womit verschiedene Netzwerk-Parameter für durchfliessenden Verkehr beeinflusst werden können. Im Rahmen der Bachelorarbeit „Netzwerk-Emulatoren auf dem Prüfstand“ von V. Condoleo, C. Dirac und R. Ruoss [78] von 2008 wurde ein Embedded-System aufgesetzt (NetEm-Box), welches das Nachstellen von verschiedenen Netzwerksituationen erlaubt. Das Embedded-System basiert auf der Linux-Distribution Fedora und kann über eine Web-Oberfläche verwaltet werden.

Diese NetEm-Box wird im Rahmen dieser Arbeit für die Emulation von verschiedenen Netzwerksituationen verwendet.

6.1. Einleitung

Wie bereits beschrieben, erlaubt die NetEm-Box den durchfliessenden Verkehr zu beeinflussen. In diesem Kapitel wird die genaue Funktionsweise des Emulator und dessen Konfiguration beschrieben.

Folgende Netzwerkparameter können verändert bzw. emuliert werden:

- Verzögerung
- Jitter (Variation der Verzögerung)
- Paketverlust
- Paketkorruption
- Paketduplizierung
- Paketumordnung
- Bandbreite

Mit dieser Spannweite an veränderbaren Netzwerkparametern können verschiedenste Netzwerksituationen nachgestellt werden. Dies erlaubt ein Testen von Applikationen unter besonderen Netzwerkbedingungen. In unserem Fall verändert die NetEm-Box den Datenverkehr, damit die EUE beim Besuchen von Webseiten unter speziellen Bedingungen gemessen werden kann.

Wie in Abbildung 34 ersichtlich ist, kann die NetEm-Box zwischen zwei Netzwerke geschaltet werden, damit der Datenverkehr dazwischen beeinflusst werden kann. Eines der Netzwerke kann auch das Internet darstellen, falls der Verkehr dorthin verändert werden soll. Es können jedoch auch direkt Computer an die NetEm-Box angeschlossen werden. Bei dieser Arbeit stellt dies den Normalfall dar (siehe Abbildung 86).

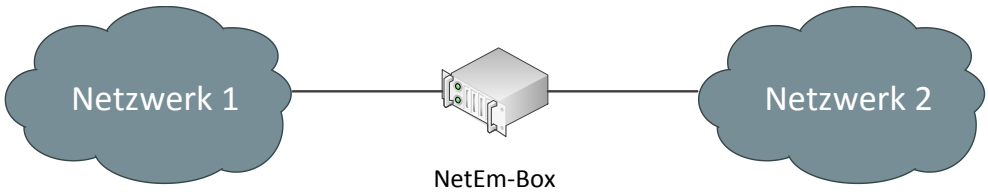


Abbildung 34: NetEm-Box im Netzwerk

Die NetEm-Box kann nicht transparent betrieben werden, sondern agiert als Router und muss bei den Clients als Standard-Gateway eingetragen werden. Diese Einstellung kann auch automatisch über DHCP verteilet werden; dies sogar direkt über die NetEm-Box, da sie über einen eigenen DHCP Server verfügt. Diese und andere Optionen zur Konfiguration des Geräts sind ausführlich im Benutzerhandbuch dokumentiert [78, Netzwerk-Emulator Benutzerhandbuch].

6.2. Funktionsweise

Damit die Bandbreite durch die NetEm-Box beeinflusst werden kann, wird ein sogenanntes Traffic Shaping (Bandbreitenlimitierung) verwendet. Dies bedeutet, dass für einen bei der NetEm-Box ankommenden Datenstrom die Bandbreite am ausgehenden Anschluss limitiert wird. Oft wird diese Technik dazu eingesetzt, um gewisse Dienste im Netzwerk gesondert zu behandeln. Telefonie-Verkehr ist z.B. viel sensitiver als anderer Datenverkehr und in der Praxis wird ihm oft eine minimale Bandbreite garantiert. Der restliche Datenverkehr teilt sich dann die übrige Bandbreite. In diesem Zusammenhang wird von QoS (Quality of Service) gesprochen.

Wie in Abbildung 35 zu erkennen ist, kann durch Traffic Shaping eine Teilbelegung der Gesamt-Bandbreite erreicht werden.

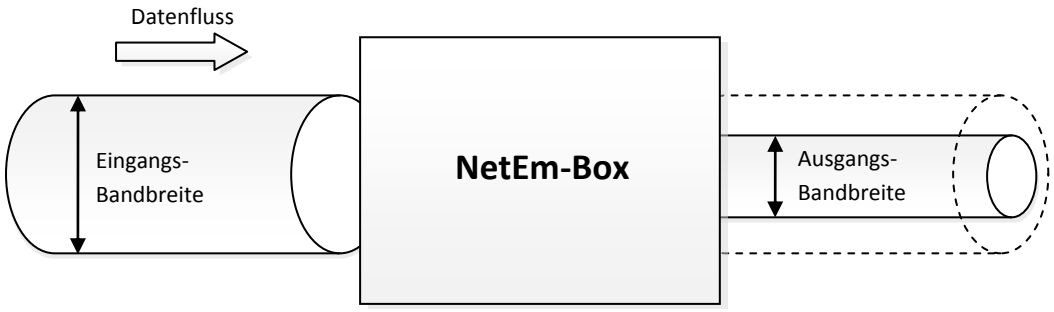


Abbildung 35: Traffic Shaping

Die Veränderung des Datenflusses wird intern in der NetEm-Box durch verschiedene Warteschlagen realisiert, welche der durchfliessende Verkehr passieren muss. Diesen Warteschlagen sind verschiedene Qdisc (Queuing Discipline, Warteschlagendisziplin) zugeordnet, welche aus verschiedenen Algorithmen bestehen und die gewünschten Effekte wie Paketverlust, Verzögerung, Traffic Shaping usw. ermöglichen [79]. Bei der Umsetzung der NetEm-Box mussten zwei Qdiscs verwendet werden, da es keinen Qdisc gibt der alle benötigten Effekte beherrscht. Es wurden die Qdisc „netem“ und „tbf“ (Token Bucket Filter) benutzt.

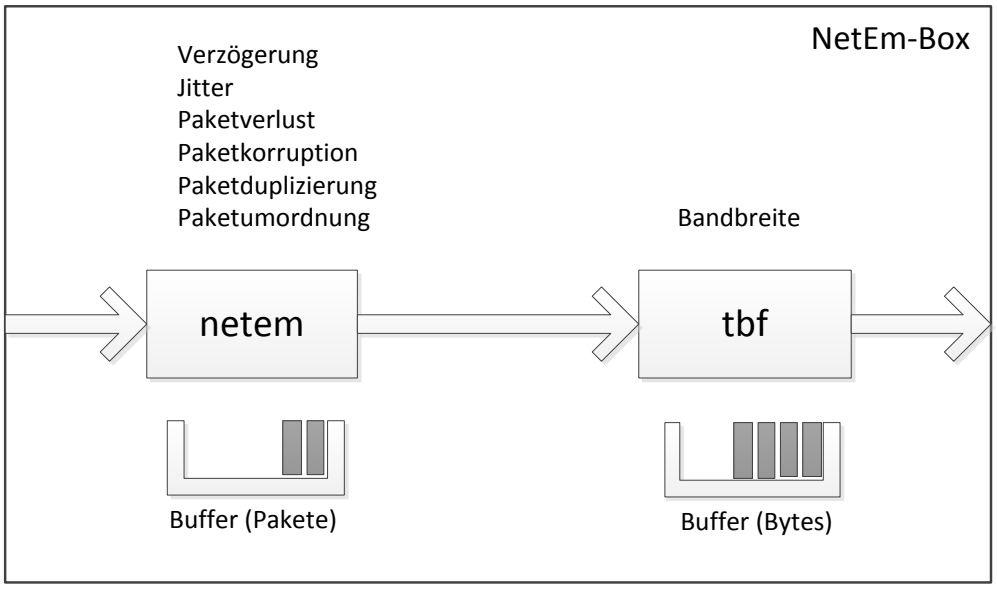


Abbildung 36: Verwendete Qdisc

Bei „netem“ handelt es sich um einen Qdisc, der ausser des Traffic Shapings alle gewünschten Effekte beherrscht. Die Effekte Paketverlust und Paketkorruption werden in Echtzeit auf den Datenverkehr angewandt. Bei den restlichen Effekten wird ein Buffer benötigt, damit die Effekte entsprechend wirken können. Hauptsächlich wird dieser Buffer jedoch vom Effekt Verzögerung beansprucht (siehe Kapitel 6.3.1.2, Buffergrösse „netem“).

Für das Traffic Shaping wird der zweite Qdisc „tbf“ benötigt, welcher folgendermassen funktioniert: Man stellt sich beim „Token Bucket Filter“ einen Bucket (Behälter) mit einem bestimmten Volumen vor (siehe Abbildung 37). Von einem Tokengenerator werden Tokens mit einer konstanten Rate erzeugt, welche den Bucket kontinuierlich füllen. Diese konstante Rate entspricht dem Parameter Bandbreite. Falls nun ein Paket versandt werden will, muss für jedes Byte des Paketes ein Token im Bucket entfernet werden können, ansonsten wird es nicht versandt.

Aus dieser Funktionsweise ergeben sich drei Szenarien:

1. Treffen die zu sendenden Pakete mit der gleichen Rate ein, wie der Tokengenerator neue Tokens erzeugt, darf jedes Paket sofort versandt werden.
2. Treffen die Pakete schneller ein, müssen sie warten, bis wieder ausreichend Tokens vorhanden sind. Dies drosselt die Senderate auf die Tokenrate des Generators, womit das Traffic Shaping aktiv wird. Die Pakete, welche nicht sofort versandt werden können, müssen im Buffer zwischengespeichert werden.
3. Treffen die Pakete mit einer geringeren Rate ein oder kommen gar keine Pakete an, tröpfeln die Tokens weiterhin in den Bucket. Ist dieser irgendwann voll, landen die überflüssigen Tokens im elektronischen Mülleimer.

Ist der Bucket voll und es kommen neue Datenpakete mit einer hohen Rate an, verbrauchen sie die angesammelten Tokens. Bis der Bucket leer ist, dürfen Pakete mit einer höheren Rate versandt werden, als dem „tbf“ eigentlich zusteht. Es kommt zu einem so genannten Burst [78, Netzwerk-Emulator Dokumentation, Kapitel 4, Teil „tbf“ teilweise übernommen].

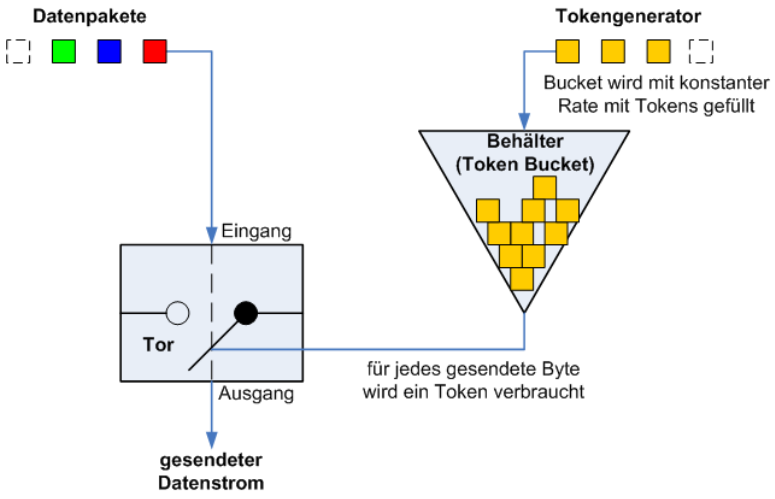


Abbildung 37: Prinzip des „Token Bucket Filter“ [78]

Um beim Traffic Shaping dem Parameter Bandbreite gerecht zu werden, dürfen keine Bursts erlaubt sein. Dies wird dadurch erreicht, dass der Bucket nur ein Volumen für Tokens von genau einem Paket besitzt.

Es existieren noch andere Qdiscs, welche für das Traffic Shaping verwendet werden könnten. Der bekannteste ist der „htb“ (Hierarchical Token Bucket), welcher im Gegensatz zum „tbf“ ein klassenbasierender Qdisc ist [80]. Damit lassen sich komplizierte Strukturen von Traffic Shaping-Anforderungen umsetzen, was jedoch für diese Arbeit nicht erforderlich ist und darum hier auch nicht weiter erläutert wird.

Eine nähere Beschreibung der beiden Qdiscs „netem“ und „tbf“ ist in der Dokumentation der Bachelorarbeit zu finden [78, Netzwerk-Emulator Dokumentation, Kapitel 4].

6.3. Optimierungen

Bei der Verwendung der NetEm-Box fielen einzelne Mängel auf. Soweit wie möglich wurden diese im Rahmen der vorliegenden Arbeit behoben.

In den folgenden Unterkapiteln werden diese Punkte beschrieben, wobei zuerst das Problem sowie deren Auswirkungen und anschliessend die Lösung präsentiert werden.

Die Web-Oberfläche der NetEm-Box besteht aus zwei Softwareteilen. Im Rahmen der Optimierungen wurden die Versionen davon folgendermassen erhöht:

- NetEmGUI v1.1.3 (vorher v1.1.2)
- PHPnetemGUI v0.12 (vorher v0.12)

6.3.1. Traffic Shaping

6.3.1.1. Problem

Falls über die Web-Oberfläche der NetEm-Box das Traffic Shaping auf beiden Schnittstellen aktiviert wird – sprich der Upload und Download beschränkt werden, bremst die NetEm-Box den Verkehr unverhältnismässig stark. Bei einem Test mit einem Traffic Shaping von 1 Mbit/s, zeigte der cnlab Performance Test [1] bei der Upload-Datenrate nur 51 kbit/s an (siehe Abbildung 39). Die Download-Datenrate lag mit 815 kbit/s nur wenig unter den Erwartungen.

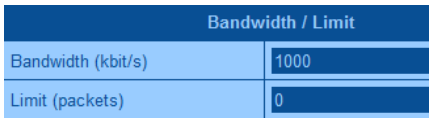


Abbildung 38: Bandbreiten-Einstellung

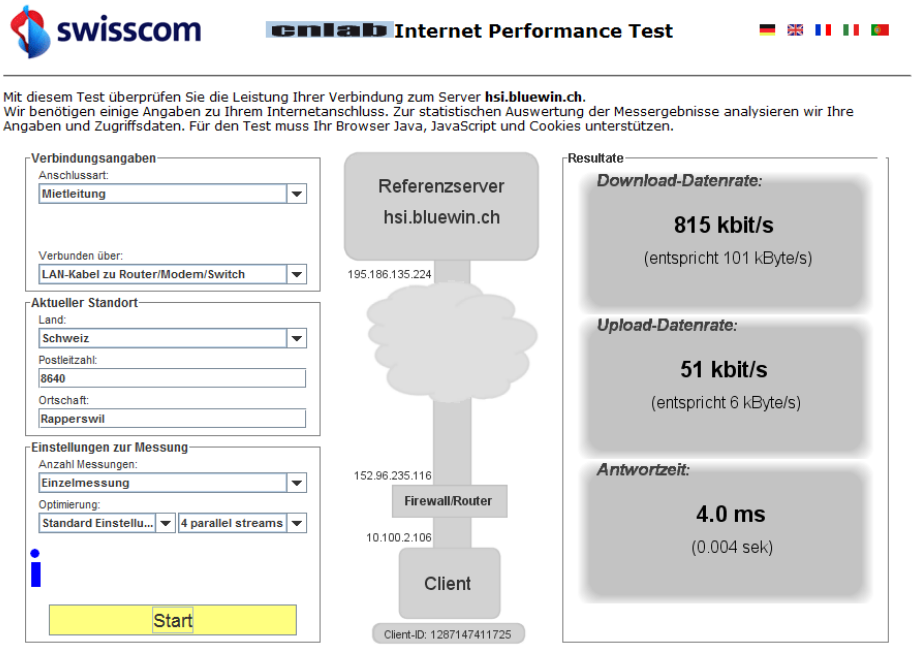


Abbildung 39: cnlab Performance Test, Shaping 1 Mbit/s

Weitere Messungen mit dem cnlab Performance Test zeigen das Problem der NetEm-Box genauer auf:

Beschreibung	Download-Datenrate	Upload-Datenrate	Antwortzeit
Ohne NetEm-Box (direkt)	94’254 kbit/s	95’112 kbit/s	3.4 ms
Mit NetEm-Box, kein Shaping	94’238 kbit/s	95’102 kbit/s	4.0 ms
Mit NetEm-Box, Shaping 1 Mbit/s	815 kbit/s	51 kbit/s	4.0 ms
Mit NetEm-Box, Shaping 10 Mbit/s	8’043 kbit/s	348 kbit/s	4.4 ms
Mit NetEm-Box, Shaping 100 Mbit/s	11’112 kbit/s	839 kbit/s	4.6 ms

Tabelle 9: cnlab Performance Test mit verschiedenem Traffic Shaping

Wie in Tabelle 9 ersichtlich ist, beginnen die Probleme mit den merkwürdig tiefen Datenraten, sobald das Traffic Shaping aktiviert wird. Wireshark-Aufzeichnungen während der Geschwindigkeitsmessung

zeigten einen Paketverlust von bis zu 10%. Dieser hohe Wert lässt die Datenrate einbrechen, da immer wieder Neuübertragungen von TCP Segmenten notwendig sind. Wie sich zeigte, ist der grosse Unterschied zwischen der Upload- und der Download-Datenrate auf die unterschiedliche Anzahl von duplizierten ACKs zurückzuführen (siehe Kapitel 0). Während beim Download der Client bei einem detektierten Paketverlust mehrere duplizierte ACKs versendet, erhält er während eines Uploads bei einem serverseitig detektierten Paketverlust jeweils nur zwei duplizierte ACKs. Dies veranlasst auf dem Client noch keine Neuübertragung des Segmentes und es muss zuerst ein Timeout (das RTO, siehe Kapitel 3.5) abgewartet werden, bis die effektive Neuübertragung beginnt. Dieser Timeout dauert in der Aufzeichnung bis zu 2.7s, wodurch keine wirkliche Datenübertragung zustande kommt. Da die Messung der Datenrate beim cnlab Performance Test nur 5s dauert, erklärt die lange Sendepause die tiefe durchschnittliche Upload-Geschwindigkeit.

859 15.113085	10.100.2.108	195.186.135.224	TCP	64860 [TCP segment of a reassembled PDU]
860 15.121275	195.186.135.224	10.100.2.108	TCP	19320 [TCP Dup ACK 855#1] http > 49201 [PSH, ACK] Seq=1
861 15.133641	195.186.135.224	10.100.2.108	TCP	19320 [TCP Dup ACK 855#2] http > 49201 [PSH, ACK] Seq=1
862 16.014740	10.100.2.108	195.186.135.224	TCP	64860 [TCP Retransmission] [TCP segment of a reassembled
863 16.018533	195.186.135.224	10.100.2.108	TCP	22080 http > 49201 [ACK] Seq=1 Ack=8361 Win=22080 Len=0
864 16.018540	195.186.135.224	10.100.2.108	TCP	24840 http > 49201 [ACK] Seq=1 Ack=9741 Win=24840 Len=0

Abbildung 40: Wireshark-Aufzeichnung mit Paketverlust

Wie bei der Analyse der NetEm-Box festgestellt wurde, ist der hohe Paketverlust auf die Grösse der Qdisc-Buffer zurückzuführen (siehe Abbildung 36). Bei einem hohen Datendurchsatz – wie während der Geschwindigkeitsmessung – mussten durch das Traffic Shaping viele Pakete zwischengespeichert werden, was zu einem Buffer-Überlauf führte. Dieser Überlauf verursacht die Paketverluste.

Die Buffer überliefen so schnell, da diese zu knapp bemessen waren. Die Buffergrösse des Qdisc „netem“ betrug standardmässig 1000 Pakete – sprich rund 1.5 MB – und die vom Qdisc „tbf“ 3000 Bytes. Vor allem die Buffergrösse von 3000 Bytes war viel zu klein gewählt und führte somit zu Paketverlusten.

Es ist zu beachten, dass der „netem“-Buffer über Anzahl Pakete gesteuert wird, der „tbf“ hingegen über Anzahl Bytes [81][82]!

6.3.1.2. Lösung

Für beide Buffer müssen sinnvolle Standardgrössen gefunden werden. Sinnvoll bedeutet, dass der Arbeitsspeicher der NetEm-Box nicht überlastet wird – sprich der Buffer nicht zu gross ist. Auf der anderen Seite darf der Buffer auch nicht zu klein sein, damit die Anwendung des Effektes nicht negativ beeinflusst wird.

Die NetEm-Box verfügt über 1024 MB RAM und die Arbeitspeicherbelastung des Betriebssystems ohne aktivierte Effekte bewegt sich um 75 MB herum.

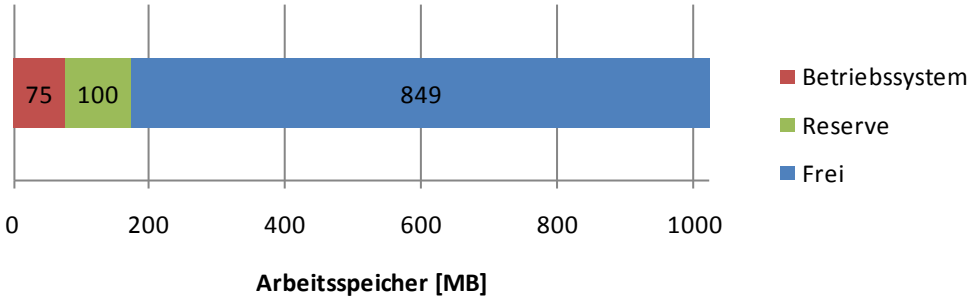


Abbildung 41: Arbeitsspeicherbelastung NetEm-Box ohne Buffer

Die beiden Buffer könnten zusammen somit eine maximale Grösse von 850 MB erreichen, falls für das Betriebssystem noch rund 100 MB Reserve eingerechnet werden. Im Folgenden werden die Buffergrössen theoretisch berechnet, damit die beiden Qdisc korrekt arbeiten können und keine unerwünschten Nebeneffekt entstehen wie die im Kapitel 6.3.1.1 beschriebenen.

Buffergrösse „netem“

Von allen für den Qdisc „netem“ konfigurierbaren Effekten ist die Verzögerung für die Defintion der Buffergrösse am wichtigsten. Die anderen Effekte Paketumordnung, Paketduplizierung und Jitter benötigen nur einen Bruchteil des Buffer, welcher für die korrekte Funktion des Effektes Verzögerung nötig ist. Somit wird die Buffergrösse bei gleicher Ein- und Ausgangs-Bandbreite lediglich durch die Bandbreite und die gewünschte Verzögerung bestimmt.

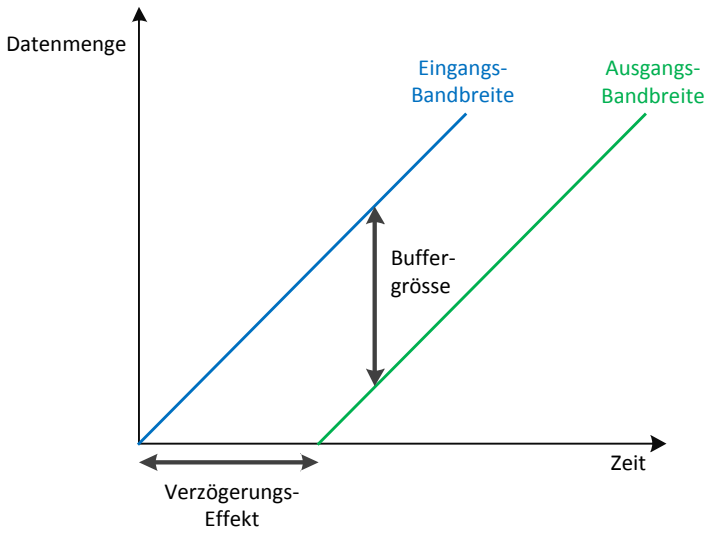


Abbildung 42: Buffergrösse „netem“

Wie in Abbildung 42 ersichtlich ist die benötigte Buffergrösse konstant und hängt alleine von der Bandbreite und der Verzögerung ab. Für die Bestimmung der Buffergrösse muss die Verzögerung und die Bandbreite maximal gewählt werden. Als maximale Verzögerung (RTT) wird 2 Sekunden gewählt, da Werte darüber in der Praxis kaum anzutreffen sind. Die maximale Bandbreite ist die Schnittstellen-Geschwindigkeit der NetEm-Box, welche 100 Mbit/s beträgt. Aus diesen Werten kann

die Buffergrösse für den Qdisc „netem“ ermittelt werden, welche schlussendlich jedoch als Anzahl Pakete konfiguriert werden muss.

$$Buffergrösse\ "netem" = Bandbreite * Verzögerung = 100\ \frac{Mbit}{s} * 2\ s = 200\ Mbit$$
$$= 23.8\ MB = \frac{23.8\ MB}{1518 \frac{Bytes}{Paket}} \approx \mathbf{16'470\ Pakete}\ (bei\ maximaler\ Framegrösse)$$

Formel 2: Buffergrösse „netem“ [83, Folie 5]

Buffergrösse „tbf“

Da der Qdisc „tbf“ nur für das Traffic Shaping verantwortlich ist, sind für dessen Buffergrösse nur die Ein- und die Ausgangs-Bandbreite relevant.

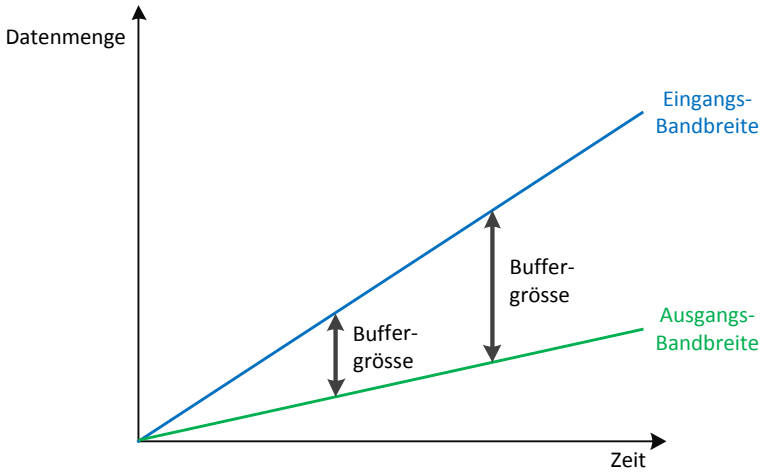


Abbildung 43: Buffergrösse „tbf“

Wie in Abbildung 43 ersichtlich ist, steigt die Buffergrösse mit voranschreitender Zeit proportional zum Bandbreiteunterschied an, falls die Ausgangs-Bandbreite stets kleiner als die Eingangs-Bandbreite ist. Die Eingangs-Bandbreite entspricht wieder der Schnittstellen-Geschwindigkeit der NetEm-Box, welche 100 Mbit/s beträgt. Unter der Ausgangs-Bandbreite ist das effektiv konfigurierte Traffic Shaping zu verstehen.

Da der „tbf“-Buffer aber nicht unendlich gross gewählt werden kann, müssen andere Gegenbenheiten hinzugezogen werden für dessen Berechnung. Bei TCP-Verbindungen bestimmt die Grösse des RWin (Receive Window, siehe Kapitel 3.4), wie gross die Datenmenge sein darf, welche maximal in das Netzwerk gesendet werden darf, ohne bestätigt worden zu sein. Somit muss im „tbf“-Buffer maximal eine RWin pro TCP-Verbindung zwischengespeichert werden.

Die Herleitung des maximalen RWin gestaltet sich nicht so einfach. Mit einem aktiven Window Scaling (siehe Kapitel 3.4) kann das RWin bis zu 1 GB betragen. In der Praxis ist ein so grosses RWin jedoch kaum anzutreffen. Beispielsweise skaliert das TCP Auto-Tuning von Windows 7 das RWin nur bis maximal 16 MB. Gemäss Bandbreite-Delay-Produkt entspricht dies einer Verzögerung von 1.34s bei einer Bandbreite von 100 Mbit/s, wenn das RWin nicht die Datenübertragung ausbremsen soll.

Für die Berechnung eines genügend grossen Buffers muss jedoch von mehreren parallelen Verbindungen ausgegangen werden. Gemäss [84] sollte der Buffer lieber zu gross als zu klein gewählt werden. Eine Buffergrösse, die für 8 parallele Verbindungen mit je maximalem RWin (16 MB unter Windows 7) ausreicht, sollte für beinahe alle Anwendungsfälle genügend gross sein.

$$Buffergrösse\ "tbf" = 8 * max.RWin = 8 * 16\ MB = 128\ MB$$

Formel 3: Buffergrösse „tbf“

Die Buffergrösse des „tbf“ lässt sich im Gegensatz zum „netem“ nur über Anzahl Bytes konfigurieren, nicht über die Anzahl Pakete!

Arbeitsspeicherbelastung durch die Buffer

Die beiden Qdisc „netem“ und „tbf“ müssen für jede Schnittstelle der NetEm-Box je einmal konfiguriert werden, sprich einmal für den Download und einmal für den Upload. Dies ergibt insgesamt folgende Arbeitsspeicherbelastung der NetEm-Box:

$$Arbeitsspeicherbelastung\ Emulator = 2 * netem + 2 * tbf$$
$$= 2 * 23.8\ MB + 2 * 128\ MB = \mathbf{303.6\ MB}$$

Formel 4: Arbeitsspeicherbelastung Emulator bei NetEm-Box

Diese Buffergrössen stellen für den Arbeitsspeicher eine gerine Belastung dar. Insgesamt stehen dannach immer noch rund 545 MB zur freien Verfügung (siehe Abbildung 44).

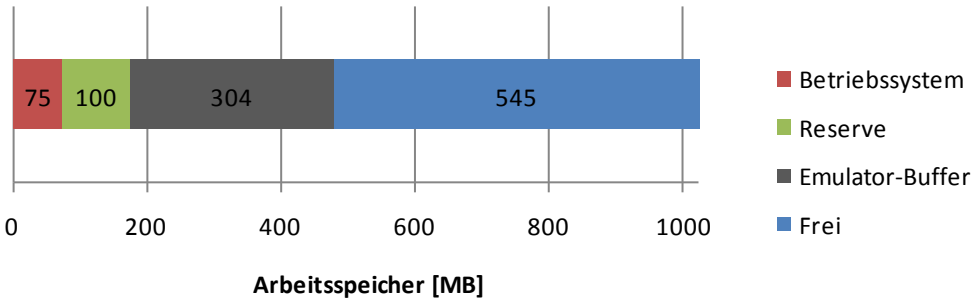


Abbildung 44: Arbeitsspeicherbelastung NetEm-Box mit Buffer

Bufferkonfiguration via Weboberfläche

Früher konnte über die Weboberfläche die Buffergrösse der „netem“ Qdisc konfiguriert werden (Eingabefeld „Limit (packets)“ im Abschnitt „Bandwith/Limit“, siehe Abbildung 38). Die Konfiguration des „tbf“ Buffers ist jedoch von grösserem Nutzen. Denn damit können Situationen emuliert werden, in welchen ein Router über zu wenig Speicher verfügt, da er für das Datenverkehrsaufkommen im Netzwerk unterdimensioniert ist. Die Folge davon ist, dass sein Buffer überläuft und er Pakete verwerfen muss. Dieses Problem besteht ab und zu [85].

Aus diesen Gründen wurde die Weboberfläche angepasst, damit neu die Buffergrösse der „tbf“ Qdisc verändert werden kann. Sie kann über das Eingabefeld „Limit (KB)“ eingestellt werden (siehe Abbildung 45).

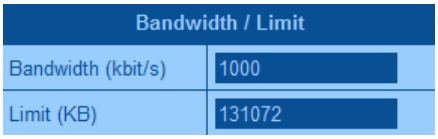


Abbildung 45: Bandbreiten-Einstellung „tbf“

6.3.1.3. Kontrolle

Um die Lösung zu verifizieren, wurden die selben Messungen mit dem cnlab Performance Test [1] nochmals durchgeführt.

Beschreibung	Download-Datenrate	Upload-Datenrate	Antwortzeit
Ohne NetEm-Box (direkt)	94’268 kbit/s	95’123 kbit/s	3.5 ms
Mit NetEm-Box, kein Shaping	94’241 kbit/s	95’111 kbit/s	4.0 ms
Mit NetEm-Box, Shaping 1 Mbit/s	936 kbit/s	967 kbit/s	4.2 ms
Mit NetEm-Box, Shaping 10 Mbit/s	9‘465 kbit/s	9‘656 kbit/s	4.4 ms
Mit NetEm-Box, Shaping 100 Mbit/s	94’157 kbit/s	94‘959 kbit/s	4.3 ms

Tabelle 10: cnlab Performance Test mit verschiedenem Traffic Shaping

Wie in Tabelle 10 ersichtlich ist, funktioniert das Traffic Shaping nun wie gewünscht. Mit allen gewählten Traffic Shapings wurde eine akzeptable effektive Datenrate erreicht.

Um die Abweichung des eingestellten Traffic Shaping zur effektiven erreichten Datenraten zu kontrollieren, wurde eine Reihe von Messungen durchgeführt und die zwei Werte verglichen. Wie aus Abbildung 46 herauszulesen ist, steigt die Abweichung mit zunehmender Datenrate an, bewegt sich prozentual aber im selben Bereich. Im Durchschnitt konnte eine Übereinstimmung von 95.3% festgestellt werden. Die Abweichung von 100% ist darauf zurückzuführen, dass das Traffic Shaping für Layer 2 (Ethernet) gilt [86], der cnlab Performance Test [1] jedoch die Datenrate auf der Stufe HTTP misst. Weiter ist in der Abbildung ein Knick bei 94 Mbit/s zu bemerken, da dies die maximale Bandbreite der verwendeten Internetverbindung darstellt.

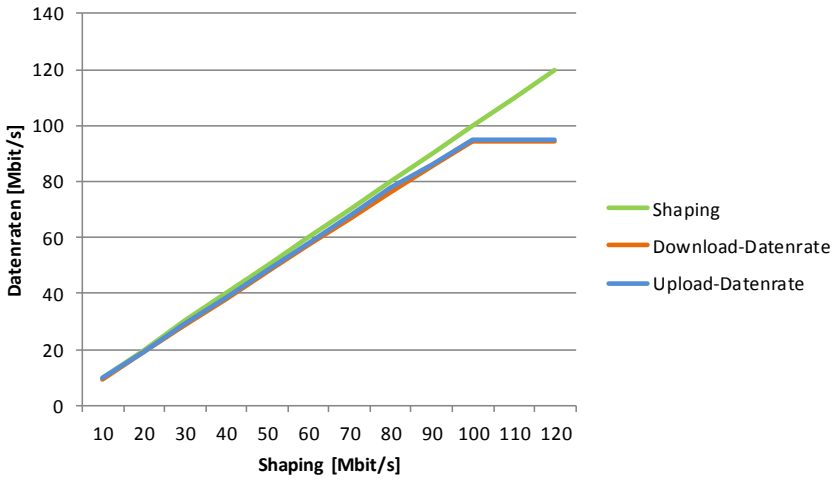


Abbildung 46: Abweichung eingestelltes Traffic Shaping zur effektiv erreichten Datenrate

Um zu beweisen, dass die NetEm-Box auch den UDP-Verkehr wie gewünscht beeinflusst, wurden einige Tests mit JPerf [87] durchgeführt.

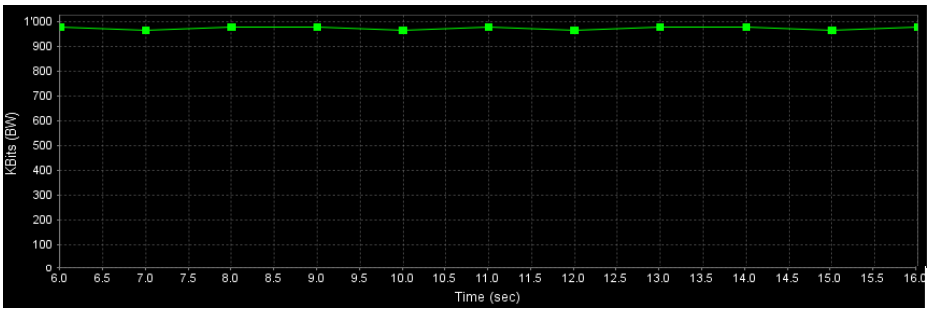


Abbildung 47: JPerf-Test UDP, Shaping 1 Mbit/s

Wie in
Abbildung 47 ersichtlich ist, hält JPerf die UDP-Geschwindigkeit konstant leicht unter 1 Mbit/s. Dieses Ergebnis entspricht vollständig den Erwartungen. Da UDP im Gegensatz zu TCP kein Bandbreiten-Management mitbringt, ist es ein leichtes Unterfangen den NetEm-Buffer zu überfüllen und damit eine hohe Paketverlustrate zu erreichen. Im gleichen Test wurde die NetEm-Box auch noch im Zusammenhang mit TCP getestet.

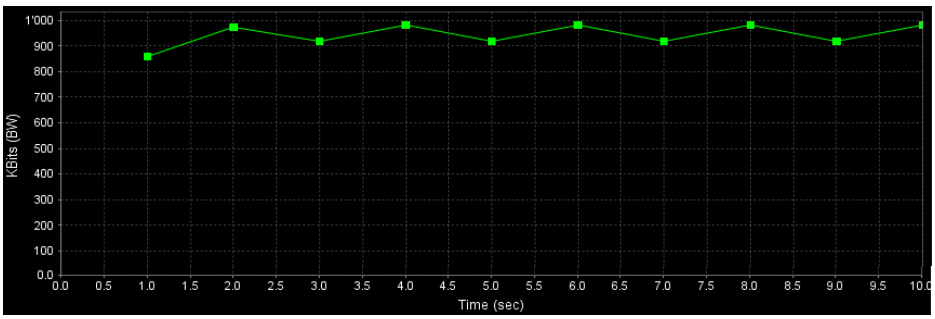


Abbildung 48: JPerf-Test TCP, Shaping 1 Mbit/s

Auch diese Ergebnisse der Messungen entsprachen den Erwartungen und decken sich mit den Ergebnissen des cnlab Performance Tests.

Abschliessend kann gesagt werden, dass die Änderung des Traffic Shapings der NetEm-Box die gewünschten Verbesserungen gebracht hat und nun auch den Anforderungen in der Praxis Stand hält!

6.3.2. DNS-Server per DHCP

6.3.2.1. Problem

In der NetEm-Box kann bei Bedarf – für die Verteilung von IP-Adressen – ein DHCP-Server aktiviert werden. Normalerweise wird dieser auf der Schnittstelle aktiviert, an welcher die Test-Clients angeschlossen sind. Für diese Clients kann dann via DHCP die IP-Adresse des Standard-Gateways auf die IP der NetEm-Box gesetzt werden.

Die Weboberfläche der Box unterstützt aber nur die Konfiguration der folgenden DHCP Parameter: IP-Bereich, Subnetzmaske und Standard-Gateway. Leider fehlt die DNS-Server-Option. Somit müssen auf allen per DHCP konfigurierten Clients manuell die DNS-Server nachgetragen werden. Dies ist bei mehreren Computern mit einem grossen Aufwand verbunden.

6.3.2.2. Lösung

In der Konfiguration der NetEm-Box wurde im Abschnitt „DHCP subnet configuration“ der Punkt „Nameserver (DNS) for this subnet“ hinzugefügt. Die dort angegeben DNS-Server werden nun auch via DHCP verteilt. Dies erspart die manuelle Konfiguration der Clients.

DHCP subnet configuration LAN B (eth1)	
Subnet	10.100.2.0
Netmask	255.255.255.0
Router for this subnet	10.100.2.1
Mask for this subnet	255.255.255.0
IP range start for this subnet	10.100.2.100
IP range stop for this subnet	10.100.2.250
Nameservers (DNS) for this subnet (up to 2 servers)	152.96.20.10 152.96.21.10

Abbildung 49: DHCP-Konfiguration mit DNS-Server

6.3.3. Navigation

6.3.3.1. Problem

Die Benutzerführung in der Weboberfläche der NetEm-Box ist nicht besonders intuitiv. Dies kommt zum einen daher, dass für die allgemeine Konfiguration der NetEm-Box wie IP-Adresse, DHCP-Server usw. und für die Konfiguration der Effekte wie Bandbreite, Verzögerung etc. zwei Weboberflächen in

eine integriert wurden. Jedoch wurden die beiden unterschiedlichen Navigationskonzepte beibehalten, was auf den ersten Blick verwirrend ist.

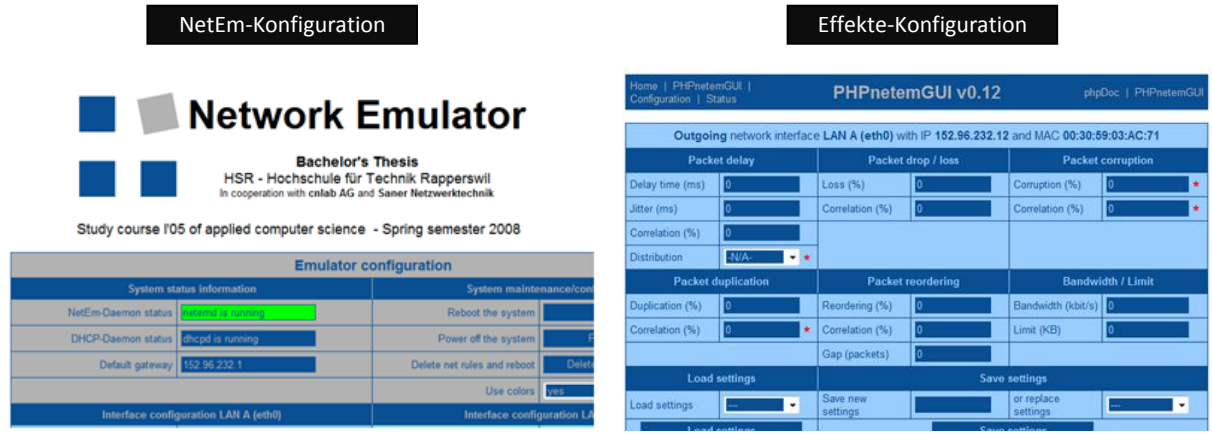


Abbildung 50: Unterschiedliche Weboberflächen

Zum anderen sind auch die Farben der Buttons und Textfelder unglücklich gewählt. Eine Unterscheidung, wo geklickt und wo etwas eingegeben werden kann, ist optisch nicht erkennbar.

Mühsam ist auch die Zwischenseite, welche erscheint, sobald auf der Startseite auf „PHPnetemGUI“ geklickt wird. Auf dieser Zwischenseite wird darauf hingewiesen, dass zurzeit nur der „Basic Mode“ implementiert ist. Mit einem weiteren Klick erreicht man dann die eigentliche Konfiguration des Netzwerk-Emulators, die Konfiguration der Effekte.

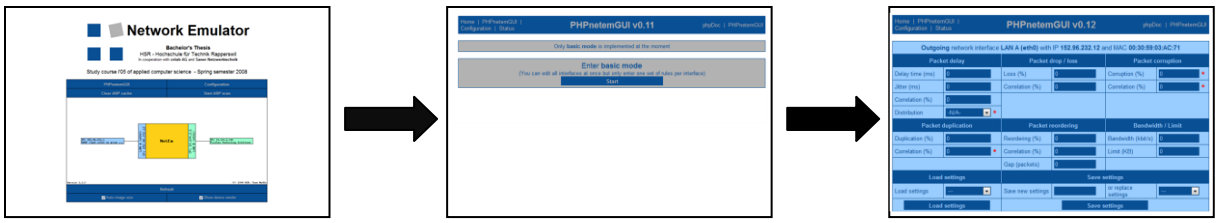


Abbildung 51: Umständliche Navigation mit Zwischenseite

6.3.3.2. Lösung

Für die ganze Web-Oberfläche der NetEm-Box müsste ein durchgängiges Navigationskonzept erstellt und umgesetzt werden. Auch die Gestaltung der Buttons sollte überarbeitet werden. Der Aufwand dafür ist jedoch zu gross, um im Rahmen dieser Arbeit zusätzlich durchgeführt zu werden.

Da die Entfernung der vorher genannten Zwischenseite sehr einfach war, wurde diese entfernt und der Benutzer gelangt nun von der Startseite mit einem Klick direkt in die Konfiguration des „PHPnetemGUI“.

6.4. Verwendung

Die Bedienung der Web-Oberfläche der NetEm-Box ist ausführlich im Benutzerhandbuch dokumentiert [78], Netzwerk-Emulator Benutzerhandbuch]. An dieser Stelle werden nur die Konfigurationsoptionen aufgezeigt, welche nicht selbsterklärend sind.

Falls auf der Startseite auf „PHPnetemGUI“ geklickt wird, erscheint die in Abbildung 52 gezeigte Ansicht. In dieser Ansicht lassen sich alle Effekte der NetEm-Box konfigurieren.

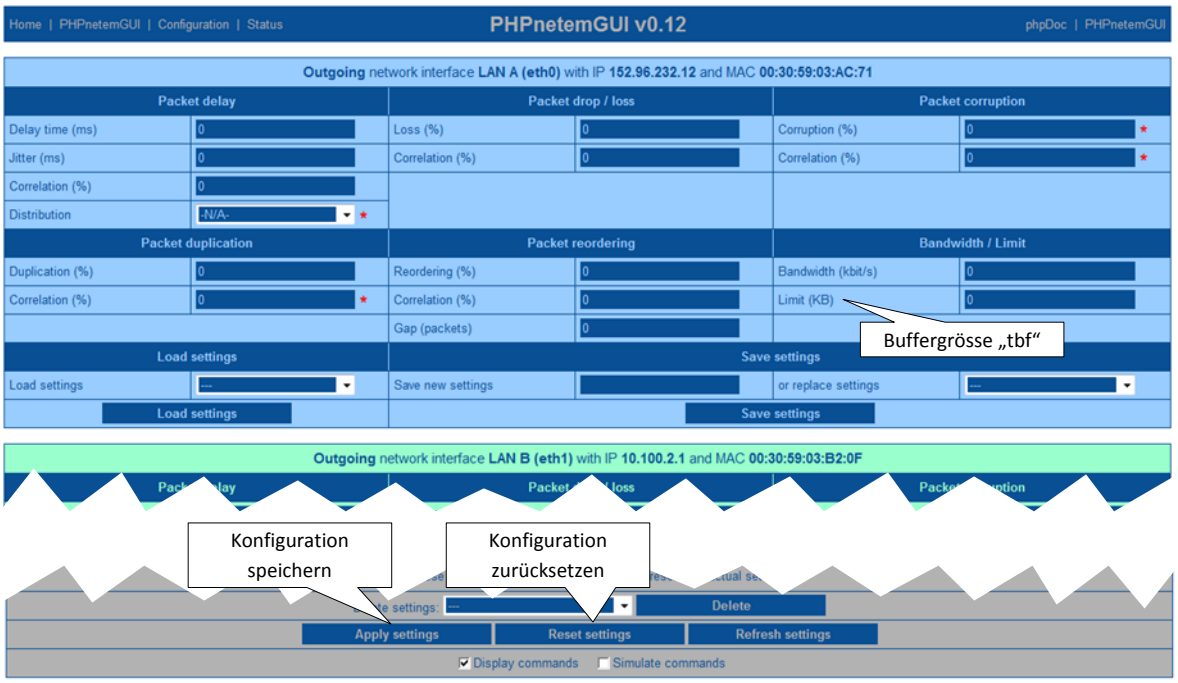


Abbildung 52: Konfiguration des PHPnetemGUI

Die meisten konfigurierbaren Optionen sind selbsterklärend. Unter „Bandwidth“ kann bei „Limit (KB)“ die Buffergrösse des Qdisc „tbf“ eingestellt werden (siehe Kapitel 6.3.1.2, Buffergrösse „tbf“). Damit lässt sich mit der NetEm-Box ein Router simulieren, welcher einen zu kleinen Buffer besitzt und es dadurch zu einer hohen Paketverlustrate kommt. Speziell beachtet werden muss, dass der obere hellblaue Teil der Konfiguration für den abgehenden Verkehr der Schnittstelle „LAN A“ und der untere hellgrüne Teil für den abgehenden Verkehr der Schnittstelle „LAN B“ gilt. Um die Konfiguration zu speichern, muss ganz unten in der Webseite auf den Button „Apply settings“ geklickt werden und nicht auf den pro Schnittstelle vorhandenen Button „Save settings“. Die Buttons „Save settings“ speichern die Schnittstellen-Konfiguration nur in einer Datei, damit diese spezifische Konfiguration später nochmals abgerufen werden kann.

7. Messungen

Dieses Kapitel informiert genauer über die Messungen der Webseiten-Aufrufe unter verschiedenen Browsern und Betriebssystemen. Hier werden die emulierten Netzwerk-Leistungsparameter, die untersuchten Webseiten sowie der genaue Messaufbau und -ablauf beschrieben.

7.1. Netzwerkparameter

Mit der NetEm-Box können eine Vielzahl von Netzwerkparametern beeinflusst werden (siehe Kapitel 6.1). Diese Arbeit beschränkt sich jedoch auf die Variation der folgenden Parameter:

- Bandbreite
- Verzögerung

Die typischen Internetabos der verschiedenen ISPs (Internet Service Provider) unterscheiden sich vor allem durch diese Werte. Der Heimbenutzer kauft das Abo primär aufgrund der versprochenen Bandbreite, dabei ist ihm die Rolle der Verzögerung meistens nicht bewusst. Je nachdem welche Peering-Verträge der eigene ISP mit anderen ISPs abgeschlossen hat, erfolgt das Routing des Datenverkehrs über andere Pfade und dementsprechend gibt es Unterschiede bei der RTT. Ein ISP kann durch Routing-Optimierungen die RTT ebenfalls beeinflussen.

Die anderen Werte wurden aus zeitlichen Gründen nicht berücksichtigt. Dies aus dem Grund, da der Messaufwand exponentiell mit der Anzahl Parameter ansteigt. Die Variation der Netzwerkparameter wird schliesslich unter zwei Betriebssystemen und mit drei Browsern untersucht. Eine Automatisierung der Messungen ist schwierig, da die Zeitpunkte „Erste Anzeige“ und „Seite benutzbar“ nur mit grossem Aufwand automatisch erfasst werden können und deshalb von Hand (mit einer Stoppuhr) gemessen werden müssen.

Bandbreite

Gemäss einer repräsentativen Studie [88] verfügten im Jahr 2007 nur noch 6% der Schweizer Bevölkerung über einen Dial-up Internetzugang. Heute dürfte dieser Wert noch weiter gesunken sein. Deshalb werden nur Geschwindigkeiten von Breitband-Angeboten (xDSL sowie Cable-Abos) in diesem Test emuliert. Dabei wird das Endkundenerlebnis bei einem langsamen (1 Mbit/s) und einem schnellen Abo (20 Mbit/s) genauer untersucht.

Verzögerung

Gemäss Auswertungen des cnlab Performance Tests liegen die RTT-Werte innerhalb Europa unterhalb von 60ms. In der Schweiz werden nur selten Werte über 30ms erreicht. Durch die Variation der Verzögerung wird emuliert, wie Benutzer aus Island (RTT ca. 59ms) oder Japan (RTT ca. 270ms) den Aufruf einer Schweizer Webseite erleben, wenn der Webseiten-Anbieter kein CDN (Content Distribution Network) wie z.B. Akamai nutzt. Surft ein Schweizer Benutzer eine Webseite an, welche in den USA angeboten wird, so hat er ein ähnliches Erlebnis (RTT ca. 120ms).

Falls ein Kunde via Satellit ans Internet angebunden ist, da sonst keine andere Technologie in seiner Nähe zur Verfügung steht oder der Ausbau derjenigen zu teuer käme, muss er mit einer noch höhe-

ren RTT von ca. 600ms rechnen. In der Schweiz sind nur sehr wenige Kunden so angebunden, gemäss cnlab weniger als 1%. In den USA mit den grossen Distanzen und der zerstreuten Besiedlung, wird diese Technologie jedoch öfters eingesetzt.

Für die Messungen wurden folgende Variationen der Netzwerkparameter verwendet:

Netzwerkparameter	**Ausprägungen**
Bandbreite [kbit/s] **Download/Upload**	1‘000 / 100 20‘000 / 2‘000
RTT [ms]	15 60 240 600

Tabelle 11: Netzwerkparameter und deren Variation

Tests mit Zwischenzeiten wie 30 oder 120ms wurden nicht durchgeführt, da erste Messungen mit 15 und 60ms kaum einen Unterschied zeigten. Deshalb und um den Messaufwand zu reduzieren, wurde entschieden, die Tests mit stärker variierenden RTTs auszuführen.

7.2. Getestete Webseiten

Für die zu testenden Webseiten wurden Seiten ausgewählt, welche in der Schweiz sehr bekannt sind und auch oft besucht werden (siehe Tabelle 12). Dabei wurde auch die persönliche Erfahrung der Autoren berücksichtigt.

Eine grobe Charakterisierung der Webseiten ist ebenfalls ersichtlich in dieser Tabelle. Da in dieser Arbeit der Fokus auf den Auswirkungen der verschiedenen Netzwerkparameter und nicht auf dem Vergleich von Webseiten liegt, wird z.B. www.sbb.ch stellvertretend für gleichartige Webseiten wie www.google.ch getestet und die Resultate von www.20min.ch sprechen auch für www.tages-anzeiger.ch. Gleichartig bezieht sich hier auf die Charakterisierung in der folgenden Tabelle.

Webseite	**Alexa Ranking** **Top 100 Schweiz [91]**	**Charakterisierung**
sbb.ch	19	Inhalt ändert selten wenige Objekte kleine Datenmenge
20min.ch	10	Inhalt ändert oft sehr viele Objekte grosse Datenmenge

Tabelle 12: Zu testende Webseiten

Die SBB-Webseite enthält im Gegensatz zur 20min-Webseite wenige Objekte und ist relativ schnell geladen. Dies spiegelt sich auch in der zu übertragenden Datenmenge wieder, wie im Kapitel 7.2.4 aufgezeigt wird.

Es wurden bewusst Webseiten mit stark unterschiedlichem Aufbau gewählt, da der Charakter der Datenübertragung stark davon abhängt. Die Auswirkungen einer hohen RTT dürften bei einer Webseite mit vielen Objekten wesentlich stärker spürbar sein, als wenn nur wenige Dateien übertragen werden müssen.

Bei den Messungen wurde nicht beachtet, dass pro Browser eventuell ein anderer Inhalt ausgeliefert wird. So wird z.B. bei Verwendung des Internet Explorers beim Aufruf von google.ch eine Werbung für Chrome angezeigt, bei Verwendung von Firefox oder Chrome hingegen nicht. Ebenso wird z.B. bei der Entwicklung von Webapplikationen mit dem Google Web Toolkit (GWT) für jeden Browser eine eigene JavaScript Datei erstellt.

7.2.1. Analyse allgemein

In den folgenden drei Kapiteln wird der grobe Aufbau der Webseiten analysiert. Dabei geht es vor allem um die Anzahl Objekte, die Datenmenge und die Anzahl Hosts. Für die Analyse wurden die Resultate der Tools HttpWatch und dynaTrace verwendet.

Mit dem dynaTrace-Plugin für den Internet Explorer wurde die Seite aufgerufen und anhand der Angaben im Tab „KPIs“ des entsprechenden Performance Reports wurden Grafiken zum Seitenaufbau erstellt. Bei HttpWatch diente der Tab „Network“ der „Page Summary“ als Datenquelle.

Die gemessenen übertragenden Datenmengen stimmen bei HttpWatch und dynaTrace nicht ganz überein, wobei der Wert bei dynaTrace ca. 7% grösser ist. Der Unterschied konnte nicht erklärt werden, ist jedoch für die Darstellung in den folgenden Kapiteln nicht relevant, da es bei der Visualisierung primär um das Verhältnis geht und nicht um genaue Zahlen. Die Anzahl Objekte variiert zw. HttpWatch und dynaTrace, da bei HttpWatch das Favicon separat betrachtet wird und bei dynaTrace nicht. Beim Favicon handelt es sich jedoch um sehr wenige Daten und kann die Feststellung im oberen Abschnitt nicht erklären.

Für die Auswertung der MIME-Typen wurden alle Bildtypen sowie alle JavaScript Dateien zusammengefasst dargestellt.

Während den Messungen hat sich der Inhalt beider Webseiten etwas verändert. Dazu sind noch andere Schwankungen bzw. Ungenauigkeiten gekommen, welche im Kapitel 9.8.6 zusammengefasst betrachtet werden. Die Schwankungen durch die Veränderung der Webseiten spiegeln sich gemäss der dortigen Ausführung bereits in den unterschiedlichen Messwerten wieder und muss nicht noch separat einberechnet werden. Der Vollständigkeit halber wurde in den folgenden Kapiteln das Ausmass der Veränderung bei den Webseiten trotzdem festgehalten.

Die nachfolgenden Auswertungen bzgl. der Datenmengen und Objekte pro Webseite wurden nur für eine spezielle Messung gemacht, da es primär um eine Veranschaulichung der Verhältnisse geht und nicht um exakte Werte. Deshalb reicht die Auswertung in dieser Form.

7.2.2. Analyse sbb.ch

Die SBB Webseite ist eine ziemlich schlanke Webseite. Es werden 19 Objekte und total 133 kB Daten abgerufen.

Nachfolgend sind auf den beiden Screenshots die Events „Erste Anzeige“ und „Seite benutzbar“ für die SBB Webseite dargestellt. „Seite benutzbar“ gilt, wenn das Navigationsmenü sowie das Formular für die Fahrplanabfrage geladen wurden.

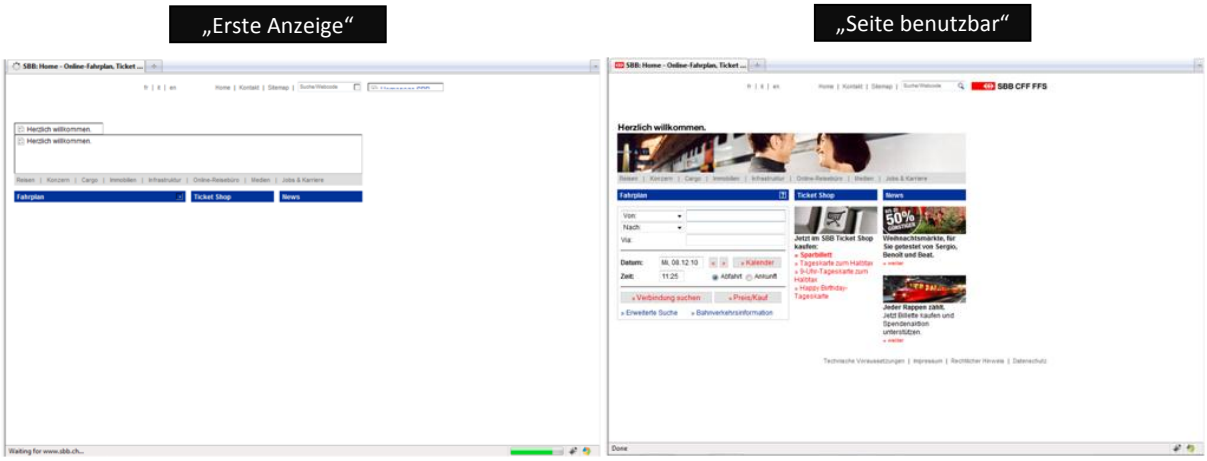


Abbildung 53: "Erste Anzeige" und "Seite benutzbar" für sbb.ch

Wie in der untenstehenden Grafik zu erkennen ist, handelt es sich bei über der Hälfte der Objekte um Bilder. Betrachtet man jedoch die Datenmenge, so machen Bilder und JavaScript den grössten Anteil aus, anschliessend folgt die HTML Seite.

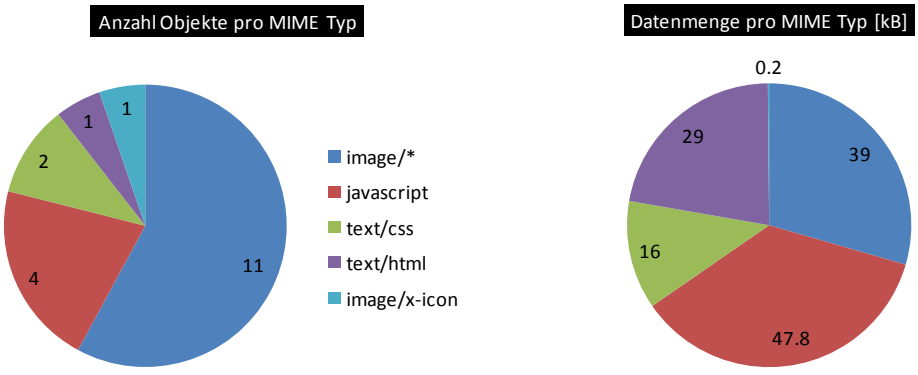


Abbildung 54: Auswertung von sbb.ch bzgl. MIME Typ

Für die Darstellung der Webseite werden Daten von zwei Hosts heruntergeladen. Der grösste Teil kommt von der sbb.ch Domain und der Rest stammt von Google Analytics. Interessant ist, dass der Google Analytics-Teil, welcher primär aus einer JavaScript-Datei besteht, doch 24 kB gross ist und daher bei einer schlanken Seite wie sbb.ch doch 18% der Daten ausmacht. Dieser Inhalt wird für Benutzerstatistiken benötigt.

Da pro Host eine DNS-Abfrage nötig ist, braucht es für den Abruf dieser Webseite zwei DNS Abfragen.

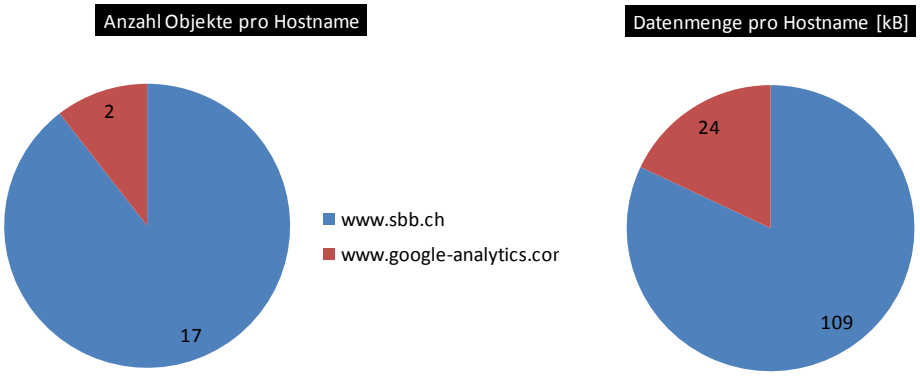


Abbildung 55: Auswertung von sbb.ch bzgl. Hosts

Während den Tests gab es nur geringe Änderungen an der Webseite. Diese waren nicht direkt sichtbar, sondern wurden erst bei der Analyse entdeckt. Es handelt sich jedoch nur um minime Änderungen: Die Datenmenge sank um 4 kB bzw. 3% und die Anzahl Objekte um zwei Stück bzw. um 10.

Alle Messungen mit der RTT 15 und 60ms fanden vor und jene mit 240 und 600ms nach der Änderungen statt. Die Änderung der Datenmenge ist zu vernachlässigen und bei der Anzahl Objekte sind die Auswirkungen ebenfalls nicht relevant. Denn die höhere Anzahl Objekte galt bei den Messungen mit den tiefe RTTs (15 und 60ms), weshalb die zusätzlichen Anfragen die gemessenen Zeiten nur minim beeinflussten.

7.2.3. Analyse 20min.ch

Die 20min.ch Webseite umfasst sehr viele Objekte. Total werden 276 Objekte und 2455 kB an Daten übertragen.

Nachfolgend sind auf den beiden Screenshots die Events „Erste Anzeige“ und „Seite benutzbar“ für die 20min Webseite dargestellt. „Seite benutzbar“ gilt, wenn beim ersten Beitrag oben links der ganze Text sowie beim Bild mehr als die Hälfte geladen wurden.

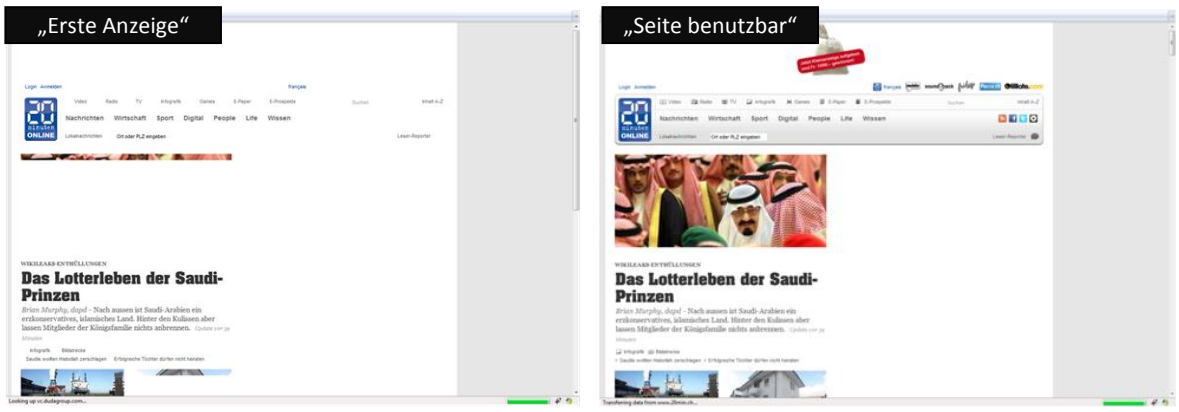


Abbildung 56: "Erste Anzeige" und "Seite benutzbar" für 20min.ch

Bei dieser Seite ist der Bilderanteil enorm, fast 92% aller Objekte sind Bilder. Bei der Datenmenge sind es immer noch 64%. Dort erscheint an zweiter Stelle auch das JavaScript.

Zum einen werden viele Bilder eingesetzt auf dieser Seite, zum anderen ist die Webseite aber auch sehr lange (siehe nächstes Kapitel). Es werden sehr viele Beiträge aufgelistet, so dass der Benutzer sehr lange scrollen kann, bis er das Ende der Seite erreicht.

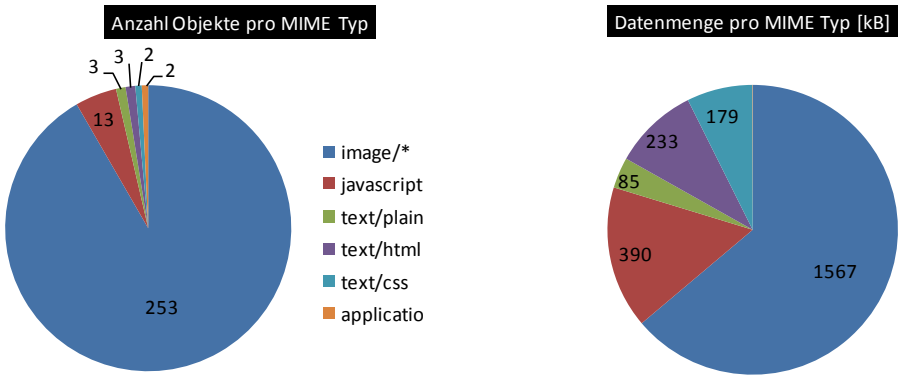


Abbildung 57: Auswertung von 20min.ch bzgl. MIME Typ

Beim Aufruf der 20min.ch Webseite werden Daten von ca. 9 Hosts geladen. Diese Zahl ändert sich je nach angezeigtem Inhalt. Der grösste Teil des Inhalts stammt von einer 20min Domain, doch werden einige Elemente für Werbung und Benutzerstatistiken von anderen Hosts geladen. Ein Teil der angezeigten Werbung wird jedoch auch direkt auf 20min.ch bereitgestellt, wobei diese oft nicht so schnell als Werbung zu erkennen ist. Es werden nur wenige Daten von fremden Hosts abgerufen (kein 20min-Domain).

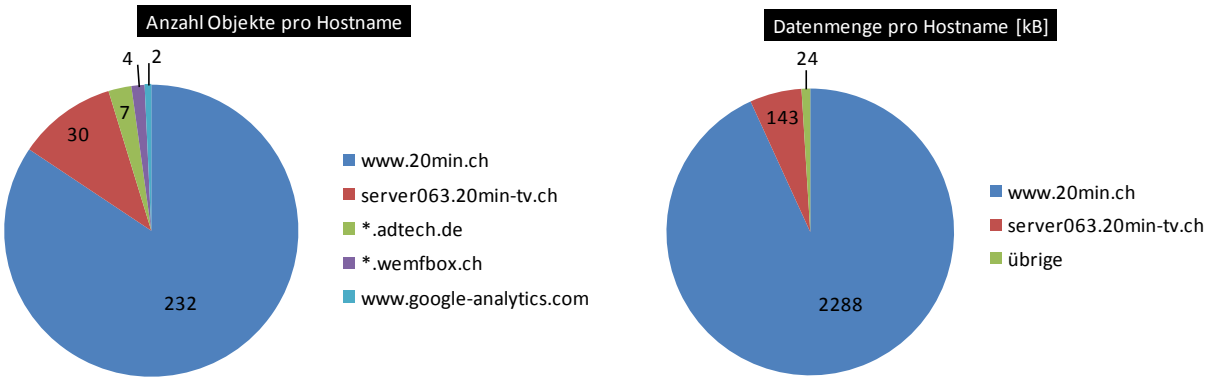


Abbildung 58: Auswertung von 20min.ch bzgl. Hosts

Die Webseite 20min ist doch grösseren inhaltlichen Schwankungen unterworfen, als zuerst angenommen wurde. Bei der Analyse zeigten sich folgende Schwankungen über die gemessenen Werte:

	Mittelwert	**Standardabweichung**
Datenmenge	2454 kB	116 kB (5%)
Anz. Objekte	285	6 (2%)

Abbildung 59: Ausmass der Schwankungen bei 20min.ch

Bzgl. der Anzahl Objekte ist eine Ungenauigkeit von ca. einer RTTs gegeben (6 Objekte bei 6 parallelen Verbindungen ergibt eine Anfrage pro Verbindung). Bzgl. der Datenmenge ist bei einer Bandbreite von 1000 kbit/s eine Ungenauigkeit von ca. 0.16s angebracht (ca. 116 kB Differenz auf sechs Verbindungen verteilt bei 125 kB/s Ladegeschwindigkeit). Diese beiden Werte müssen addiert werden. Bei der Bandbreite von 20‘000 kbit/s ist der zweite Anteil tiefer, noch ca. 0.01s.

Total ergibt sich also eine Ungenauigkeit von 0.01s/0.16s + 1 RTTs. Dabei ist das Slow Start-Phänomen nicht berücksichtigt worden. Die Tabelle 13 zeigt die absoluten Ungenauigkeiten bei verschiedenen Netzwerkparametern.

	RTT 15ms	**RTT 60ms**	**RTT 240ms**	**RTT 600ms**
1‘000 kbit/s	0.18	0.22	0.40	0.76
20‘000 kbit/s	0.03	0.07	0.25	0.61

Tabelle 13: Ungenauigkeit der 20min.ch Messungen in Sekunden

Gemäss den Ausführungen im Kapitel 9.8.6 wurde die Ungenauigkeit für die Messungen der 20min.ch Webseite berechnet (die konkreten Berechnungen dazu sind auf der CD zu finden). Dabei kam ein Wert von 0.41s über alle Messungen der 20min.ch Seite zustande, was ungefähr mit den Werten der obenstehenden Tabelle übereinstimmt.

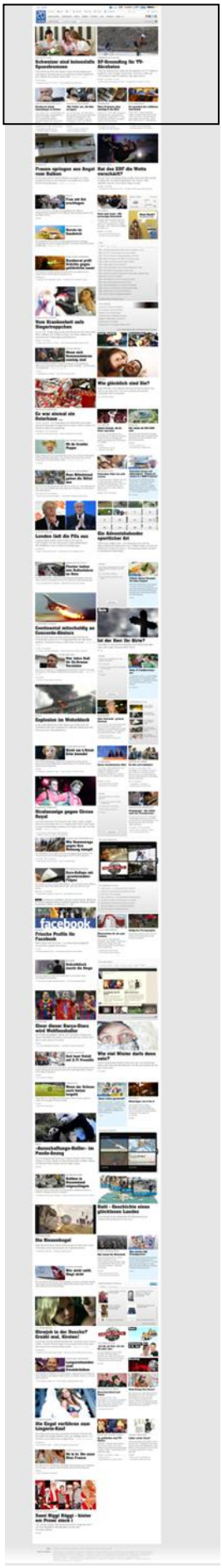


Abbildung 61: Vergleich der Screenshots der gesamten Webseite, 20min.ch (L) vs. sbb.ch (R)

7.2.4. Vergleich 20min.ch mit sbb.ch

Wie in der nebenstehenden vertikalen Grafik zu erkennen ist, ist der Unterschied in der Länge der Webseite enorm zwischen 20min.ch und sbb.ch. Bei einer Bildschirmauflösung von 1680x1050 Pixeln ist die 20min.ch Seite ca. 13x länger. Dies erklärt auch, warum beim Aufruf der 20min.ch Seite viel mehr Daten übertragen werden. Es werden dazu noch viel mehr Bilder eingesetzt als bei sbb.ch. Dies macht die Seite sehr bunt.

Beim Aufruf der Webseite 20min.ch werden wie beim Aufruf von sbb.ch nur 24 kB Daten von fremden Hosts geladen. Erstaunlich ist, dass es bei einer so grossen Seite wie 20min nicht mehr Daten sind. Es ist jedoch anzumerken, dass für die Erfassung von Benutzerstatistiken nicht viel Code benötigt wird.

Nachfolgend ein Vergleich der beiden Webseiten in Zahlen, um die Verhältnisse klarer darzustellen. Dabei handelt es sich um die Darstellung einer spezifischen Messung. Bei 20min.ch gibt es einige Schwankungen wie im vorherigen Kapitel erwähnt.

	sbb.ch	20min.ch
Datenmenge	133	2455
Anzahl Objekte	19	276
Anzahl Hosts	2	9
DNS Abfragen	2	9-17

Abbildung 60: Vergleich des Seitenaufbaus von sbb.ch und 20min.ch

Die Anzahl DNS-Abfragen bei 20min.ch schwanken, da Firefox und Chrome ein DNS Prefetching machen und der Internet Explorer nicht. Weitere Informationen zu dieser Funktion sind im Kapitel 7.3.5 zu finden.

7.3. Resultate

Es ist wichtig anzumerken, dass die hier präsentierten Resultate sich nur auf die Ladegeschwindigkeit der Browser der Webseiten www.sbb.ch sowie www.20min.ch beziehen. Da nur diese zwei Seiten getestet wurden und bei der Auswertung der Hauptfokus auf der Zeit „Seite benutzbar“ lag, kann keine pauschale Aussage gemacht werden. Der Aufbau einer Webseite hat einen sehr grossen Einfluss auf die Ladegeschwindigkeit im Browser. Zu diesem Thema gibt es eine ganze Industrie, welche sich der Optimierung von Webseiten widmet und es gibt auch einige Bücher dazu, auch von Steve Souders (siehe Kapitel 9.6.3). Inwiefern die verschiedenen Browser über eine grosse Anzahl an Webseiten reagieren, konnte im Umfang dieser Arbeit nicht getestet werden. Dazu wäre eine komplette Automatisierung der Tests nötig aufgrund der hohen Anzahl benötigter Messungen.

Andere wichtige Kriterien wie RAM Verbrauch, Startgeschwindigkeit der Applikation oder Bedienungskomfort wurden in dieser Arbeit nicht betrachtet.

Die Abbildung 62 zeigt eine Übersicht aller Messungen mit 1 Mbit/s der Webseite 20min.ch Jede einzelne „Kurve“ stellt eine Messreihe eines Browser unter den untersuchten Betriebssystemen dar. Für jede RTT wurde eine andere Farbe verwendet. Bei drei untersuchten Browsern auf zwei Betriebssystemen ergibt das pro Farbe resp. RTT sechs Kurven. In dieser Grafik sollen nicht einzelne Messreihen identifiziert werden können, sondern viel mehr, ein Gefühl für die Spannweite der Resultate gegeben werden. Die meisten Messungen liegen im selben Bereich und grobe Ausreisser konnten nicht erkannt werden.

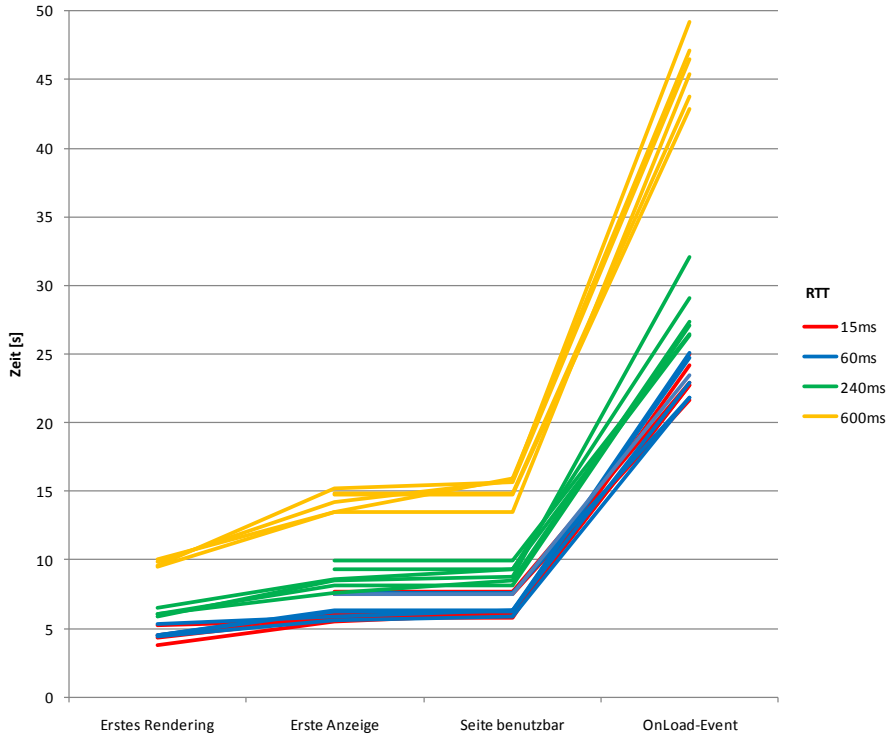


Abbildung 62: Übersicht der Messungen, 1 Mbit/s, 20min.ch

Diese Grafik steht stellvertretend für alle anderen Messungen mit anderen Webseiten und Geschwindigkeiten. Die Aussage, dass keine weite Streuung gemessen wurde, trifft auf alle durchgeführten Messungen zu.

7.3.1. Schneller Internet Explorer

Der Internet Explorer war bei den Messungen bei einigen Kombinationen unerwarteterweise der schnellste der untersuchten Browser. Dies ist doch erstaunlich, da der Internet Explorer betreffend seiner Geschwindigkeit in den IT-Magazinen oft angeschwärzt wird.

20min.ch

Bei den Messungen der Webseite 20min.ch war der Internet Explorer bei „Erste Anzeige“ konstant schneller als die anderen Browser. Der Firefox war im Durchschnitt über alle Messungen bei 20min.ch – sprich bei veränderter RTT und Bandbreite – 4% langsamer als der Internet Explorer. Bei Chrome verzögerte sich die „Erste Anzeige“ deutlich mehr, sie betrug im Durchschnitt 15%. Diese beiden Werte sind jedoch mit Vorsicht zu geniessen, da die Messgenauigkeit in diesem Fall ungefähr bei 5% liegt. Im Minimum kann aber von der Tendenz gesprochen werden, dass der Internet Explorer bei „Erste Anzeige“ für diese untersuchte Webseite besser abschneidet als die anderen beiden Browser, da er meistens schneller war.

Bei „Seite benutzbar“ liegt der Internet Explorer nur wenig hinter dem Firefox zurück und ein wenig vor dem Chrome. Da die Werte hier aber genauso eng zusammen liegen, kann aufgrund der Messungenauigkeit keine konkrete Aussage gemacht werden.

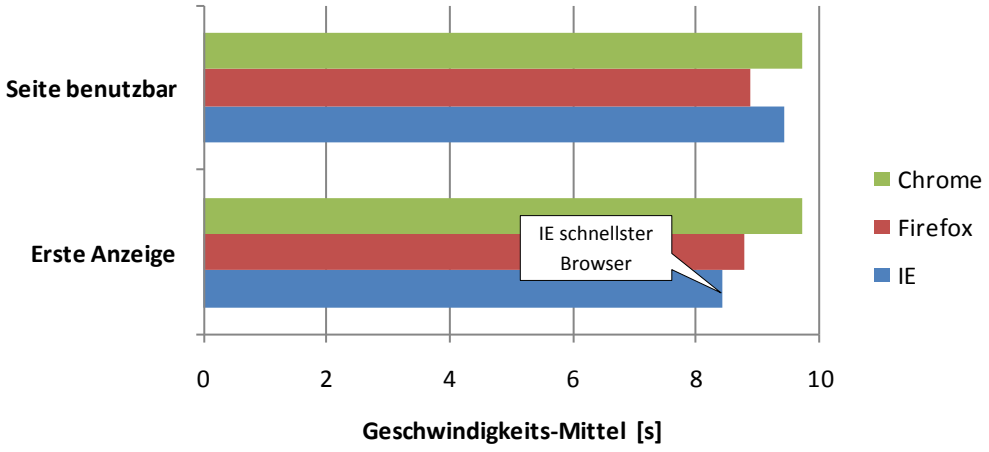


Abbildung 63: Browser-Vergleich 20min.ch, kleiner ist besser

sbb.ch

Interessanter ist das Ergebnis der Messungen bei sbb.ch. Hier schneidet der Internet Explorer bei „Seite benutzbar“ klar besser ab als die beiden anderen Browser. Chrome benötigt durchschnittlich 18% mehr Zeit als der Internet Explorer. Zum Firefox ist der Unterschied noch grösser: Im Durchschnitt ist die Webseite 26% später benutzbar als im Internet Explorer. Die Messgenauigkeit liegt ungefähr um 5%. Somit kann bei der Webseite sbb.ch von einem konkreten Geschwindigkeitsgewinn mit dem Internet Explorer gesprochen werden!

Bei „Erste Anzeige“ überzeugt der Internet Explorer hingegen nicht. Jedoch ist die schnelle Benutzbarkeit bei der Webseite sbb.ch höher zu werten als deren schnelle erste Anzeige. Beim Internet Explorer liegen die beiden Werte „Seite benutzbar“ und „Erste Anzeige“ ohnehin nur minim auseinander.

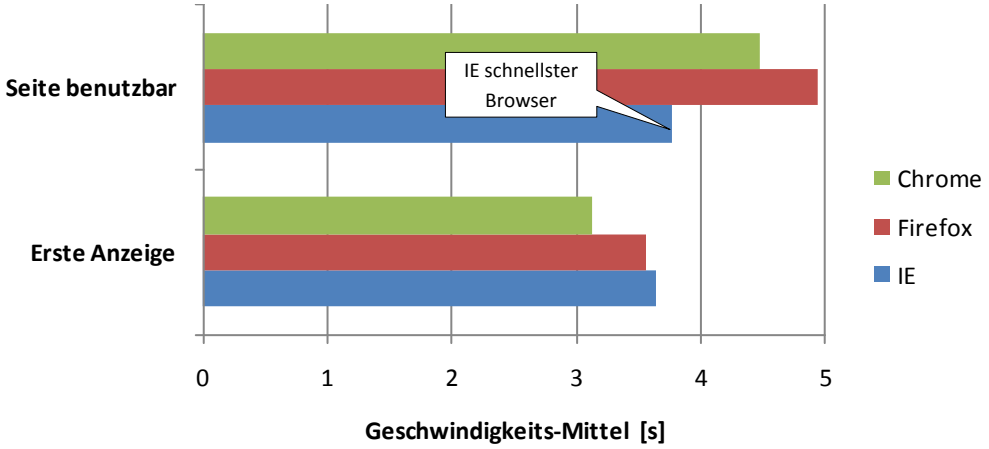


Abbildung 64: Browser-Vergleich sbb.ch, kleiner ist besser

Allgemein

Eine vollständige Antwort auf die Frage, wo die Gründe für eine schnelle erste Anzeige oder eine schnelle Benutzbarkeit liegen, kann im Rahmen dieser Arbeit nicht gegeben werden. Hierfür wären einige browserinterne Details notwendig, welche z.B. für den Internet Explorer gar nicht offengelegt sind. Es konnte jedoch festgestellt werden, dass die unterschiedliche Rendering-Reihenfolge der verschiedenen Browser (siehe Kapitel 7.3.4) einen Einfluss auf die in diesem Kapitel beleuchteten Zeitpunkte „Erste Anzeige“ und „Seite benutzbar“ hat.

Dass der Internet Explorer in gewissen Situationen bei „Erste Anzeige“ und „Seite benutzbar“ die Nase vorne hat, überrascht ein wenig. Häufig ist dieser doch eher als langsamer Browser aus diversen Testberichten bekannt [93]. Schaut man sich jedoch die Browsertests genauer an, so wird darin meistens hauptsächlich die JavaScript-Geschwindigkeit betrachtet resp. verglichen und nicht die von uns analysierte EUE.

Aus diesem Grund wurden die Messungen für alle Browser auf einer JavaScript-lastigen Webseite erneut durchgeführt. Dabei wurde auf die Variation der RTT verzichtet, da es primär um den Einfluss des JavaScripts geht. Als JavaScript-intensive Webseite wurde vom Betreuer – aufgrund seiner Kenntnisse über deren Aufbau – www.cnlab.ch/fussball vorgeschlagen. Nach dem Laden der ersten Elemente werden mittels JavaScript weitere Daten geladen und die Webseitendarstellung mittels JavaScript angepasst.

Abbildung 65: Lade-Ansichten cnlab.ch/fussball, Firefox

Da das Endkundenerlebnis bei dieser Seite stark von der Menge der geladenen und darzustellenden Informationen abhängt, wurde der während der Tests standardmässig geladene Datensatz notiert. Es handelt sich hierbei um Daten des folgenden Spiels: FC Oberglatt – FC Republika Srpska, 1. Halbzeit vom 06.09.2010, 20:00 (18 Tracks).

Wie in Abbildung 66 sichtbar ist, schneidet der Internet Explorer bei einer solch JavaScript-lastigen Webseite massiv schlechter ab als die andere Browser. Bis die Webseite benutzbar ist, benötigt der Firefox im Durchschnitt 26% weniger Zeit als der Internet Explorer. Chrome ist sogar um 44% schneller. Die Messgenauigkeit liegt in diesem Fall bei 6%. Damit wurde bewiesen, dass die Testberichte, welche über einen langsamen Internet Explorer klagen, dies nicht zu Unrecht tun. In diesen Tests wird jedoch die EUE nur bedingt betrachtet.

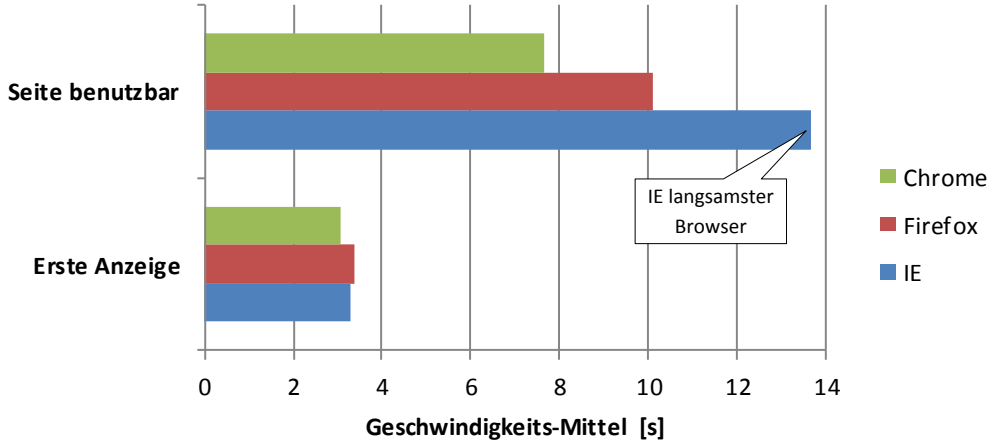


Abbildung 66: Browser-Vergleich cnlab.ch/fussball, kleiner ist besser

7.3.2. Windows 7 nicht schneller als XP

Bei der Zeit, bis die Webseite das erste Mal benutzbar ist („Seite benutzbar“), zeigen sich zwischen Windows 7 und Windows XP keine gravierenden Unterschiede. Dies war auch nicht zu erwarten, da das TCP Auto-Tuning des „Next Generation TCP/IP Stack“ von Windows 7 (siehe Kapitel 4.2.1) am Anfang von Datenübertragungen noch nicht richtig wirken kann.

In der Abbildung 67 sind Messungen dargestellt, welche auf den beiden Betriebssystemen Windows 7 sowie Windows XP gleichermassen durchgeführt wurden. Dies bedeutet für die zwei vertikal zusammengehörenden Messungen, dass diese mit identischen Browsern unter denselben Netzwerkbedingungen durchgeführt wurden und jeweils nur das Betriebssystem variiert wurde. Als Webseiten wurde 20min.ch und sbb.ch genommen, wobei dies in diesem Zusammenhang nicht relevant ist.

Wie zu sehen ist, liegen fast alle Messungen der beiden Betriebssysteme sehr nahe beieinander, wobei die grösseren Abweichungen Dreck-Effekte sind. Aufgrund der Messgenauigkeit von 5% kann zu der Zeit, bis die Webseite das erste Mal benutzbar ist, keine konkrete Abweichung zwischen Windows 7 und Windows XP angegeben werden. Die beiden Betriebssysteme liegen bei diesem Wert in etwa gleich auf.

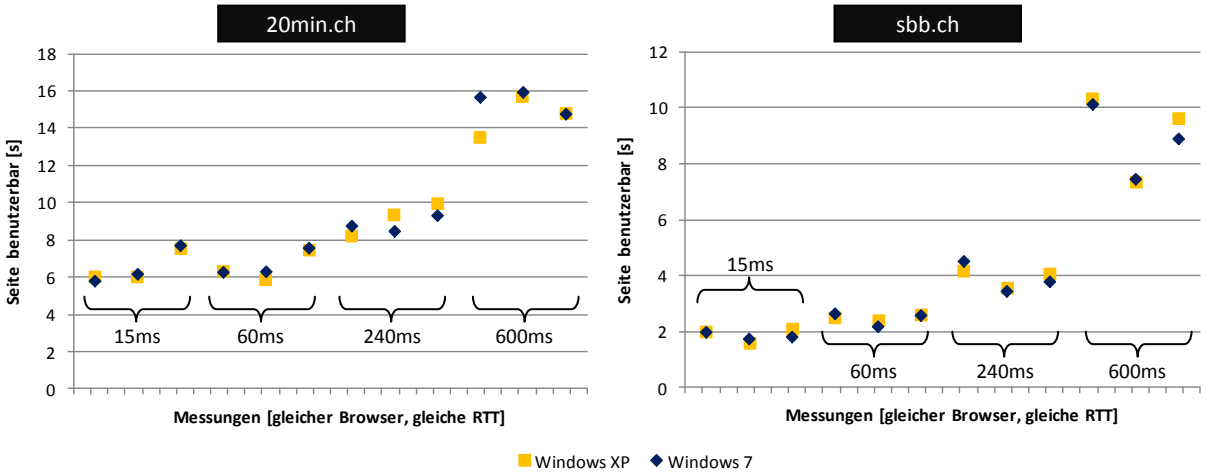


Abbildung 67: Verteilung „Seite benutzbar“, Windows XP / 7, kleiner ist besser

Hingegen könnte aufgrund des „TCP Receive Window Auto-Tuning“ von Windows 7 (siehe Kapitel 4.2.1) erwartet werden, dass eine datenmässig grosse Webseite wie 20min.ch in Windows 7 gegenüber XP schneller vollständig geladen ist („OnLoad-Event“). Denn bei kurzlebigen Verbindungen hat das Auto-Tuning kaum Zeit für eine optimale Bestimmung der Einstellungen.

Damit ein Geschwindigkeitsvorteil durch das Auto-Tunings beobachtet werden kann, muss das Bandbreiten-Delay-Produkt (BDP, siehe Kapitel 0) grösser sein als die maximale RWin-Grösse von Windows XP, welche 65‘535 Bytes beträgt (siehe Kapitel 4.1). Bei einer Bandbreite von 20 Mbit/s und einer Verzögerung von 600ms ist dies gegeben. Dann müsste das RWin bei einer Verbindung 1‘500‘000 Bytes gross sein. Da aber bei unserem Beispiel zu 20min.ch – zu dem in der Abbildung 68 markierten Zeitpunkt – 12 TCP-Verbindungen offen sind, kann die RWin in diesem Fall nochmals durch 12 geteilt werden. Dies ergibt ein RWin von 125‘000 Bytes pro Verbindung. Die 12 TCP-Verbindungen kommen durch zwei Hosts à 6 Verbindungen zustande.

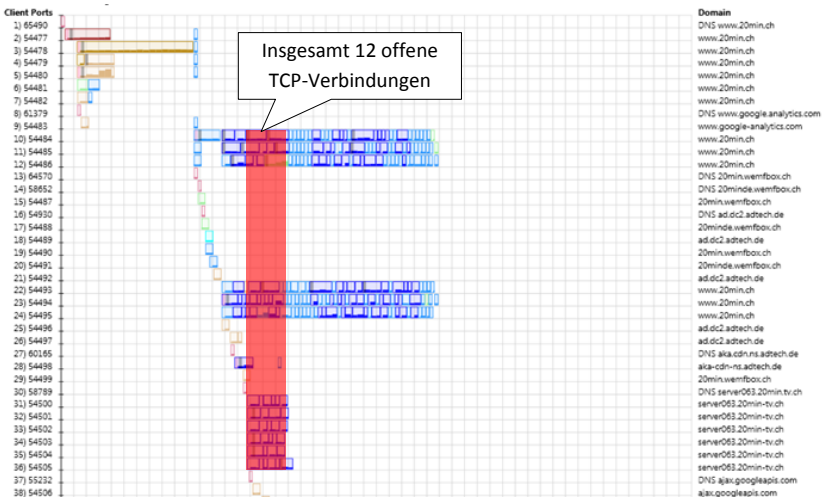


Abbildung 68: Microsoft Visual Round Trip Analyzer, offene TCP-Verbindungen, 20min.ch

Die in der Abbildung 69 dargestellten Messungen der Webseite 20min.ch haben jedoch gezeigt, dass bei einer Bandbreite von 20 Mbit/s und einer Verzögerung von 600ms die Webseite in Windows 7 im Vergleich zu XP nicht schneller vollständig geladen wird. Ein allgemeiner Vergleich zu den Messungen

mit 1 Mbit/s zeigen nur einen minimen effektiven Geschwindigkeitsgewinn bei der 20 Mbit/s-Verbindung. Dies ist sehr erstaunlich, da die Bandbreite doch um den Faktor 20 grösser ist. Hier zeigt sich, dass die RTT einen grösseren Einfluss auf die EUE hat, als die Bandbreite.

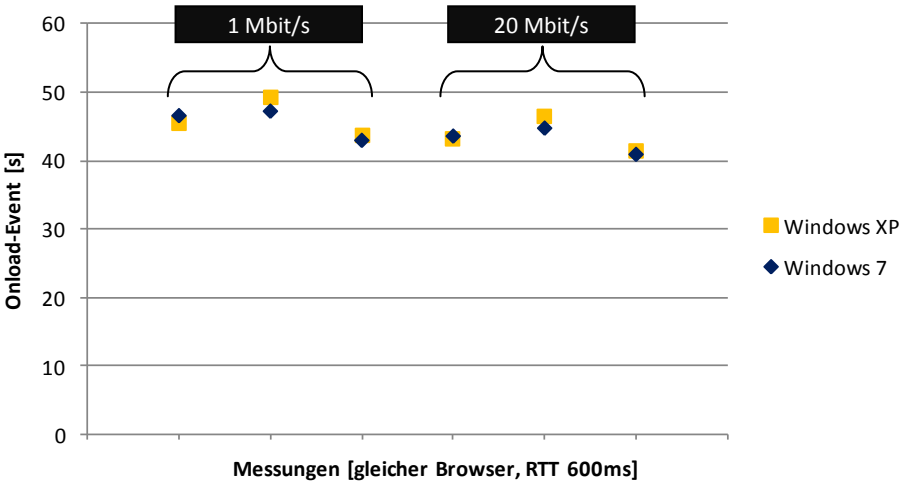


Abbildung 69: Verteilung „OnLoad-Event“, Windows XP / 7, 20min.ch, kleiner ist besser

Wireshark-Auswertungen haben gezeigt, dass die Beschränkung der Geschwindigkeit auf die gewählte RWin-Grösse von Windows 7 zurückzuführen ist. Von den 12 Verbindungen – welche zur Webseite 20min.ch aufgebaut werden – besitzt nur eine einzige ein grösseres RWin als normal und dieses beträgt 186‘300 Bytes. Alle anderen Verbindungen haben die gleiche RWin von 66‘640 Bytes. Die RWins aller Verbindungen addiert ergibt ein Gesamt-RWin von 919‘340 Bytes. Es wäre jedoch ein Wert von 1‘500‘000 Bytes nötig, damit über alle Verbindungen gesehen, die volle Bandbreite genutzt werden könnte.
Diese – nicht optimal gewählten – RWin Grössen sind die einschränkenden Faktoren für die Übertragungsgeschwindigkeit bei einem solch grossen BDP.

Weiter wurden die Auswirkungen einer zu kleinen RWins und die des Auto-Tunings auf einen Download untersucht. Es wurde eine Datei der Grösse 19‘517 KB [94] mit Firefox heruntergeladen unter Windows XP sowie Windows 7 bei einer Bandbreite von 20‘000 kbit/s und einer RTT von 600ms. Dabei erreichte Windows 7 eine durchschnittliche Datenrate von 130.1 KB/s und Windows XP von 103.8 KB/s.
Wie die Auswertungen der Wireshark-Aufzeichnungen gezeigt haben, ist auch hier wieder das zu kleine RWin für die Limitierung verantwortlich. Wie im vorherigen Fall könnte nur mit einer RWin von 1‘500‘000 Bytes die volle Geschwindigkeit erreicht werden. Wie in Abbildung 70 ersichtlich ist, erreicht Windows 7 durch Auto-Tuning eine höhere Download-Geschwindigkeit als Windows XP und benötigt für den Download ca. 20% weniger Zeit als Windows XP. Jedoch besteht auch bei Windows 7 noch Verbesserungspotential: Die RWin-Grösse könnte besser an das BDP angepasst werden. Eine andere Möglichkeit wäre, mehrere Verbindungen für Download einer einzelnen Datei aufzubauen, damit die RWin dieser Verbindungen addiert werden kann. Zum jetzigen Zeitpunkt unterstützt diese Funktion jedoch keiner der untersuchten Browsern (IE, Firefox, Chrome).

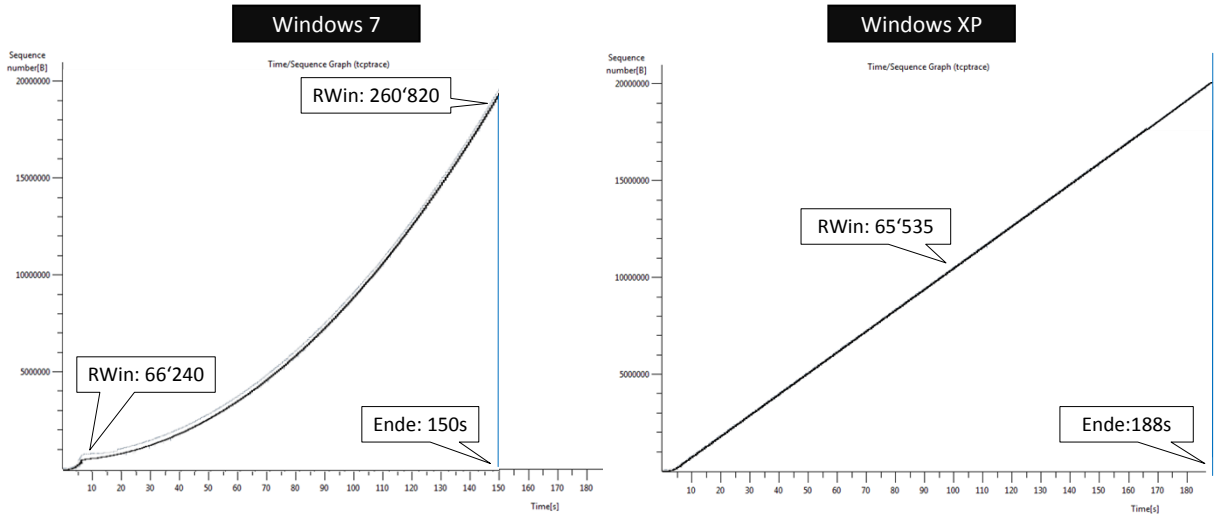


Abbildung 70: tcptrace des Downloads unter Windows XP / 7,

7.3.3. Steigende RTT wirkt nicht linear

Bei zunehmender RTT kann festgestellt werden, dass die Kurven der verschiedenen Messwerte unterschiedlich stark ansteigen. Wie in der Abbildung 71 zu erkennen ist, besitzt der „OnLoad-Event“ immer die steilste Kurve. Dies ist dadurch zu begründen, dass bei diesem Messwert immer am meisten Webseiten-Elemente geladen werden müssen und sich die zunehmende RTT im Verhältnis zu den anderen Messwerten stärker auswirkt. Die Differenz der Messwerte verhält sich jedoch ungefähr linear.

Die einzelnen Kurven der Messwerte zeigen kein lineares Wachstumsverhalten. Es kann eine frühe Phase einer exponentiellen Kurve erkannt werden, wobei diese abgesehen von der „OnLoad-Event“-Kurve fast einem linearen Wachstum entspricht.
Von einem eigentlichen exponentiellen Verhalten kann in der Realität jedoch nicht gesprochen werden. Denn viel höhere RTTs, als die von uns getesteten Werte, sind nicht praxisrelevant und somit kommt eine immer stärkere exponentielle Steigung nie wirklich zum Zug. Der Grund für die nicht lineare Steigung ist nicht vollständig klar.

Bei der Webseite sbb.ch konnte beobachtet werden, dass mit zunehmender RTT der Server nach einer oder mehreren GET-Anfragen die TCP-Verbindung unerwartet schliesst. Dies obwohl ein genügend hohes Keep-Alive von 15s in der Anfrage gesetzt wurde und noch weitere Objekte der Webseite abgerufen werden müssen. Der dadurch bedingte erneute Verbindungsaufbau benötigt zusätzliche Zeit und führt vermutlich zu dem nicht linearen Ergebnis. Es ist nicht vollständig klar, ob dies der einzige Faktor ist. Es steht jedoch fest, dass es nicht mit der RWin zusammenhängt, da diese in allen Fällen jeweils dieselbe war und auch zusätzliche Paket-Verluste ausgeschlossen werden können. Ob die Browser-interne Verarbeitung eine Rolle spielt, kann nicht abgeschätzt werden.

Auch bei 20min.ch konnte ein unnötiger Abbau und Neuaufbau von TCP-Verbindungen beobachtet werden. Hierbei konnte jedoch nicht die Regelmässigkeit festgestellt werden, dass eine zunehmende RTT auch mehr Neuverbindungen zur Folge hätte. Somit lässt sich die Begründung von sbb.ch nicht auf 20min.ch übertragen und der Effekt der Nichtlinearität, vor allem bei der „OnLoad-Event“-Kurve,

ist nicht erklärbar. Ob auch hier die Browser-interne Verarbeitung den entscheidenden Faktor spielt, kann nicht abgeschätzt werden.

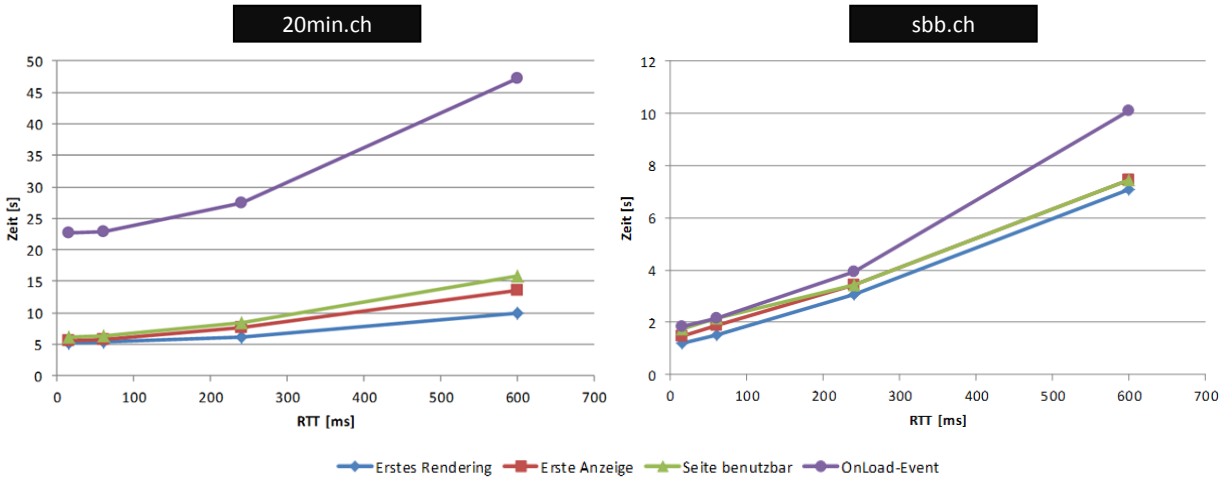


Abbildung 71: Verschiedene RTT, 1Mbit/s, Internet Explorer, Windows 7

Die abgebildeten Graphen beziehen sich auf den Internet Explorer bei einer Geschwindigkeit von 1 Mbit/s. Die beiden anderen Browser Firefox und Chrome verhalten sich analog zum Internet Explorer. Auch bei einer Bandbreite von 20 Mbit/s kommt es zu einem ähnlichen Verhalten, nur dass dort aufgrund des limitierenden RWin die „OnLoad-Event"-Kurve noch steiler steigt.

7.3.4. Einfluss der Rendering-Reihenfolge

Bei den Messungen ist aufgefallen, dass die verschiedenen Browser die in Webseiten enthaltenen Elemente häufig in einer unterschiedlichen Reihenfolge rendern. Dies wirkt sich entscheidend auf die EUE aus, da die für die Webseite am wichtigsten Elemente zuerst dargestellt werden sollten. So ist es z.B. bei einer News-Seite für den Endbenutzer am wichtigsten, dass zuerst die neusten Schlagzeilen angezeigt werden anstatt irgendwelche Umfragen oder Werbung.

Die Rendering-Reihenfolge fällt bei einer tiefen RTT (15ms und 60ms) kaum auf und fällt bei den kurzen Ladezeiten auch nicht ins Gewicht. Bei einer hohen RTT (240ms und 600ms) sowie einer tiefen Bandbreite (1000 kbit/s), bei welcher der Browser relativ lange auf die einzelnen Objekte warten muss, sind die Unterschiede bei der Darstellung hingegen deutlicher sichtbar.

sbb.ch

Bei der SBB-Webseite sieht man die Unterschiede bei der Reihenfolge der Darstellung zwischen den verschiedenen Browsern ziemlich gut. Dabei geht es in den folgenden Screenshots, die diesen Effekt illustrieren, nicht um die genauen Zeiten, sondern primär um den Ablauf des Seitenaufbaus. Deshalb wurden dort auch keine Zeitangaben hinterlegt.

Firefox lädt die Seite Stück für Stück. Ein sehr wichtiges Element der Webseite – das Formular für die Fahrplanabfrage – erscheint jedoch erst am Schluss. Das Navigationsmenü wird jedoch bereits direkt zu Beginn angezeigt.

Abbildung 72: Lade-Ansichten sbb.ch, Firefox

Beim Internet Explorer erscheinen hingegen fast alle Textinhalte sowie das Formular für die Fahrplanabfrage direkt zu Beginn. Die Bilder, welche meistens nicht zwingend nötig sind für eine erste Interaktion mit der Webseite, werden erst anschliessend angezeigt. Diese Lade-Reihenfolge lässt im Vergleich zu den anderen Browsern eine Fahrplanabfrage am schnellsten zu.
Gemäss Kapitel 5.2 kann der IE Scripts und Bilder nicht parallel herunterladen (siehe Spalte „|| Script Bild“ in Tabelle 7). Dies ist vermutlich der Grund für die späte Anzeige der Bilder.

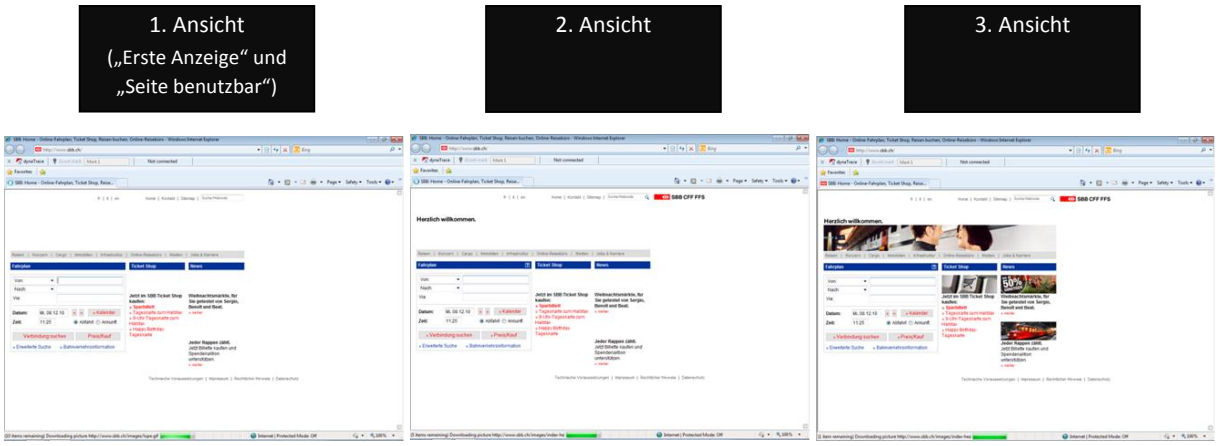


Abbildung 73: Lade-Ansichten sbb.ch, Internet Explorer

Beim Chrome sieht der Ablauf ziemlich ähnlich aus wie beim Firefox. Die Webseite wird Stück für Stück geladen und das Formular für die Fahrplanabfrage erscheint erst am Schluss.

Abbildung 74: Lade-Ansichten sbb.ch, Chrome

20min.ch

Bei der Webseite 20min.ch besteht zwischen der Rendering-Reihenfolge der Browsern nicht so ein grosser Unterschied wie bei sbb.ch. Bei „Erste Anzeige“ wird immer bereits die Navigation und Teile der ersten Schlagzeile dargestellt. Unterschiede bestehen hier hauptsächlich darin, in welcher Reihenfolge die Bilder anschliessend geladen wird.

Im Firefox ist im Durchschnitt die Webseite 20min.ch am schnellsten benutzbar. Wichtig dafür ist der erste Beitrag und das Bild dazu (siehe Kapitel 7.2.3). Interessant ist, dass die Schlagzeile als Bild hinterlegt ist. Im Gegensatz zum Internet Explorer wird im Firefox diese zuerst als Text angezeigt (sprich aus dem HTML, siehe Abbildung 75 links) und erst im Nachhinein wird das entsprechende Bild geladen. Für die Schlagzeile wird wohl ein Bild verwendet, da es sich um eine spezielle Schriftart handelt, welche nicht auf allen Computern der Webseitenbesucher installiert ist. Damit jedoch die gewünschte Darstellung erreicht wird, werden Bilder eingesetzt.

Abbildung 75: Lade-Ansichten 20min.ch, Firefox

Der Internet Explorer hat im Gegensatz zu den anderen Browsern die schnellste erste Anzeige, braucht dafür aber danach umso mehr Zeit bis die Webseite benutzbar ist. Störend ist, dass die Schlagzeile erst spät sichtbar wird. Denn es wird nicht die Textform davon angezeigt wie im Firefox und das entsprechende Bild der Schrift wird zu spät geladen.

Abbildung 76: Lade-Ansichten 20min.ch, Internet Explorer

Beim Chrome dauert es relativ lange, bis der erste Inhalt erscheint. Dafür wird bei der ersten Anzeige mehr Inhalt dargestellt als bei den anderen Browsern. Auffallend ist, dass Chrome den rechten Beitrag von Beginn an anzeigt.

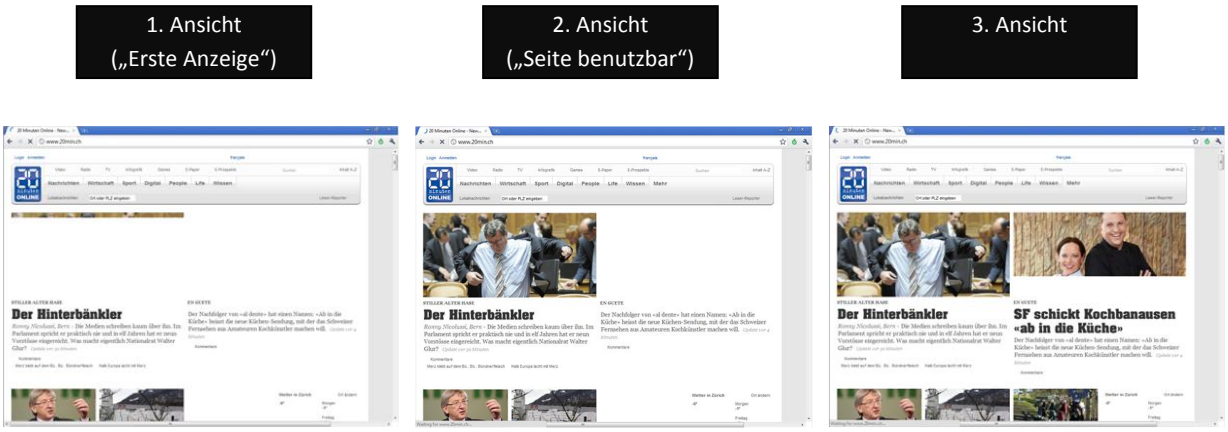


Abbildung 77: Lade-Ansichten 20min.ch, Chrome

Allgemein

Die Ursache der verschiedenen Rendering-Reihenfolge ist die unterschiedliche Browser-interne Verarbeitung der Webseite. Dieser Aspekt konnte im Rahmen dieser Arbeit nicht weiter behandelt werden. In den Messungen wurde jedoch ersichtlich, dass die drei Browser eine leicht unterschiedliche Lade-Reihenfolge der Webseiten-Elemente haben. Dies wirkt sich wiederum auf deren Rendering-Reihenfolge aus.

Durch einen guten Aufbau einer Webseite kann die Rendering-Reihenfolge positiv beeinflusst werden, was in einem besseren Endkundenerlebnis resultiert. Ein wichtiger Ansatz ist hierbei das asynchrone Nachladen von Webseiten-Teilen mittels JavaScript. In diesem Zusammenhang können die Bücher von Steve Souders (siehe Kapitel 9.6.3) empfohlen werden.

7.3.5. DNS Prefetching

Bei den Messungen der 20min.ch-Seite ist aufgefallen, dass der Firefox bei jeder Messung mehr DNS-Anfragen macht als der Internet Explorer. Im Mittelwert macht der Firefox 18 und der IE 12 DNS Anfragen. Bei der Webseite sbb.ch ist das Verhalten bei beiden Browsern gleich, beide machen nur 2 DNS-Anfragen.

Dieser Unterschied kann mit dem DNS Prefetching erklärt werden. Firefox und auch Chrome unterstützen diese Funktion, der Internet Explorer nicht, es gibt jedoch Plugins dafür [95]. Diese Funktion ist bei Firefox ab Version 3.5 standardmässig aktiviert [95] und bei Chrome bereits seit der ersten Version (ab September 2008) verfügbar. Dabei untersucht der Browser im Hintergrund die Links auf der aktuellen Seite und macht DNS-Abfragen für diese Hosts, sofern er die Adressen noch nicht bereits im Cache hat.

Klickt nun der Benutzer später einen dieser Links tatsächlich an, so ist die benötigte DNS-Abfrage bereits getätigt worden und die IP-Adresse bereits im Cache (beim DNS-Server sowie lokaler DNS-Cache). Somit kann etwas Zeit gespart werden beim tatsächlichen Aufruf der Webseite hinter dem Link. Die DNS-Abfrage zu www.sandiego.com dauert z.B. beim ersten Mal 645ms, wenn noch keine Adresse im Cache des lokalen DNS-Servers liegt. Bei der erneuten Anfrage dauert die Auflösung nur noch 3ms. Auch bei DNS-Abfragen zu Schweizer Domains gibt es einen Geschwindigkeitsgewinn, doch fällt er dort geringer aus: Bei www.hungerbuehlers.ch dauert es 3 anstatt 50ms. Diese Zeiten wurden mit dem DNS-Tool dig [96] gemessen, bei Verwendung des DNS-Servers der HSR für die DNS-Auflösung.

Gemäss Google [97] erfolgt der erste Besuch einer Webseite mit Chrome im Durchschnitt 250ms schneller bei aktiviertem DNS Prefetching. Chrome macht bereits beim Browserstart ein DNS Prefetching für die auf der Chrome Startseite dargestellten zuletzt besuchten Webseiten. Weiterhin macht Chrome bereits DNS-Abfragen während der Benutzer die URL ins Adressfeld eintippt, noch bevor die Enter-Taste gedrückt wird. Dabei handelt es sich um Abfragen zu einem Teil des Hostnamens.

Bei Suchmaschinen-Ergebnisseiten verspricht das DNS Prefetching am meisten Zeitgewinn, denn diese Seiten enthalten viele Links, welche der Benutzer noch nicht besucht hat. Bei einem Webportal, das primär Links auf interne Ressourcen mit demselben Hostnamen enthält, kann kaum profitiert werden. Dass jedoch auch dort DNS Prefetching stattfindet, liegt an allfälligen Subdomains bzw. anderen Hostnamen oder der eingebundenen Werbung.

Das DNS Prefetching führte jedoch bei unseren Tests zu keinem Geschwindigkeitsvorteil bei Firefox und Chrome. Dies lag jedoch an der Testweise und nicht an der Funktion. Denn es wurde nur der erste Aufruf der Webseite getestet und nicht noch anschliessende Navigation auf der Seite. Erst dort hätte sich diese Funktion ausgewirkt.

Konkret wurden beim Aufruf der 20min Webseite zusätzlich zu den bereits im Kapitel 7.2.3 gelisteten Hostnamen noch folgende DNS Namen aufgelöst:

- www.20minutenmobile.ch
- ajax.googleapis.com
- www.piazza.ch
- ch.tillate.ch
- www.beilagen.20min.ch
- www.jobwinner.ch
- www.homegate.ch
- leserreporter.20min.ch
- www.adventskalender.20min.ch
- www.bodycoach.20min.ch

Je nach angezeigtem Inhalt auf der Webseite variiert, die Anzahl der obigen angefragten Hostnamen.

Gemäss [95] ist das Prefetching heikel bzgl. der Privatsphäre, da es zu einem falschen Verdacht führen kann. Ruft der Benutzer eine Seite auf, welche z.B. eine Liste von gesperrten Domains enthält, so macht der Browser für all diese Domains eine DNS-Abfrage, obwohl der Benutzer keine dieser Seiten besucht hat. Dies wird dann in der Logdatei des DNS-Servers abgespeichert zusammen mit der eigenen IP. Aus diesem Grund haben einige Webseiten wie Wikileaks eine Zwischenseite eingebaut bei Links zu externem Inhalt [95].

Weitergehend ist anzumerken, dass Firefox sogar noch weiter geht, indem er Link Prefetching [98] macht (wird von Chrome und IE (noch) nicht unterstützt, ist aber Bestandteil von HTML5 [99]). Dabei lädt er den Inhalt von speziell markierten Links herunter, bevor der Benutzer diesen angeklickt hat. Dies jedoch erst, nachdem die aktuelle Seite komplett geladen wurde, um diesen Vorgang nicht zu stören. Link Prefetching verursacht jedoch mehr Datenverkehr als DNS Prefetching und ist bei sich dauernd ändernden dynamischen Seiten sinnlos, da diese beim echten Zugriff erneut geladen werden müssen. Die Wahrscheinlichkeit eine solche Seite anzutreffen, ist grösser, als dass sich der DNS Eintrag zwischen Prefetching und tatsächlichem Aufruf ändert.

8. Schlussfolgerungen

Resultate

Im Rahmen dieser Arbeit wurden die Top 3-Browser Internet Explorer, Firefox und Chrome bezüglich des Endkundenerlebnisses beim Surfen unter Windows XP sowie Windows 7 untersucht. Es zeigte sich, dass das Endkundenerlebnis von vielen Faktoren abhängig und nur schwer objektiv messbar ist. Nachfolgend die wichtigsten Ergebnisse der Untersuchungen.

Bei einer Internet-Verbindung mit hohem Bandbreite-Delay-Produkt brachte das TCP Auto-Tuning von Windows 7 im Vergleich zu Windows XP nur einen geringen Geschwindigkeitsvorteil. Die automatische Vergrösserung des Receive Window hielt sich in engen Grenzen. Allgemein konnte festgestellt werden, dass die Antwortzeit den grösseren Einfluss auf das Endkundenerlebnis hat als die verfügbare Bandbreite. Dies kommt daher, dass die meisten TCP-Verbindungen beim Aufruf einer Webseite einen grossen Teil ihrer kurzen Lebenszeit in der Slow Start-Phase bei geringer Datenrate verbringen. Wie lange diese Phase dauert wird massgeblich durch die Antwortzeit bestimmt.

Bei zunehmender Antwortzeit konnte zudem festgestellt werden, dass die Zeit bis die Webseite vollständig geladen ist, überproportional zunimmt. Wie Untersuchungen gezeigt haben, sind vermutlich TCP-Verbindungsabbrüche auf der Serverseite für dieses Phänomen verantwortlich.

Beim Vergleich der verschiedenen Browsern war der als langsam eingestufte Internet Explorer erstaunlicherweise bei der Darstellung des ersten Webseiten-Inhalts eher schneller als seine Kontrahenten.

Für die untersuchten Webseiten und Leistungsparameter sind keine gravierenden Unterschiede zwischen den Browsern und Betriebssystemen zu beobachten. Erst bei tiefer Bandbreite und hoher Antwortzeit sind Unterschiede in der Rendering-Reihenfolge sichtbar, welche das Endkundenerlebnis stärker beeinflussten als erwartet.

Verbesserungen

Leider konnten keine statistisch relevanten, quantitativen Resultate gewonnen werden, da aufgrund der fehlenden Test-Automation nur wenige Messungen durchgeführt werden konnten. Weil der Aufwand der Tests exponentiell mit der Anzahl Leistungsparameter wächst, ist eine Automatisierung sehr zu empfehlen, auch wenn der Aufwand dafür gross ist. Ebenso könnten dann die Tests für eine weitaus grössere Anzahl an Webseiten durchgeführt werden, als dies in der vorliegenden Arbeit gemacht wurde.

Ausblick

Für weitere Tests wäre ein umfangreicherer Vergleich der Betriebssysteme wie Windows mit Linux oder OS X interessant. Zusätzlich wäre der Einbezug der Browser Safari und Opera denkbar. Um die Browser besser vergleichen zu können, sind Messungen über mehrere Webseiten nötig, da das Endkundenerlebnis stark von deren Aufbau abhängt. Interessant wäre sicher auch die vertiefte Analyse einzelner Aspekte wie Optimierungen im Webseiten-Aufbau. Da sehr viele Faktoren die EUE beeinflussen, konnten viele interessante Punkte leider nur oberflächlich betrachtet werden.

9. Anhang

9.1. Aufgabenstellung

Herbstsemester 2010 (13.09.2010 - 13.02.2011)

Betreuer: Prof. Dr. P. Heinzmann
Industriepartner: cnlab AG

Mit den cnlab Performance Tests (Performance Applet für Browser) [a] und der cnlab iPhone App [b] können Endkunden und Internet Service Provider die Leistungswerte ihrer Internet-Verbindung überprüfen. Täglich führen mehrere Tausend Endkunden Tests durch und starten dabei Up- und Downloads zu cnlab Referenzsystemen. Während den Tests werden auf der Client- und auf der Serverseite [c] verschiedenste Leistungsparameter erfasst. Die Tests liefern detaillierte Auswertungen zu Up-/Download-Datenraten und zu Antwortzeiten (Round-Trip-Time, RTT). Auch erfasst werden Einstellungen und Lokationen der Systeme, auf denen Testmessungen ausgeführt wurden.

Im Rahmen dieser Arbeit sind die Einflüsse der verschiedenen Leistungsparameter auf das „Endkundenerlebnis“ zu untersuchen. Besondere Beachtung ist dabei dem Einfluss der verschiedenen Browserprodukte und Browserversionen zu schenken. Es ist aufzuzeigen, unter welchen Bedingungen welche Parameter den bestimmenden Faktor für das „Endkundenerlebnis“ bilden. Vor allem die Einflüsse der RTT sollen beim Zugang zu verschiedenen Web-Portalen untersucht werden.

Zuerst sind mit Hilfe eines Netzemulators [d] und mit verschiedenen Netzwerk-Testwerkzeugen (z.B. cnlab Performance Test Advanced Mode [e] und iPerf [f]) fundierte Kenntnisse zu den Einflüssen der Parameter zu erarbeiten. Danach ist anhand von Tests bei verschiedenen Web-Portalen der Einfluss der Leistungsparameter auf die Darstellung der Webseiten mit Internet Explorer, Firefox und Chrome aufzuzeigen. In diesem Zusammenhang sind die Hinweise und Informationen von Google Page Speed [g] und Untersuchungen zur Leistungsfähigkeit verschiedener Browser [h] zu studieren.

Im Rahmen der Untersuchungen sollen auch „alternative Möglichkeiten der Leistungserfassung“ aufgezeigt werden. Es hat sich gezeigt, dass beispielsweise nur knapp hundert Tester pro Tag Messungen mit der Mobile App durchführen (mit dem PC-Browser sind es einige tausend pro Tag). Mittelfristig soll daher mit Hilfe von Messungen des (normalen) Datenverkehrs auf der Serverseite eine Aussage über die Leistungsfähigkeit des Access-Netzes ermöglicht werden. Im zweiten Teil der Arbeit – oder allenfalls in einer Folgearbeit – sind die Möglichkeiten solcher (passiver) Messungen aufzuzeigen.

In einer letzten Phase der Arbeit oder in einer Fortsetzungsarbeit soll das Konzept mit passiven Performance-Messungen in Mobilnetzen anhand einer speziellen Android-EcoHelper-App aufgezeigt werden.

Quellen

[a] cnlab Performance Test Browser
http://www.cnlab.ch/speedtest/

[b] cnlab Performance Test iPhone
http://itunes.apple.com/de/app/cnlab-speedtest/id326450707?mt=8

[c] M. Mathis, J Heffner and R Reddy, "Web100: Extended TCP Instrumentation for Research, Education and Diagnosis", ACM Computer Communications Review, Vol 33, Num 3, July 2003
http://www.web100.org/

[d] Bachelorarbeit V. Condoleo, C. Dirac, R. Ruoss, Netzwerk-Emulatoren auf dem Prüfstand, HSR, FS2008

[e] cnlab Speedtest Advanced Mode
http://hsi.bluewin.ch/speedtest/?adv=1

[f] iPerf Netzwerk Test Programm
http://en.wikipedia.org/wiki/Iperf

[g] Google Page Speed
http://code.google.com/intl/de-DE/speed/page-speed/

[h] Browserscope, Browser Performance Vergleich
http://www.browserscope.org/

9.2. Erklärung über die eigenständige Arbeit

Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellungen und Beiträgen gemäss folgender Aufstellung. Die nicht angegebenen Kapitel wurden gemeinsam erarbeitet.

Gegenstand, Leistung	**Person**	**Funktion**
Kapitel 1, 7.3, 9.6, 9.7; Durchführung der Messungen	Florian Hungerbühler	Autor der Arbeit
Kapitel 1, 0, 0, 5, 7.1, 7.2, 9.8, 0; Durchführung der Messungen	Raphael Neumann	Autor der Arbeit
Idee, Aufgabenstellung, allgemeines Pflichtenheft, Betreuung während der Arbeit	Prof. Dr. P. Heinzmann	Betreuer

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt habe,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Rapperswil, den 23.12.2010 .....

Florian Hungerbühler, Student

Rapperswil, den 23.12.2010 .....

Raphael Neumann, Student

9.3. Rechtevereinbarung

9.3.1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der *Studienarbeit „Internet Performance und End User Experience"* von Florian Hungerbühler und Raphael Neumann unter der Betreuung von Prof. Dr. P. Heinzmann geregelt.

9.3.2. Urheberrecht

Die Urheberrechte stehen den Studenten zu.

9.3.3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, d.h. von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Professor sowie vom Industriepartner verwendet und weiter entwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht, aber nicht Pflicht.

Rapperswil, den 23.12.2010 .....

Florian Hungerbühler, Student

Rapperswil, den 23.12.2010 .....

Raphael Neumann, Student

Rapperswil, den 23.12.2010 .....

Dr. Prof. P. Heinzmann, Betreuer

Rapperswil, den 23.12.2010 .....

Dr. Prof. P. Heinzmann, cnlab AG, Industriepartner

9.4. Erfahrungsberichte

9.4.1. Florian Hungerbühler

Für mich stellte diese Studienarbeit „Internet Performance und End User Experience“ die erste eigene Forschungsarbeit dar, welche einen solchen grossen Umfang besass. Der grösste Reiz dabei bestand darin, unter dem Kontext von verschiedenen Internet-Leistungswerten neue Erkenntnisse betreffend des Endkundenerlebnisses herausfinden zu können.

Eindrücklich und zugleich auch sehr aufwändig fand ich die Analyse der Messungen. Ich hatte nicht erwartet, dass bereits bei diesen „nicht sehr breit angelegten“ Messungen eine solche Fülle an Analysen, Diagrammen und Schlüssen möglich sind. Aus diesem Grund ist es auch nicht ganz trivial, die richtigen Dinge zu erkennen und herauszupicken. Was ungemein hilft, ist, wenn man eine konkrete Vorstellung hat, wie das Ergebnis aussehen könnte.

Die eigentlichen Messungen wurden von mir unterschätzt. Es stellte sich als sehr aufwändig heraus, für die verschiedenen Leistungswerte alle Messungen immer wieder zu wiederholen. Dass – um aussagekräftige Werte zu erhalten – für jede Kombination von Betriebssystem, Browser und Leistungswerte mehre Messungen angefertigt werden mussten, erhöhte den Messaufwand nochmals immens. Für ein anderes Mal müsste zuerst eingehender abgeklärt werden, ob nicht doch auf irgendeine Weise eine Automation der Messungen möglich wäre. Dies hätte auch eine breiter angelegte Messung ermöglicht und zugleich auch die Qualität der Messwerte durch eine nochmals erhöhte Messhäufigkeit erhöht. Jedoch schätze ich eine Automation mit den gewählten Parametern als extrem schwierig ein.

Im Rahmen der Arbeit wurde das Verständnis des Protokolls TCP doch sehr verbessert. Es ist erstaunlich, wie komplex TCP doch immer noch ist, obwohl man meint, man habe alle grundlegenden Prinzipien verstanden.

Ein wenig Schade ist die Tatsache, dass oft nicht genügend Zeit vorhanden war, um bei gewissen Punkten tiefer zu graben oder sogar überhaupt zu behandeln. Der Webseiten-Aufbau konnte aus Zeitgründen überhaupt nicht analysiert werden, obwohl dieser eine sehr entscheidende Rolle für das Endbenutzererlebnis hat. Allgemein kann aber gesagt werden, dass so viele Abhängigkeiten und Einflüsse existieren, dass eine generelle Analyse praktisch unmöglich scheint.

Als einen Nachteil von einer Forschungsarbeit empfinde ich, dass das Ergebnis von vornherein nicht klar ist und darum auch immer ein wenig darum gebangt werden muss. Das Resultat ist sehr schwer abschätzbar, nicht wie bei einer Programmierarbeit, wo zum Schluss sicher ein Produkt in den „Händen“ gehalten werden kann. Aus diesem Grund war die Anpassung der NetEm-Box eine dankbare Arbeit.

Die Zusammenarbeit mit Raphael Neumann verlief bestens und ist somit eine gute Voraussetzung für die gemeinsame Bachelorarbeit. Die Gruppengrösse von zwei scheint mir ideal zu sein. Die wöchentlichen Sitzungen fand ich sehr hilfreich und man bleibt auf das Ziel fokussiert.

Allgemein kann gesagt werden, die Studienarbeit liefert eine gute Grundlage für die bevorstehende Bachelorarbeit.

9.4.2. Raphael Neumann

Am Ende der 14-wöchigen Semesterarbeit kann ich auf eine gute und angenehme Zusammenarbeit mit meinem Mitstudenten Florian Hungerbühler zurückblicken. Für mich war es die erste grosse Arbeit, welche im Team erarbeitet wurde und verlief für diesen Umstand überraschend gut. Dazu trugen vermutlich auch die nützlichen, wöchentlichen Sitzungen mit unserem Betreuer bei.

Weniger schätzte ich jedoch die Ungewissheit bzgl. der Resultate der Arbeit, da es sich um eine Art Forschungsarbeit handelte. Dass die Resultate aufgrund der Ungenauigkeit keine statistisch relevanten, quantitativen Aussagen erlauben, ist schade. Denn der Aufwand für die Messungen war um einiges grösser als erwartet. Dies primär, da damit sehr viel Handarbeit verbunden war. Eine vollständige Automatisierung der Tests ist jedoch kaum möglich, da es beim Endkundenerlebnis auch um die menschliche Wahrnehmung geht.

Es konnten auch nicht für alle zu Beginn vereinbarten Netzwerkparameter wie z.B. Paketverlust oder Jitter die Auswirkungen auf die EUE getestet werden, da der Messaufwand exponentiell mit der Anzahl Parameter steigt. Vermutlich wäre ein Test mit weniger Browsern bzw. Betriebssystemen, dafür mehr Messungen und genaueren Auswertungen aussagekräftiger und zufriedenstellender gewesen.

Bei vielen Punkten fehlte die Zeit für eine genaue Analyse und ein detailliertes Verständnis der Effekte, da das Thema sehr weitläufig ist und sehr viele Aspekte mit einfliessen. Die Auswertung der Messwerte brauchte mehr Überlegungen und Aufwand als erwartet. Die nachfolgende Erklärung der gefundenen Effekte gestaltete sich kompliziert, da uns die Gründe z.T. auch unklar waren und die Zeit knapp war. Die Formulierung der Ursachen und der Abhängigkeiten der verschiedenen Aspekte, so dass es auch Aussenstehende verstehen, war eine Herausforderung.

Es konnten jedoch auch viele sehr interessante Erkenntnisse gewonnen werden. Das zuvor mehr oder weniger verstanden geglaubte TCP Protokoll stellte sich im Details als viel komplexer heraus. Die Details sind nun viel klarer, dies auch aufgrund des Studiums zahlreicher RFCs. Diese waren angenehmer zu lesen als erwartet.

Sehr aufschlussreich fand ich die Recherche-Ergebnisse bzgl. Web Performance und dass in den letzten Jahren sehr viel Arbeit in diesem Bereich geleistet wurde und auch aktuell intensiv daran gearbeitet wird. Diesbezüglich war die Webseite von Steve Souders sehr informativ. Die Optimierungsmöglichkeiten bei der Webseitengestaltung hätte ich gerne genauer betrachtet, gibt es dort doch viele, teils unerwartete, Faktoren.

Wir hatten zwar bereits ziemlich früh mit der Dokumentation der Arbeit begonnen, doch musste einiges an Text umgeschrieben werden, da sich die Richtung des Berichts mehrmals leicht änderte. Dies war etwas hinderlich. Trotz des frühen Dokumentationsstarts gab die Überarbeitung, die Korrektur sowie das Zusammenfügen der einzelnen Dokumente sehr viel zu tun. Für die Bachelorarbeit muss für den Schluss auf jeden Fall mehr Zeit dafür eingeplant werden, da auch noch zusätzliche Punkte wie Abstract, Poster, Schlussfolgerung und Erlebnisbericht geschrieben werden müssen. Die Verwaltung der grossen Anzahl an Verweisen im Word Dokument bereitete einiges Kopfzerbrechen, als mehrere Dokumente zusammengefügt wurden.

Ich habe einiges gelernt durch diese Arbeit und hoffe auf eine interessante Bachelorarbeit.

9.5. Risikoanalyse

In diesem Kapitel werden die wichtigsten Risiken analysiert und geeignete Massnahmen gesucht, um den Schaden möglichst klein zu halten.
Diese Analyse beschränkt sich gemäss Diskussion mit unserem Betreuer auf die realistischen Risiken

9.5.1. Risikoliste

Folgende relevante Risiken wurden evaluiert und jeweils bewertet mit einer Eintrittswahrscheinlichkeit (EW) der Ereignisses und des allfälligen Schadens in Stunden (S[h]). Wobei der Schaden Mehrarbeit oder verlorene Zeit bedeutet. Durch die Multiplikation der Wahrscheinlichkeit mit der Stundenanzahl erhält man die einzuplanende Reservezeit (R[h]).

ID	Bezeichnung	Beschreibung	Auswirkungen	EW	S [h]	R [h]
R01	Arbeit wird nicht bestanden.	Die Arbeit wird vom Experten als ungenügend bewertet. - Qualität der Arbeit ist ungenügend. - Einsatz nicht ausreichend. - unterschiedliche Vorstellung der Resultate.	Die Arbeit muss wiederholt werden.	0.1	240	24
R02	Verzettelung	Es wird zu viel Arbeit in einen unwichtigen Teil investiert oder am Ziel vorbei gearbeitet.	Nur teilweise Erfüllung des Ziels oder zu viel Arbeitsaufwand nötig.	0.2	32	6.4
R03	Fehlendes Know How	Es ist nicht genügend Zeit vorhanden für die Erarbeitung des nötigen Know Hows.	Qualität der Forschungsabgabe nimmt ab.	0.2	50	10
R04	Krankheit	Ein Student wird unerwartet krank und fällt für einige Tage aus.	Die Fertigstellung einzelner Arbeitspakete verzögert sich oder die andere Person muss ev. diese Arbeit mit übernehmen.	0.1	20	2

ID	Bezeichnung	Beschreibung	Auswirkungen	EW	S [h]	R [h]
R05	Datenverlust	Verlust eines Teils der Daten (Doku oder Mess-resultate/-auswertun-gen)	Diese Arbeit/ Messun-gen müssen erneut ausgeführt werden, es geht wertvolle Zeit verloren.	0.05	16	0.8
R06	NetEm-Box Aus-fall	Die NetEm-Box fällt aus	Ein neuer Netzwer-kemulator muss auf-gesetzt werden, was Zeit kostet.	0.05	6	0.3
					Total nötige Reserve	1.1

Tabelle 14: Auflistung der Risiken

EW: Eintrittswahrscheinlichkeit
S: Schaden in Anzahl Stunden
R: nötige Reserve in Anzahl Stunden (EW * S)

Zu Beginn waren noch folgende Risiken in der Auflistung:

- Die Definition der EUE bereitet Probleme und kann nicht automatisiert gemessen werden.
- Der Einfluss der Netzqualität auf die Performance der Webseitenanzeige ist zu gross.

Bei der Besprechung der Risikoanalyse mit dem Betreuer hat sich herausgestellt, dass dies eigentlich Bestandteil der Aufgabenstellung ist und nicht zwingend erfüllt werden muss für die Zielerreichung. Hauptkriterium ist dabei die genaue Analyse und wenn sich dabei herausstellt, dass es nicht umsetz-bar ist oder nur in eingeschränkter Weise, so ist das auch ok.

9.5.2. Risikomassnahmen

Um den Schaden zu minimieren, werden folgende Massnahmen ergriffen.

ID	Bezeichnung	Massnahme
R01	Die Arbeit wird nicht bestanden.	Wöchentliche Sitzungen mit dem Experten und regelmässiges Feedback einholen, auch bei einem Teil der Arbeit. Abgabe von Teilen der Arbeit bei Milestones.
R02	Verzettelung	Früh einen guten Projektplan erstellen und mit dem Betreuer besprechen.
R03	Fehlendes Know How	Frühe Einarbeitung und Absprache mit Betreuer um ev. Aufgabenstellung anzupassen.
R04	Krankheit	Alle Daten werden in der Wuala Cloud gespeichert und sind damit allen überall zugänglich. Weiterhin gibt es einen täglichen Wissensaustausch, damit jeder über den Stand der Arbeiten des anderen informiert ist.
R05	Datenverlust	Die Daten werden in der Wuala Cloud gespeichert und sind so redundant verfügbar. Zusätzlich wird noch wöchentlich ein lokales Backup erstellt. Wiederherstellungstest des Backups machen.
R06	NetEm-Box Ausfall	Image der NetEm-Box erstellen.

Tabelle 15: Risikomassnahmen

9.5.3. Risiken nach Massnahmenergreifung

Nach Evaluation der Massnahmen werden die Risiken nochmals neu beurteilt.

ID	Bezeichnung	EW	S [h]	R [h]
R01	Arbeit wird nicht bestanden.	0.05	240	12
R02	Verzettelung	0.1	32	3.2
R03	Fehlendes Know How	0.15	50	7.5
R04	Krankheit	0.1	15	1.5
R05	Datenverlust	0.01	10	0.1
R06	NetEm-Box Ausfall	0.05	2	0.1
Total nötige Reserve				24.4

Tabelle 16: Reevaluation der Risiken

Mit Hilfe der Massnahmen wurde also ungefähr eine Halbierung der Risikoauswirkungen erreicht.

9.5.4. Risikomatrix

Auf eine grafische Darstellung und Einteilung der Risiken mittels einer Risikomatrix wird gemäss mündlicher Diskussion verzichtet.

9.6. Literatur-Recherche

Die Literatur-Recherche hat den Zweck, dass allfällige Publikationen im gleichen oder einem ähnlichen Themenumfeld gefunden werden. Damit kann einerseits sichergestellt werden, dass die Ziele dieser Studienarbeit sich nicht mit bereits gewonnen Information überschneiden. Andererseits können relevante Publikationen unter Angabe der Quelle zur eigenen Wissensbildung weitverwendet werden.

9.6.1. Schlagwörter

Die Datenbanken wurden mit den in der folgenden Liste enthaltenen Schlagwörtern durchsucht. Diese wurden beliebig kombiniert und auch nötigenfalls noch minimal ergänzt. Die Suche wurde weiter zusätzlich zeitlich eingeschränkt, es wurden nur Publikationen nach dem Jahr 2004 beachtet.

- End User Experience
- End User Performance
- High Performance Websites
- Monitoring
- Website Performance Test
- Internet Performance Parameter
- Website First Impression OnLoad
- TCP HTTP
- Browser Comparison
- Page Load Time
- Website Testing Tools
- Episodes
- User Latency
- Web Timing Working Draft
- Page Speed
- Webpage
- Browser Rendering
- Optimize Websites
- Steve Souders
- Pipelining

9.6.2. Datenbanken

In den folgenden Datenbanken wurde nach relevantem Material für die Studienarbeit gesucht.

- ACM Digital Library [100]
- The Guide to Computing Literature [100]IEEE Xplore Digital Library [101]
- Web of Science [102]
- Inspec [103]
- Emerald [104]
- Google [105]

9.6.3. Resultate

Die Literatur-Recherche ergab am 01.10.2010 folgende Resultate.

Veröffentlichung	Titel	Autoren	Bemerkungen
Juni 2009	Even Faster Web Sites	- Steve Souders, Google	- Buch - Konzentriert sich auf Web 2.0 (AJAX) Performance - Beschreibung von Tools
Februar 2009	Improving Performance on the Internet	- Tom Leighton, Akamai Technologies	- Allgemeine Analyse der Flaschenhälse im Internet - Optimierungen der Performance, insbesondere CDN - Eventl. befangen da von Akamai
Dezember 2008	High-Performance Web Sites	- Steve Souders, Google	- ACM-Artikel zu seinem Buch - Kurzübersicht über Performance-Verbesserungen von Webseiten
März 2008	WebAccel: Accelerating Web access for low-bandwidth hosts	- Tae-Young Changa - Zhenyun Zhuangb - Aravind Velayuthamc - Raghupathy Sivakumar	- Browser-Geschwindigkeit bei geringer Bandbreite - Priorisieren von gewissen Anfragen
September 2007	Improving the Delivery of Multimedia Embedded in Web Pages	- Adam Serbinski, Ryerson University Toronto - Abdolreza Abhari, Ryerson University Toronto	- Performance-Gewinn durch Prefetching - Nur am Rande interessant für uns

Veröffentlichung	Titel	Autoren	Bemerkungen
September 2007	High Performance Web Sites	- Steve Souders, Yahoo!	- Buch - Aspekte der End User Experience - Performance-Verbesserungen von Webseiten - Beschreibung von Tools
April 2004	Improving the performance of client Web object retrieval	- Alexander P. Pons	- Performance-Gewinn durch Prefetching - Nur am Rande interessant für uns

Tabelle 17: Resultate der Literatur-Recherche

9.6.4. Beurteilung

Die Literatur-Recherche hat im Allgemeinen keine - direkt auf unsere Aufgabe - passende Ergebnisse geliefert. Die meisten in der Liste aufgeführten Publikationen haben einfach zu einer erweiterten Sichtweise auf die Problematik der Webseite Performance geführt.

Zwei Publikationen – die beiden Bücher von Steve Souders – heben sich von den anderen Publikationen ab. Diese sind sehr spezifische Fachliteratur über die Beschleunigung von Webseiten in Bezug auf die EUE. In „Even Faster Websites“ ist eine sehr ausführliche Auflistung von verschiedenen Tools zur Messung der EUE enthalten. Jedoch befassen sich beide Bücher sehr tief mit der Optimierung von Webseiten, was nicht das Hauptziel unsere Studienarbeit ist.

9.7. Tools

Damit die verschiedenen Browser-Aktivitäten beim Aufruf einer Webseite verfolgt werden können, bietet sich eine Reihe von Tools an. Einige davon klinken sich beim Browser direkt als Plugin ein, andere beschränken sich auf die Analyse des Netzwerkverkehrs, ob nun als Proxy oder als Netzwerk-Sniffer.

Diese Übersicht von Tools beschränkt sich auf die Aufzeichnungs-Funktionalität der verschiedenen Browser-Aktivitäten, welche sich für die Messung der EUE eignen (siehe Kapitel 2.5). Andere Tools zur Analyse und Optimierung der Webseite – wie Yahoo YSlow [106] oder Google PageSpeed [107] – werden nicht behandelt.

9.7.1. Firebug

Firebug [108] ist eine Erweiterung für den Browser Firefox, welche eine grosse Fülle an Funktionen bietet. Für Web-Entwickler ist der Firebug besonders wertvoll, da dieser eine schnelle Analyse einer Webseite zulässt und unter anderem auch das Debugging von JavaScript zulässt. Das für uns interessante Panel des Firebug ist das „Netzwerk". Beim Aufrufen einer Webseite werden die Ladezeiten der verschiedenen Webseitenelemente und auch Browser-Events aufgezeichnet. Diese Aufzeichnung wird übersichtlich in einem Wasserfall-Diagramm dargestellt, was eine komfortable Analyse der Browser-Aktivitäten erlaubt.

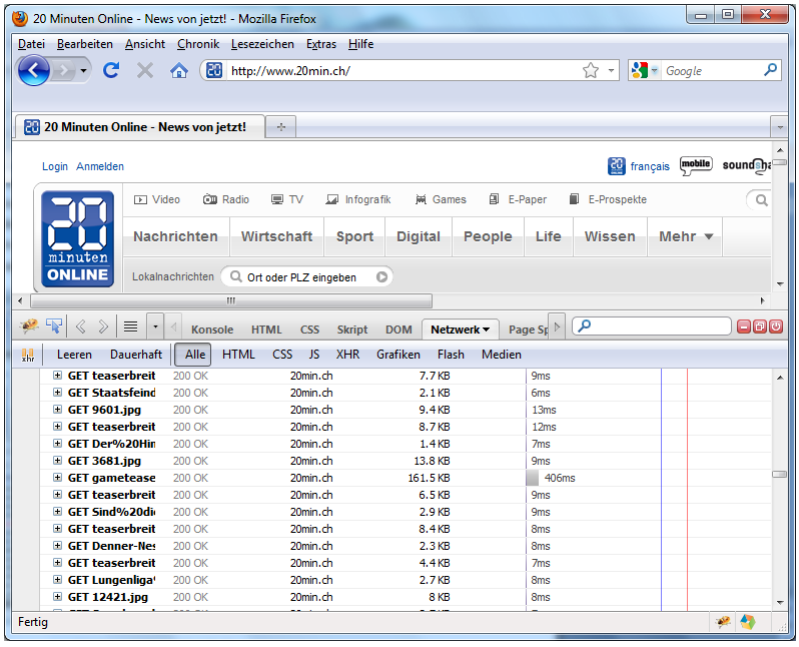


Abbildung 78: Screenshot Firebug

+ Gute Aufschlüsselung der verschiedenen Schritte beim Laden der einzelnen Elemente (DNS, Verbinden, Blockieren, Warten, Senden, Empfangen)
+ Viele zusätzliche Plugins wie YSlow oder Page Speed verfügbar
+ Gute Export-Möglichkeiten
- Keine Rendering-Angaben
- Nur für Firefox verfügbar

9.7.2. HttpWatch

Der HttpWatch [109] verrichtet seine Arbeit als Plugin, welche für die Browser Firefox und Internet Explorer verfügbar sind. Die Aufzeichnung ist sehr detailliert und wird wie bei Firebug mittels eines Wasserfall-Diagramms visualisiert. Ein grosser Vorteil des HttpWatch ist die Anzeige des ersten Renderings im Browser. Dadurch erhält man eine Richtgrösse, ab wann der Benutzer etwas zu sehen bekommt.

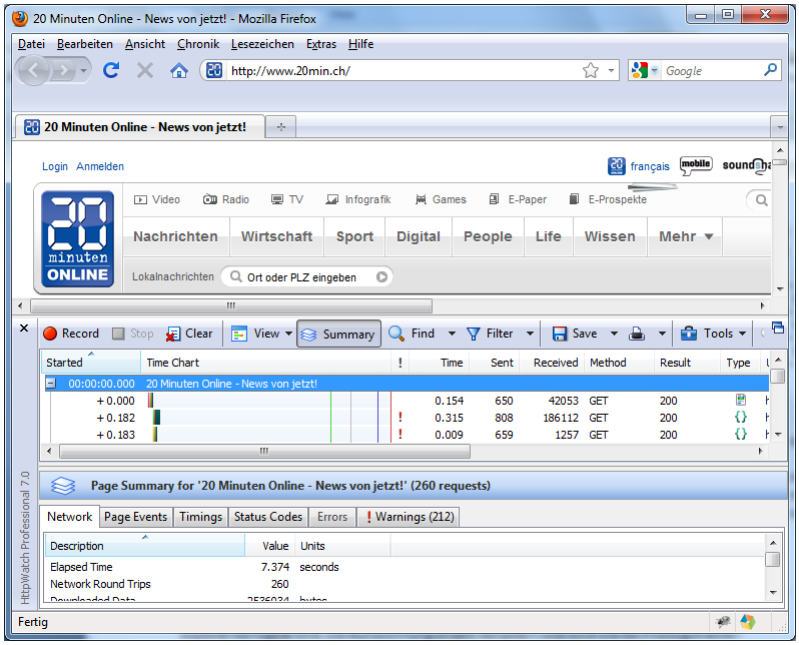


Abbildung 79: Screenshot HttpWatch

\+ Erfassung des ersten Rendering-Events
\+ Für Internet Explorer und Firefox verfügbar
\+ Gute Export-Möglichkeiten
\- Für Chrome nicht verfügbar
\- Kostenpflichtig (nur für 100 Top Alexis-Webseiten gratis)

9.7.3. dynaTrace AJAX Edition

Die AJAX Edition von dynaTrace [110] wurde speziell für die Messung der EUE entwickelt. Spezieller Augenmerk wurde in dieser Edition auf die Analyse der AJAX-Aktivität gelegt, wobei da die Möglichkeiten fast grenzenlos scheinen. Die Rendering-Aktivitäten werden sehr umfassend aufgezeichnet, was eine sehr detaillierte Analyse dessen erlaubt. Leider ist dynaTrace AJAX Edition nur als Plugin für den Internet Explorer und nicht auch für die anderen Browser verfügbar.

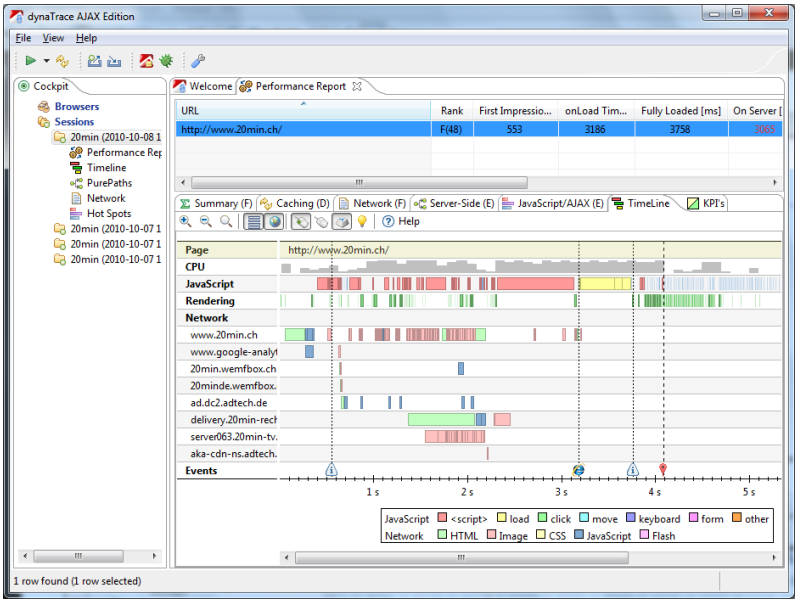


Abbildung 80: Screenshot dynaTrace AJAX Edition

+ Erfassung des ersten Rendering-Events
+ Sehr gute Aufzeichnung der AJAX-Aktivitäten
+ Gute grafische Darstellung in Diagrammen
- Unter den vielen aufgezeichneten Werten leidet die Benutzerfreundlichkeit des Tools
- Nur für Internet Explorer verfügbar

9.7.4. Microsoft Visual Round Trip Analyzer

Beim Microsoft Visual Round Trip Analyzer (MVRT) [111] handelt es sich um ein Analyse-Tool, welches auf Aufzeichnungen des Microsoft Network Monitor [112] basiert. Damit wird im Hintergrund der ganze Datenverkehr des Computers aufgezeichnet und beim Abschluss der Aufzeichnung werden die Daten im MVRT analysiert und grafisch dargestellt. Das Hauptaugenmerk dieses Tool liegt auf der Analyse der TCP-Aktivitäten, wobei unter anderem der Datendurchsatz über die Zeit aufgeschlüsselt wird und auch der Aufbau der verschiedenen Verbindungen beim Aufruf einer Webseite dargestellt wird.

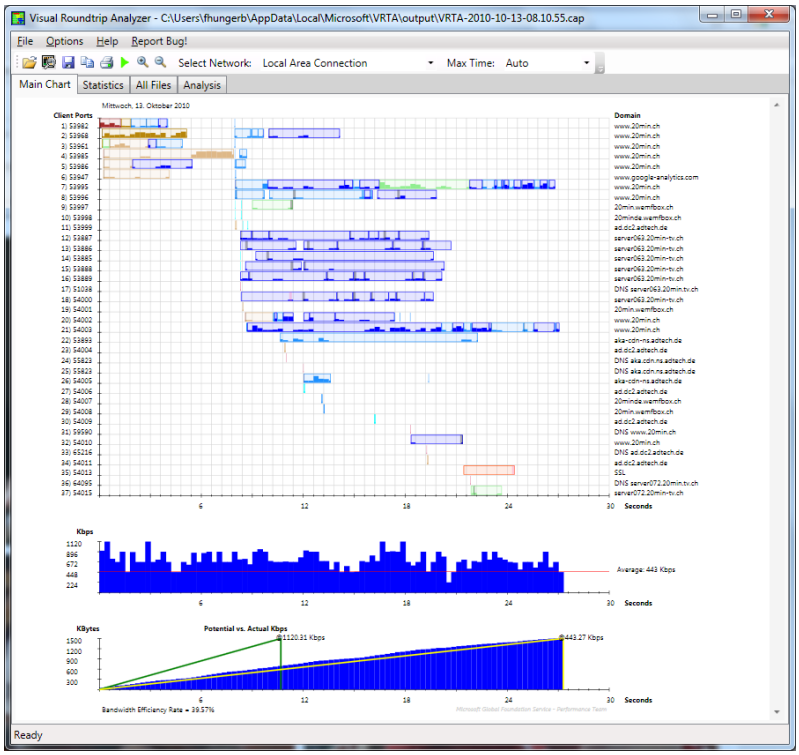


Abbildung 81: Screenshot Microsoft Visual Roundtrip Analyzer

+ Ausführliche TCP-Statistiken mit Grafiken unterlegt
+ Offene parallele Verbindungen sind gut ersichtlich
+ Browserunabhängig
- Browser-Events (Rendering usw.) werden nicht aufgezeichnet

9.7.5. Fiddler

Die Funktionsweise von Fiddler [113] unterscheidet sich von den anderen Tools. Fiddler agiert nicht transparent, sondern ist als Proxy implementiert. Der ganze HTTP-Verkehr fliesst durch Fiddler, wobei – was ein grosser Vorteil von Fiddler ist – die HTTP-Anfragen / - Antworten verändert werden können. Es ist möglich aufgrund von verschiedenen Kriterien Breakpoints zu setzen und dann manuell die Anfrage / Antwort anzupassen. Dies kann auch automatisiert mittels JavaScript erfolgen. Die TCP-Verbindungsparameter und das HTTP-Protokoll werden sehr detailliert ausgewertet.

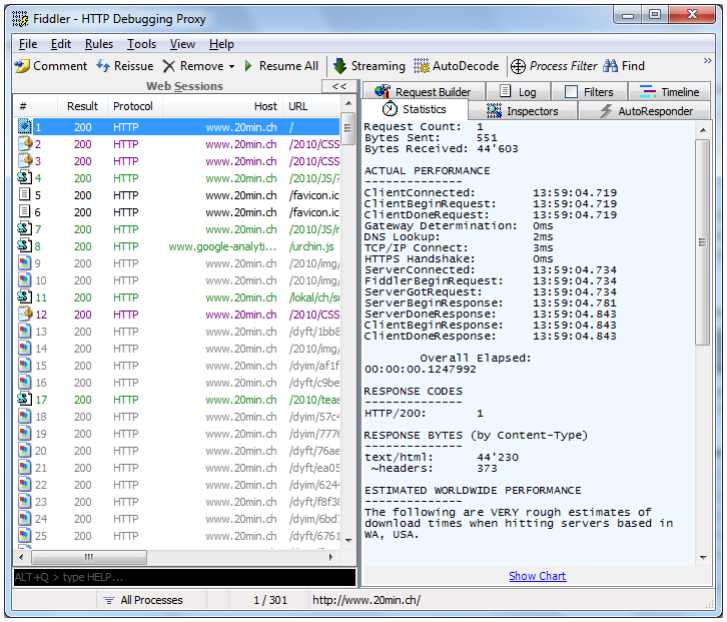


Abbildung 82: Screenshot Fiddler

+ Erlaubt das Setzen von Breakpoints / Manipulierung von HTTP-Anfragen
+ Detaillierte Auswertung von TCP / HTTP möglich
+ Gute Export-Möglichkeiten
+ Browserunabhängig
- Agiert als Proxy, wodurch unter Umständen das Resultat verfälscht werden könnte
- Browser-Events (Rendering usw.) werden nicht aufgezeichnet

9.7.6. Google Speed Tracer

Bei Google Speed Tracer [114] handelt es sich um ein Plugin für den Browser Chrome, wobei dieser als Pendant des Netzwerk-Panels des Firebug angesehen werden kann. Der Aufruf der Webseite wird auch in einem Wasserfall-Diagramm dargestellt, wobei zwischen Netzwerk- und Event-Ansicht gewechselt werden kann. Im oberen Teil des Speed Tracers wird eine optische ansprechende Grafik geboten, welche die Netzwerk- und Event-Belastung darstellt.

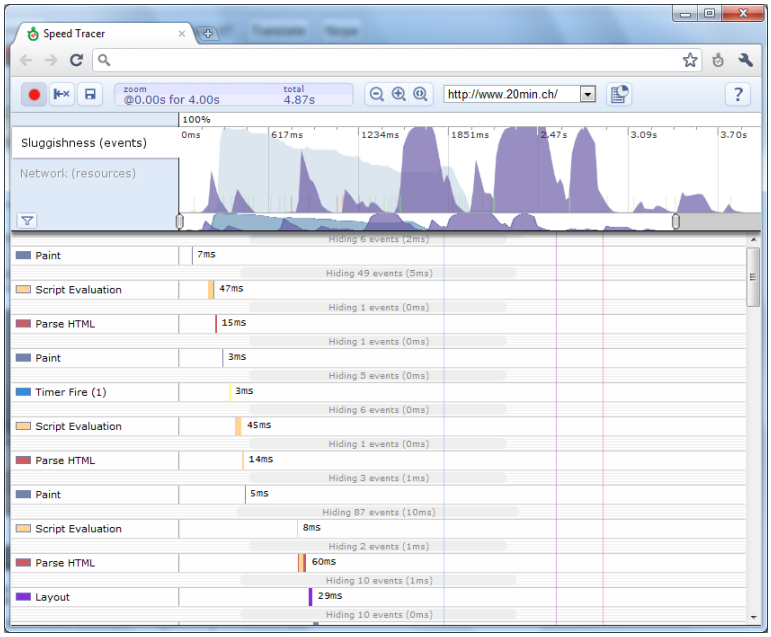


Abbildung 83: Screenshot Google Speed Tracer

\+ Gute Aufschlüsselung der verschiedenen Schritte beim Laden der einzelnen Elemente
\+ Ausführliche Event-Aufzeichnung, wobei ein Pendant zum HttpWatch-Event „Erstes Rendering“ fehlt
\- Schlechte Export-Möglichkeiten
\- Nur für Chrome verfügbar

9.7.7. Wireshark

Wireshark [115] zeichnet den gesamten Netzwerk-Verkehr auf, wobei jedoch keine speziellen Auswertungen für das HTTP-Protokoll vorhanden sind. Somit eignet sich Wireshark im Zusammenhang mit der EUE nur für die Analyse der TCP-Parameter. In diesem Bereich ist Wireshark aber praktisch unschlagbar.

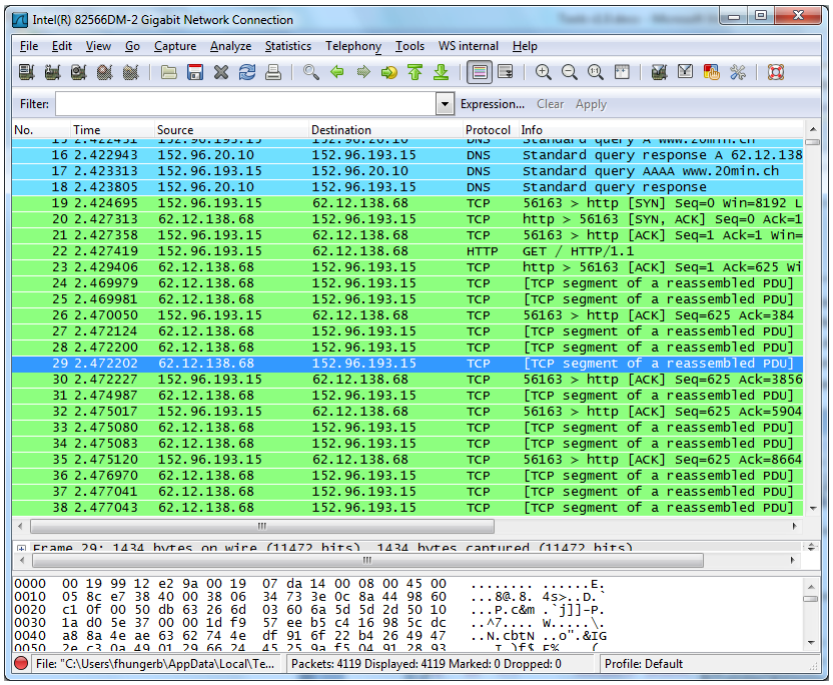


Abbildung 84: Screenshot Wireshark

\+ Detaillierte Auswertung von TCP möglich
\+ Browserunabhängig
\- Keine speziellen Auswertungen von HTTP möglich
\- Browser-Events (Rendering usw.) werden nicht aufgezeichnet

9.7.8. Vergleich

Damit die Eignung der Tools für die Messung der EUE bewertet werden kann, werden folgend die in diesem Kapitel vorgestellten Tools verglichen. Die untersuchten Events wie „DOM-bereit“ sind im Kapitel 2.5 näher beschrieben.

	Firebug	HttpWatch	dynaTrace AJAX Edition	Microsoft Visual Round Trip Analyzer	Fiddler	Google Speed Tracer	Wireshark
Browser (-Integration)							
Firefox	x	x		x	x		x
Internet Explorer		x	x	x	x		x
Chrome				x	x	x	x
Events							
DOM-bereit	x	x				x	
Erstes Rendering		x	x				
Erste Anzeige							
Seite benutzbar							
OnLoad-Event	x	x	x			x	
HTTP-Load	x	x	x			x	

Tabelle 18: Tools-Vergleich

9.7.9. Fazit

Es existiert keine einziges Tool, welches alle Anforderungen für die Messung der EUE erfüllt. Die Aufzeichnung der Events „Erste Anzeige“ und „Seite benutzbar“ ist in keinem Tool möglich, da diese beiden Werte auch sehr Webseiten-abhängig sind. Ein Browser-Vergleich wird dadurch erschwert, dass für kein Tool ein Plugin für alle Browser vorhanden ist.

Für einen Browser-Vergleich erschwert, dass kein einziges Tool Plugins für alle drei Browser liefert.

Das Tool mit der grössten Abdeckung an Messwerten ist der HttpWatch. Dieser wurde im Rahmen dieser Arbeit angeschafft und für die Messungen im Internet Explorer sowie Firefox verwendet. Für Chrome wurde der Google Speed Tracer verwendet, da dieser die einzige Alternative darstellt.

9.8. Messsetup

In diesem Kapitel wird die genaue Konfiguration des für die Messungen verwendeten Computers aufgezeigt.

Es wurde entschieden, für die Messungen virtualisierte Computer zu verwenden. Diese Wahl wurde aus folgenden Gründen getroffen:

- VM (Virtuelle Maschine) kann für später gesichert werden
- VMs ist portabel und kann auch auf einem anderen System für Tests genutzt werden
- Für spätere Tests müssen nicht zuerst noch die zwei Testsysteme eingerichtet werden
- Snapshot Möglichkeit als einfache Wiederherstellungsmöglichkeit eines früheren Zustands

Aufgrund der bereits vorhandenen Erfahrung mit VMware Produkten, wurde VMware Workstation als Virtualisierungssoftware für die Mess-Computer gewählt. Als Host System wurden die von der HSR zur Verfügung gestellten PCs benutzt (siehe Tabelle 19). Die schematische Darstellung ist in Abbildung 86 sehen.

9.8.1. Computerkonfiguration

Da die meisten Leute daheim noch eine 32-bit (x86) Windows Version verwenden, wurde auch für die Messungen diese Version installiert. Nur auf dem Host wurde die 64-bit Version installiert, um die ganzen 4 GB RAM nutzen zu können. Alle Software wurde in der englischen Version mit den Standardeinstellungen installiert, sofern keine weiteren Angaben dazu gemacht wurden. In der Windows 7 VM wurde von Aero zum Basic Theme gewechselt, um unnötige grafische Berechnungen zu vermeiden.

Alle Betriebssysteme wurden mittels Windows Update auf die aktuellste Version gebracht. Anschliessend wurde der Update-Dienst deaktiviert, um Versionsänderungen zu verhindern und Netzwerkverkehr neben den Messungen zu vermeiden. Weiterhin wurde der automatische Scan des Windows Defenders auf dem Host und dem Gast System deaktiviert und auf die Installation eines Virenscanners auf Performancegründen verzichtet.

	Host	**Gast Windows 7**	**Gast Windows XP**
Version OS	Enterprise 64-bit	Professional 32-bit	Professional 32-bit, SP3
Patch Stand	24.11.2010	24.11.2010	24.11.2010
CPU	Intel Core 2 Duo E6750 2.66 GHz, 1333 MHz FSB		
RAM	4 GB	1536 MB	1024 MB
Netzwerk	GigaBit (Intel E1000)	Bridge Mode	Bridge Mode
Video	ATI FireGL V5200		
Modell	Fujitsu Siemens Celsius W360		
Hypervisor	VMware Workstation 7		

Tabelle 19: Computer Details

VMware Workstation wurde so konfiguriert, dass es das ganze virtuelle RAM der Gast Systeme ins reservierte RAM des Hosts abbildet (siehe Abbildung 85). Dies verbessert im Vergleich zu den anderen beiden Optionen die Performance.

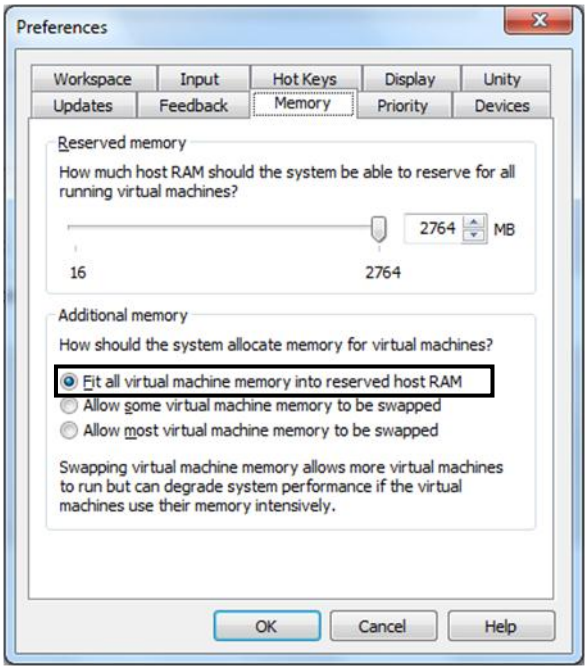


Abbildung 85: Screenshot VMware Workstation Konfiguration

9.8.2. Software für Messungen

In der Tabelle 20 sind die installierten Browserversionen inkl. der verwendeten Plugins in den VMs (Mess-Computer) aufgelistet. Dabei handelt es sich jeweils um die 32-bit Version.

	Internet Explorer	**Firefox**	**Chrome**
Version	8.0.7600.16385 (Win 7) 8.0.6001.18702 (Win XP)	3.6.12	7.0.517.44 (Official Build 64615)
Plugins	HttpWatch Pro 7.0.26 dynaTrace 2.0.0.574 Adobe Flash 10.1.102.64 JRE[6] v6 Update 22	HttpWatch Pro 7.0.26 Firebug 1.5.4 Adobe Flash 10.1.102.64 JRE v6 Update 22	Speed Tracer 0.18 Adobe Flash 10.1.102.64 JRE v6 Update 22

Tabelle 20: Browser Versionen

Die aktuellste Internet Explorer Version wurde mit dem Windows Update bezogen, als das Gast OS mit allen Patches versorgt wurde. Nach der Installation der Browser wurden die automatischen Updates bei allen Browsern deaktiviert.

[6] JRE = Java Runtime Environment

Spezialität bei Google Chrome Plugin Speed Tracer

Damit das Speed Tracer Plugin bei Google Chrome richtig funktioniert, muss Chrome mit folgendem Argument gestartet werden: „--enable-extension-timeline-api“

Damit die Aufzeichnungen der Messresultate des Plugins später wieder betrachtet werden können, muss für das Speed Tracer Plugin der Zugriff aufs Dateisystem wie folgt erlaubt werden:
Chome Menü -> Tools -> Extensions -> Speed Tracer: „Allow access to file URLs“
Die erzeugten Aufzeichnungen liegen im HTML Format vor und können nur im Chrome mit installiertem Speed Tracer Plugin betrachtet werden. Nach dem Öffnen der Datei muss auf den Button „Open Monitor!“ geklickt werden.

Nachfolgend ist eine Auflistung der Softwareversionen zu finden, welche für den Betrieb der VMs bzw. darin installiert wurden neben den bereits erwähnten Browsern.

Produkt	**Version**
VMware Workstation	7.1.3-324285
VMware Tools	8.4.5 b324285
Wireshark	1.4.2
Microsoft Visual Round Trip Analyzer	v3.0.0250.1024

Tabelle 21: installierte Software für die virtuellen Maschinen

9.8.3. Dateinamenkonvention für Messergebnisse

Die Messergebnisse, welche mit den Tools HttpWatch, Speed Tracer, Wireshark sowie dynaTrace aufgezeichnet wurden, wurden jeweils exportiert und im Dateisystem abgelegt. Für eine einfache Zuordnung der Messungen zu den eingestellten Netzwerkparametern wurden die Dateien gemäss der im Folgenden erklärten Notation benannt (Syntax gemäss [92]).

Der Dateinamen setzt sich aus dem verwendeten OS und Browser sowie den Netzwerkparametern wie folgt zusammen:

<OS>-<Browser>-<Downloaddatenrate>-<RTT>-<TestNr>

	Ausprägungen und Bedeutung		
OS	xp = Windows XP	7 = Windows 7	
Browser	ch = Chrome	ie = Internet Explorer	ff = Firefox
Downloaddatenrate	Downloaddatenrate in kbit/s (Werte gemäss Tabelle)		
RTT	Round Trip Time in ms (Werte gemäss Tabelle 1)		
TestNr	Nummer der Messung, 3 Messungen pro Seite und NetEm Einstellung		

Tabelle 22: installierte Software für die virtuellen Maschinen

Der konkrete Aufbau des Dateinamens sieht also wie folgt aus:

{7|xp}-{ch|ie|ff}-{1000|20000}-{15|60|240|600}-[1|2|3]

9.8.4. Netzwerkkonfiguration

Im Kapitel 2.4 wurde bereits grob dargestellt, wie die NetEm-Box eingebunden wird. Die genaue Funktionsweise des Emulators ist im Kapitel 6 zu finden.

Alle Messungen wurden an der HSR und über deren Internetanschluss durchgeführt. Dabei handelt es sich gemäss eigenen Messungen um einen synchronen 100 Mbit/s Internet-Anschluss (siehe Tabelle 10).

Konkret erfolgte der Anschluss über das Notebook Netz der HSR, da nur dort eigene Geräte wie die NetEm-Box angeschlossen werden dürfen. Bei diesem Zugriff aufs Internet wird jedoch ein Traffic Shaping durch die Hochschule vorgenommen. Auf Antrag wurde jedoch für die Arbeit eine Ausnahme gemacht und die IP der NetEm-Box vom Shaping ausgenommen.

Ein weiterer wichtiger Punkt ist, dass die HSR keinen Proxy einsetzt. Dies würde das ganze Ladeverhalten der Browser aufgrund des Cachings der Inhalte beeinflussen.

Schematisch das Messsetup wie folgt aus:

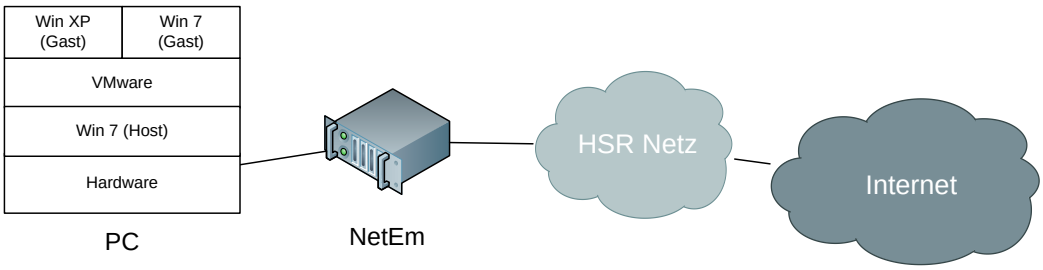


Abbildung 86: Konkretes Messsetup

9.8.5. Messvorgehen

Die Abbildung 87 zeigt schematisch das Vorgehen bei den Messungen.

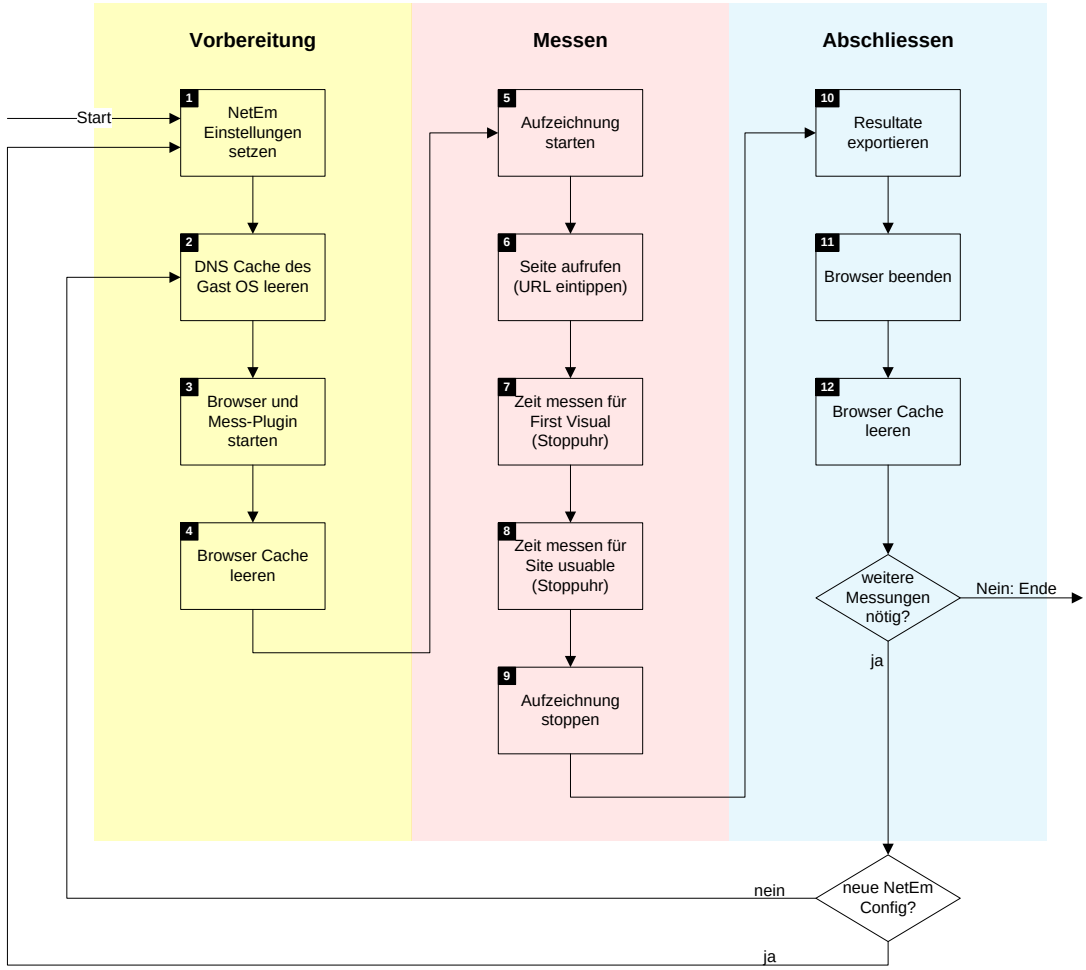


Abbildung 87: Vorgehen Messung

Für alle Browser ist der Ablauf in etwa gleich. Der grösste Unterschied besteht im verwendeten Plugin für die Aufzeichnung. Ein weiterer Unterschied ist, dass der Cache bei Schritt 4 im IE und Firefox durch das HttpWatch Plugin geleert wird und bei Chrome manuell.
Bei Chrome muss der Browser Cache bereits vor dem Beenden des Browsers geleert werden (Schritt 12), da er sonst zu Beginn direkt einige Daten zu den zuletzt geöffneten Seiten lädt wie z.B. die DNS Auflösung. Bei IE und Firefox ist dieser Schritt nicht nötig.

Die mit HttpWatch aufgezeichneten Daten wurden als *.hwl Datei gespeichert. Dabei handelt es sich um ein proprietäres Dateiformat. Das von vielen anderen Tools unterstützte *.har Format (HTTP Archive) wird auch unterstützt, doch werden darin nicht alle Werte abgespeichert.
Nach den Messungen wurden die exportieren Resultate geöffnet und dort die folgenden Werte ausgelesen:

- IE / Firefox: Erstes Rendering (Render Start) und Page Load (OnLoad-Event)
- Chrome: OnLoad-Event

Folgende Werte wurden von Hand mit einer Stoppuhr gemessen:

- Erste Anzeige
- Seite benutzbar

Diese Werte mussten notgedrungen von Hand gemessen werden, da sie von keinem der getesteten Tools bestimmt werden konnten. Die Genauigkeit von Stoppuhr Messungen von geübten Personen liegt bei ca. ± 0.1s [89]. Eine automatische Auswertung der „Ersten Anzeige“ wäre mit einigem Aufwand möglich gewesen, doch reichte dafür die Zeit nicht. Spätestens bei „Seite benutzbar“ ist jedoch eine automatische Erkennung kaum mehr möglich, da dies sehr vom Seitenaufbau abhängt.

Der obige Ablauf wurde pro Kombination von Browser, Betriebssystem und Netzwerkparametern drei Mal durchlaufen und von den gemessenen Werten wurde jeweils der Durchschnitt genommen für die Auswertung. Diese Massnahme war nötig, um Streuungen der Werte abschätzen zu können und um offensichtliche Ausreisser zu erkennen. Offensichtliche Ausreisser gingen nicht in die Statistik ein und es wurde erneut gemessen. Für weitere Angaben zur Messgenauigkeit sei auf das Kapitel 9.8.6 verwiesen. Die genaue Definition der obengenannten Events ist im Kapitel 2.5 zu finden.

Je nach Bedarf wurde auch eine Messung mit Wireshark und dynaTrace gemacht. Vor Aufruf der Webseite wurde der Netzwerksniffer Wireshark gestartet und nach dem OnLoad-Event wurde die Aufzeichnung gestoppt. Die dynaTrace Messung gibt es nur für den IE, da dieses Tool nur für den IE verfügbar ist. Diese Messungen wurden primär für die Analyse der Webseiten durchgeführt, da dort der Aufbau der Seite bzgl. MIME Typen und Hostnamen ausgewertet wird. Mit der Wireshark Aufzeichnung ist es möglich, den Verlauf der RWin zu verfolgen während des Seitenabrufs über alle aufgebauten TCP Verbindungen. So kann das Verhalten diesbezüglich von Windows XP und Windows 7 verglichen werden.

Zur Genauigkeit der Messungen von HttpWatch und Speed Tracer konnten keine genauen Angaben gefunden werden. Auf der HttpWatch Webseite wird erwähnt, dass sich die Software nur minimal auf die Interaktion zwischen Browser und Webseite auswirkt. Dasselbe ist beim Speed Tracer Plugin zu erwarten. Beide Tools geben den Zeitpunkt der Events auf Millisekunden genau an. Da kann angenommen werden, dass die Werte mindestens auf eine Hundertstelsekunde genau sind.

9.8.6. Messgenauigkeit

Zusammengefasst machen bei den Messungen folgende Faktoren die Ungenauigkeit bzw. die Schwankungen der Messwerte aus:

- Zeitmessung von Hand mit der Stoppuhr: ± 0.1s
- Schwankungen im Netzwerk sowie in der Serverbelastung: ± 50ms
- Veränderungen der Webseite selber (Datenmenge, Anz. Objekte)
  (siehe Kapitel 0 und 7.2.3 für Details)

Es wurden für jede Kombination von Betriebssystem, Browser, Netzwerkparametern und Webseite drei Messungen durchgeführt. Offensichtliche Ausreisser wurden nicht notiert, sondern es wurde erneut gemessen. Anschliessend wurde pro Kombination für jeden EUE-Event der Durchschnitt sowie die Standardabweichung berechnet.

Für ein aussagekräftiges Resultat wären natürlich noch mehr Messungen nötig gewesen, doch der so berechnete Durchschnitt und die Standardabweichung geben eine ungefähre Genauigkeit der Messungen an. Die oben genannten Schwankungen sollten durch die Anzahl Messungen abgedeckt sein und in der berechneten Standardabweichung mit einbezogen sein. Eine separate Betrachtung ist deshalb nicht nötig. Der Vollständigkeit halber wurden die Veränderungen der Webseiten trotzdem in den Kapiteln 0 und 7.2.3 festgehalten.

Da bei der Auswertung (siehe Kapitel 7.3) mehrere Messwerte (die Durchschnitte) zusammen verrechnet wurden, müssen auch die dazugehörigen Standardabweichungen berücksichtigt werden. Um für das Resultat ebenfalls eine Standardabweichung anzugeben, müssen die Standardabweichungen der einzelnen Messungen wie folgt kombiniert werden [90]:

$$\sigma\,(\text{total}) = \sqrt{\frac{\sum_{i=1}^{n}(\sigma_i{}^2)}{n}}$$

Formel 5: Berechnung der Gesamt-Standardabweichung

Es wird die Quadratsumme der Standardabweichungen der einzelnen Messwerte gebildet und durch die Anzahl der einzelnen Messwerte geteilt. Anschliessend wird davon Wurzel gezogen.

In der Excel Datei, wo die Messwerte ausgewertet wurden, wurde diese Formel für alle einzelnen Messwerte des Internet Explorers pro getestete Webseite angewandt. Es wurde angenommen, dass die dadurch berechnete totale Standardabweichung bei dieser Webseite auch für die anderen Browser gilt.

9.8.7. NetEm -Einstellungen

Bei der NetEm-Konfigurations-Webseite werden die Parameter „Bandwith“ und „Delay Time“ so gesetzt, dass beim Durchführen des cnlab Performance Test aus der VM heraus die Werte erscheinen, welche in Tabelle 11 aufgelistet sind. Die beim NetEm eingestellten Werte sind in der untenstehenden Tabelle zu finden. Dabei wurde darauf geachtet, dass der effektiv erreichte Wert nicht mehr als 10% vom gewünschten abweicht.

	Gewünscht	Eingestellt bei NetEm	Effektiv erreicht
Download 1000 kbit/s	1‘000	1‘060	1‘013
Download 20‘000 kbit/s	20‘000	21‘170	20‘019
Upload 100 kbit/s	100	110	106
Upload 2000 kbit/s	2‘000	2‘070	1‘997
RTT 15ms	15	5	15
RTT 60ms	60	27	59
RTT 240ms	240	116	237
RTT 600ms	600	296	598

Tabelle 23: NetEm Parameter für Netzwerkparameter von Tabelle 11

Die Werte für die Antwortverzögerung (Delay Time) müssen beim NetEm auf beiden Interfaces eingestellt werden.

Dabei ist zu beachten, dass beim cnlab Performancetest mit zunehmender eingestellter Verzögerung die eingestellte Bandbreite nicht mehr erreicht wird. Dies hat damit zu tun, dass beim Test die Downloadgeschwindigkeit über 5s gemessen wird und bei einer hohen RTT der grösste Teil dieser Zeit für den Slow Start draufgeht. Dies ist sehr schön in der Abbildung 88 sehen.

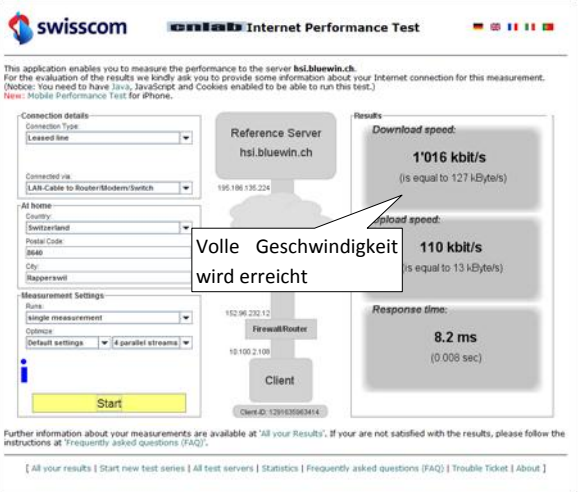
Für die Abbildung 88 wurde der cnlab Internet Performance Test ausgeführt und dabei der Datenverkehr mit Wireshark aufgezeichnet. Der Test wurde zwei Mal durchgeführt, jeweils mit anderen Netzwerkparametern:

1. Bandbreite: 1000/100 kbit/s (Download/Upload)
   RTT: 8 ms
2. Bandbreite: 1000/100 kbit/s (Download/Upload)
   RTT: 600 ms

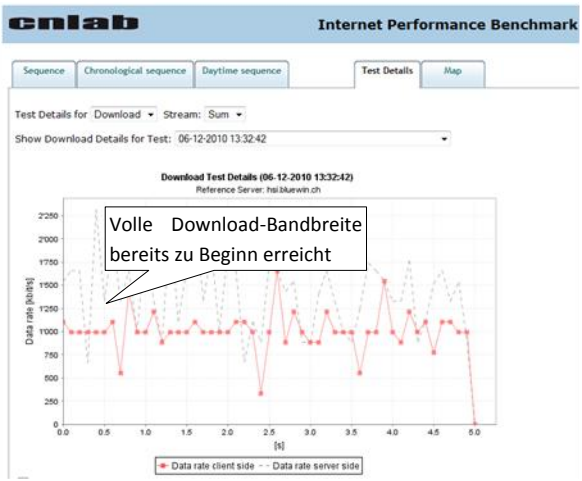
In der folgenden Gegenüberstellung der Screenshots (Abbildung 88) sind die Unterschiede sehr gut zu erkennen. In der ersten Reihe sind die Resultate des Tests zu sehen und in der zweiten Reihe handelt es sich um die Detailauswertungen der Messung. In dieser steht die gestrichelte Linie für die auf der Serverseite gemessene Geschwindigkeit und die durchgezogene Linie für den auf dem Client gemessenen Wert. Die Datenrate auf dem Server schwankt stark, da dieser in einen Buffer sendet und erst durch diese Zwischenspeicherung die Schwankungen ausgeglichen werden.

In der untersten Grafik, dem mit Wireshark erzeugten tcptrace, sieht man auf der rechten Seite den TCP Slow Start sehr deutlich. Der Test wird sogar ca. 600ms später abgebrochen, da aus Clientsicht erst nach einer RTT die Daten vom Testserver eintreffen und die Messung erst dann beginnt.

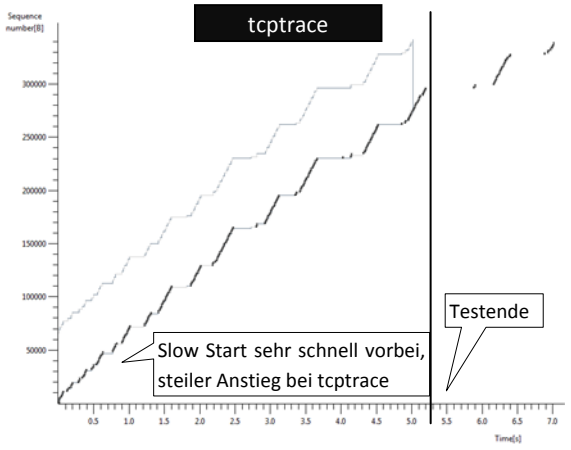
RTT = 8 ms
Bandbreite = 1000/100 kbit/s



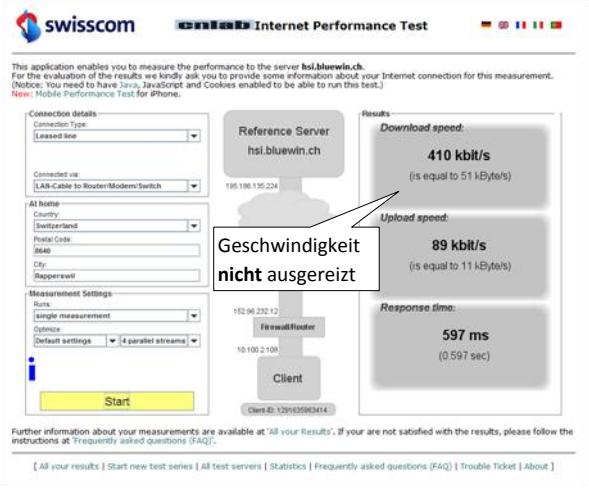
Details Downloadmessung



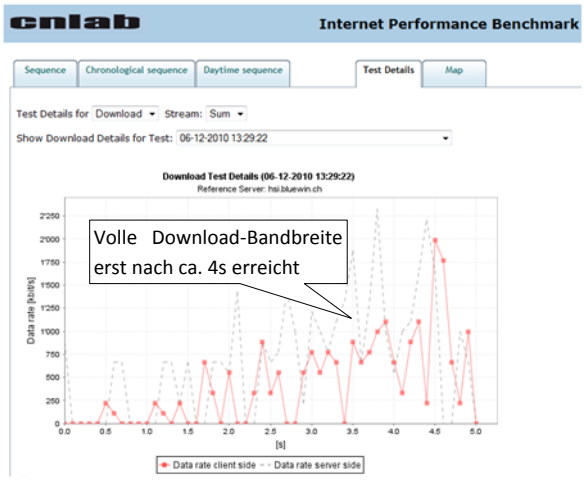
tcptrace



RTT = 600 ms
Bandbreite = 1000/100 kbit/s



Details Downloadmessung



tcptrace

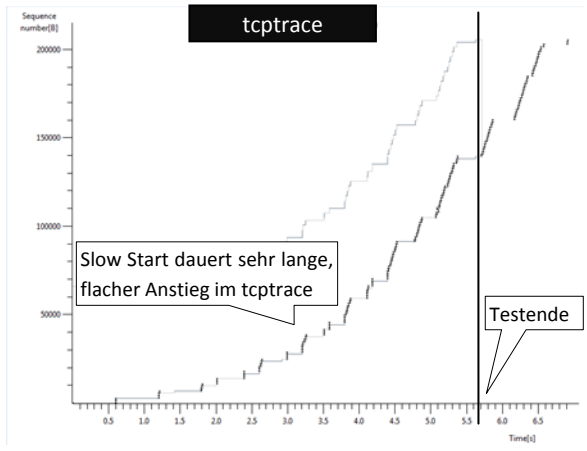


Abbildung 88: Vergleich Internet Performance Messung mit verschiedenen RTTs

9.9. Protokolle

Die Protokolle der wöchentlichen Sitzungen sind auf der CD zu finden.

9.10. Verzeichnisse

9.10.1. Glossar

Genauere Informationen zu den Begriffen können unter www.wikipedia.org gefunden werden.

ACK	Acknowledge (Bestätigung)
Auto Tuning	Automatische Einstellung
Bucket	Behälter
Buffer	Zwischenspeicher
Burst	Kurzzeitige höhere Datenrate (höher als der Durchschnitt)
Cache	Zwischenspeicher
CDN	Content Distribution Network
Congestion	Verstopfung
Congestion Avoidance	Verstopfungsvermeidung
CPU	Central Processing Unit (Prozessor)
CSS	Cascading Style Sheets
cwnd	TCP Congestion Window
Delayed ACK	Verzögertes ACK
DF-Flag	Don't Fragment Flag
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System
DOM	Document Object Model
DSL	Digital Subscriber Line
Embedded-System	Eingebettetes System, dediziert für eine spezielle Aufgabe
EUE	End User Experience (Endbenutzererlebnis/Endkundenerlebnis)
Event	Ereignis
Fast Retransmit	Schnelles erneutes Senden
FF	Firefox
Framework	Fachwerk/Rahmenstruktur, Programmiergerüst
GET-Anfrage	Befehl des HTTP Protokolls, englisch: GET-Request
htb	Hierarchical Token Bucket
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IE	Internet Explorer
Injection	Injektion
ISP	Internet Service Provider
Jitter	Variation der Verzögerung
JRE	Java Runtime Environment
k.A.	keine Angabe
Keep-Alive	Wach halten, Verbindung aufrecht erhalten
MIME	Multipurpose Internet Mail Extensions
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
MTU Path Discovery	MTU Bestimmung für einen bestimmten Pfad
NetEm	Netzwerk-Emulator
netem	Qdisc für Verzögerung, Jitter, Paketverlust /-duplizierung usw.
Overhead	Mehraufwand
Payload	Nutzlast
Pipelining	Befehlsverknüpfung

Plugin	Erweiterungsmodul
PPP	Point-to-Point Protocol
Prefetching	Etwas im Voraus laden
Qdisc	Queuing Discipline, Warteschlagedisziplin
QoS	Quality of Service, Servicequalität
Receive Window	Empfangsfenster (TCP)
Rendering	Bei Browser: Webinhalte in Bildschirmdarstellung darstellen
Retransmit	Erneute Übermittlung
RTO	TCP Retransmission Timeout
RTT	Round Trip Time (Paketumlaufzeit, Antwortzeit)
RWin	TCP Receive Window
ssthresh	Slow Start Threshold (Slow Start Schwellwert)
Stream	kontinuierlicher Datenstrom
tbf	Token Bucket Filter, ein Qdisc für Traffic Shaping
TCP	Transmission Control Protocol
Three-Way-Handshake	Drei-Wege-Handschlag, bei TCP Verbindungsaufbau
Timeout	Zeitüberschreitung
Traffic Control	Verkehrskontrolle (im Netzwerk)
Traffic Shaper	Gerät für Traffic Shaping
Traffic Shaping	Bandbreitenlimitierung
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VM	Virtuelle Maschine
VoIP	Voice over IP
W3C	World Wide Web Consortium
WLAN	Wireless Access Network
WWW	World Wide Web, Internet

9.10.2. Quellen

[1] cnlab Internet Performance Test, 21.10.2010
http://hsi.bluewin.ch/speedtest/

[2] The Experience of Broadband Speeds, Peter Heinzmann, Leslie Haddon, Thomas Bruhin, Anna Vettori, COST298 Projekt, 2010

[3] Web metrics: Size and number of resources, 17.12.2010
https://code.google.com/intl/de-CH/speed/articles/web-metrics.html

[4] High Quality on Youtube, 19.12.2010
http://blog.jimmyr.com/High_Quality_on_Youtube_11_2008.php

[5] Video Streaming Need To Know: Part 1- Encoding, Bit Rates and Errors, 19.12.2010
http://www.tomsguide.com/us/video-streaming-need-to-know-part-1,review-760-7.html

[6] packet-loss.de - Fazit, 19.12.2010
http://www.packet-loss.de/fazit.php

[7] StatCounter Glocal Stats, Top 5 Browsers from Sep 09 to Sep 10, 06.10.2010
http://gs.statcounter.com/#browser-ww-monthly-200909-201009
http://gs.statcounter.com/#browser-eu-monthly-200909-201009

[8] StatCounter Glocal Stats, Top 5 Operating Systems from Sep 09 to Sep 10, 06.10.2010
http://gs.statcounter.com/#os-ww-monthly-200909-201009
http://gs.statcounter.com/#os-eu-monthly-200909-201009

[9] Performance Test: Statistics, OS share, 06.10.2010
http://www.cnlab.ch/speedtest/stats.jsp#tab5

[10] Velocity Konferenz, 19.12.2010
http://www.velocityconf.com

[11] Speed Matters, 19.12.2010
http://googleresearch.blogspot.com/2009/06/speed-matters.html

[12] A Faster Web - It's Not about the Network Anymore, 19.12.2010
http://www.readwriteweb.com/archives/a_faster_web_-_its_not_about_the_network_anymore.php

[13] Fast by default?, 19.12.2010
http://omniti.com/seeds/fast-by-default

[14] Steve Souders, 19.12.2010
http://stevesouders.com/bio.php

[15] Browserscope, 14.12.2010
http://www.browserscope.org

[16] Google Page Speed, 19.12.2010
http://code.google.com/intl/de-DE/speed/page-speed/

[17] Google's Obsession With Speed Comes to the Web Server, 19.12.2010
http://www.readwriteweb.com/archives/googles_obsession_with_speed_comes_to_the_web_serv.php

[18] SPDY, 19.12.2010
http://en.wikipedia.org/wiki/SPDY

[19] SPeeDY – twice as fast as HTTP, 19.12.2010
http://www.stevesouders.com/blog/2009/11/12/speedy-twice-as-fast-as-http/

[20] Google adds site speed to search ranking, 19.12.2010
http://www.stevesouders.com/blog/2010/04/09/google-adds-site-speed-to-search-ranking/

[21] Ensuring Web Site Performance - Why, What and How to Measure Automated and Accurately, 24.09.2010
http://blog.dynatrace.com/2010/01/13/ensuring-web-site-performance-why-what-and-how-to-measure-automated-and-accurately/

[22] Are you ready for this, 08.10.2010
http://www.hunlock.com/blogs/Are_you_ready_for_this

[23] JavaScript-Event onLoad und die bessere Alternative, 24.09.2010
http://phpperformance.de/javascript-event-onload-und-die-bessere-alternative/

[24] Move Over onLoad. onDomReady Is Here, 08.10.2010
http://www.codestore.net/store.nsf/unid/BLOG-20070604

[25] Episodes Whitepaper, 07.10.2010
http://stevesouders.com/episodes/paper.php

[26] The many faces of end-user experience monitoring, 06.10.2010
http://blog.dynatrace.com/2010/01/18/week-2-the-many-faces-of-end-user-experience-monitoring/

[27] Navigation Timing, Editor's Draft October 5, 2010
https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/NavigationTiming/Overview.html

[28] Transmission Control Protocol, 22.10.2010
http://de.wikipedia.org/wiki/Transmission_Control_Protocol

[29] Transmission Control Protocol, 22.10.2010
http://en.wikipedia.org/wiki/Transmission_Control_Protocol

[30] Microsoft Windows Server 2003 TCP/IP Implementation Details, Version 2.4
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=06c60bfe-4d37-4f50-8587-8b68d32fa6ee&displaylang=en

[31] Path MTU Discovery, 17.12.2010
http://msdn.microsoft.com/en-us/library/ms817967.aspx

[32] TCP Maximum Segment Size (MSS) and Relationship to IP Datagram Size, 20.10.2010
http://www.tcpipguide.com/free/t_TCPMaximumSegmentSizeMSSandRelationshiptoIPDatagra.htm

[33] RFC 1191. 17.12.2010
http://tools.ietf.org/html/rfc1191

[34] RFC 2988, 22.10.2010
http://www.rfc-editor.org/rfc/rfc2988.txt

[35] RFC 1323, 22.10.2010
http://www.ietf.org/rfc/rfc1323.txt

[36] TCP Analysis - Section 6:TCP Options, 20.10.2010
http://www.firewall.cx/tcp-analysis-section-6.php

[37] I. Psaras, and V. Tsaoussidis, "The TCP Minimum RTO Revisited", IFIP NETWORKING, 2007
http://www.intersys-lab.org/media/papers/2007/networking07.pdf

[38] 4.2.3.2 When to Send an ACK Segment, 20.10.2010
http://webee.technion.ac.il/labs/comnet/netcourse/CIE/RFC/1122/110.htm

[39] Peter Heinzmann, HSR, "12.4 Transmission Control Protocol (TCP)", Vorlesung Computernetze 1 vom HS 2009

[40] Delayed Acknowledgments, 20.10.2010
http://msdn.microsoft.com/en-us/library/aa505957.aspx

[41] Computer Networks/Congestion Control, 20.10.2010
http://en.wikibooks.org/wiki/Computer_Networks/Congestion_Control

[42] tcptrace, zu finden im Wireshark unter Statistics -> TCP Stream Graph -> Time/Sequence Graph (tcptrace)
beruht auf dem Tool tcptrace von www.tcptrace.org
Eine genauere Beschreibung des Tools ist zu finden unter [6, S. 31]

[43] TCP Slow Start und Congestion Avoidance, 29.10.2010
http://www.teialehrbuch.de/Kostenlose-Kurse/Internet-Technik/16220-TCP-Slow-Start-und-Congestion-Avoidance.html

[44] TCP congestion avoidance algorithm, 03.11.2010
http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm

[45] Roman Dunaytsev, Yevgeni Koucheryavy, Jarmo Harju, "The Impact of RTT and Delayed ACK Timeout Ratio on the Initial Slow Start Phase", Institute of Communications Engineering, Tampere University of Technology, Finland, 2005
http://www.cs.tut.fi/~dunaytse/papers/ipsmome2005.pdf

[46] Next Generation TCP/IP-Stack unter Windows Vista und Windows Server Longhorn, 10.11.2010
http://www.microsoft.com/germany/technet/datenbank/articles/600902.mspx

[47] TCP/IP-Protokolle und Netzwerkkomponenten der nächsten Generation, 10.11.2010
http://technet.microsoft.com/de-de/library/cc754287%28WS.10%29.aspx

[48] Compound TCP, 10.11.2010
http://en.wikipedia.org/wiki/Compound_TCP

[49] Automatische Optimierung des TCP-Empfangsfensters, 10.11.2010
http://technet.microsoft.com/de-de/magazine/2007.01.cableguy.aspx

[50] TCP's Initial Congestion Window (ICW or IW), 17.11.2010
http://kb.pert.geant.net/PERTKB/TcpInitialWindow
An Argument for Increasing TCP's Initial Congestion Window. 17.11.2010
https://code.google.com/intl/de-CH/speed/articles/tcp_initcwnd_paper.pdf

[51] Slow Start and Congestion Avoidance Algorithms, 17.11.2010
http://msdn.microsoft.com/en-us/library/ms818965.aspx

[52] New Networking Features in Windows Server 2008 and Windows Vista, 17.11.2010
http://technet.microsoft.com/en-us/library/bb726965.aspx

[53] Xiuchao Wu, “A Simulation Study of Compound TCP”, National University of Singapore, July 2008
http://www.comp.nus.edu.sg/~wuxiucha/research/reactive/publication/ctcp_study.pdf

[54] IIS 7 Usage Statistics, 19.11.2010
http://trends.builtwith.com/Web-Server/IIS-7

[55] Port80's 2010 - Top 1000 Corporation Web Servers, 19.11.2010
http://www.port80software.com/surveys/top1000webservers

[56] What is IE's Maximum Parallel Connection Accross All Hosts, 19.11.2010
http://stackoverflow.com/questions/2566612/what-is-ies-maximum-parallel-connection-accross-all-hosts

[57] HTTP pipelining, 19.11.2010
http://en.wikipedia.org/wiki/HTTP_pipelining

[58] Gaming Tweak - Disable Nagle's algorithm. 19.11.2010
http://www.speedguide.net/articles/windows-7-vista-2008-tweaks-2574

[59] CSS3 Selectors Test, 15.12.2010
http://www.css3.info/selectors-test/

[60] Acid (Browsertests), 15.10.2010
http://de.wikipedia.org/wiki/Acid_(Browsertests)#Acid3

[61] WebKit SunSpider, 15.12.2010
http://ie.microsoft.com/testdrive/benchmarks/sunspider/default.html
grafische Auswertung siehe: JavaScript Performance Vergleich.xlsx

[62] WebKit, 15.12.2010
http://en.wikipedia.org/wiki/SunSpider_JavaScript_Benchmark#SunSpider

[63] HTTP Pipelining: A security risk without real performance benefits, 15.12.2010
http://devcentral.f5.com/weblogs/macvittie/archive/2009/04/02/http-pipelining-a-security-risk-without-real-performance-benefits.aspx

[64] HTTP Delay Estimation, 15.12.2010
Pratima Akkunoor, Arizona State University, USA
http://lerci.tagus.ist.utl.pt/applets/http/http.html

[65] Roundup on Parallel Connections, 15.12.2010
http://www.stevesouders.com/blog/2008/03/20/roundup-on-parallel-connections/

[66] Dromaeo JavaScript Tests, 15.12.2010
http://dromaeo.com/?dromaeo

[67] Browser script loading roundup, 16.12.2010
http://www.stevesouders.com/blog/2010/02/07/browser-script-loading-roundup/

[68] Evolution of Script Loading, 16.12.2010
http://www.stevesouders.com/blog/2010/12/06/evolution-of-script-loading/

[69] Loading Scripts Without Blocking, 16.12.2010
http://www.stevesouders.com/blog/2009/04/27/loading-scripts-without-blocking/

[70] ControlJS part 1: async loading, 16.12.2010
http://www.stevesouders.com/blog/2010/12/15/controljs-part-1/

[71] 14 Rules for Faster-Loading Web Sites, 16.12.2010
http://stevesouders.com/hpws/rules.php

[72] Google's Ausfall – die tatsächliche Bedeutung für das Web, 16.12.2010
http://www.kiutalk.de/2009/05/google-s-ausfall-die-tatsaechliche-bedeutung-fuer-das-web/

[73] jQuery, 16.12.2010
http://www.jquery.com/

[74] Selektoren, 15.12.2010
http://jendryschik.de/wsdev/einfuehrung/css/selektoren

[75] FireDownload, 16.12.2010
https://addons.mozilla.org/en-US/firefox/addon/10615/

[76] CSS Sprites: How Yahoo.com and AOL.com Improve Web Performance, 16.12.2010
http://www.websiteoptimization.com/speed/tweak/css-sprites/

[77] Which browsers support HTTP Pipelining?, 19.12.2010
http://www.quora.com/Which-browsers-support-HTTP-Pipelining

[78] V. Condoleo, C. Dirac, R. Ruoss, „Netzwerk-Emulatoren auf dem Prüfstand", Teil NetEm, Bachelorarbeit FS2008, Hochschule Rapperswil, 2008.

[79] Queueing Disciplines for Bandwidth Management, 20.12.2010
http://lartc.org/howto/lartc.qdisc.html

[80] tc-htb(8) - Linux man page, 11.12.2010
http://linux.die.net/man/8/tc-htb

[81] tc-tbf(8) - Linux man page, 18.12.2010
http://linux.die.net/man/8/tc-tbf

[82] [Netem] limit parameter, 18.12.2010
https://lists.linux-foundation.org/pipermail/netem/2007-August/001148.html

[83] Peter Heinzmann, HSR, „2.8 Ethernet Frame", Vorlesung Computernetze 1 vom HS 2009

[84] K.E. Avrachenkov, U. Ayesta, E. Altman, C. Barakat, "The effect of router buffer size on the TCP performance", 20.12.2001
http://homepages.laas.fr/urtzi/argi/StPeter_paper.pdf

[85] Aussage von Prof. Dr. P. Heinzmann am 22.10.2010 an einer wöchentlichen Besprechung

[86] kernel_flow | The Linux Foundation, 18.12.2010
http://www.linuxfoundation.org/collaborate/workgroups/networking/kernel_flow

[87] JPerf v2.0.0, xjperf - Graphical frontend for IPERF written in Java, 19.11.2010
http://code.google.com/p/xjperf/

[88] Digital TV verbreitet sich, Mobile TV nicht, 15.10.2010
http://www.pctipp.ch/news/kommunikation/41307/digital_tv_verbreitet_sich_mobile_tv_nicht.html

[89] Zeitfehler, 01.12.2010
http://de.wikipedia.org/wiki/Zeitfehler

[90] Auswertung von Messergebnissen: Addition von Varianzen, 21.12.2010
http://www.chemieonline.de/forum/showthread.php?t=41529

[91] Top Sites in Switzerland, 01.12.2010
http://www.alexa.com/topsites/countries/CH

[92] Command-Line Syntax Key, 02.12.2010
http://technet.microsoft.com/en-us/library/cc771080%28WS.10%29.aspx

[93] Internet Explorer 8 im Test, 13.12.2010
http://www.tecchannel.de/kommunikation/extra/1841449/test_ie8_firefox_chrome_opera/index9.html

[94] Video LAN Client v1.1.5 Win32, 10.12.2020
http://mirror.switch.ch/ftp/pool/3/mirror/videolan/vlc/1.1.5/win32/vlc-1.1.5-win32.exe

[95] Wegen ein paar Millisekunden: Gefahr durch DNS-Prefetching, 13.12.2010
http://www.zdnet.de/sicherheits_analysen_wegen_ein_paar_millisekunden_gefahr_durch_dns_prefetching_story-39001544-41532432-1.htm

[96] dig(1) - Linux man page, 13.12.2010
http://linux.die.net/man/1/dig

[97] Google Chrome: Schneller surfen dank DNS-Prefetching, 13.12.2010
http://www.golem.de/0809/62530.html

[98] Link prefetching FAQ, 13.12.2010
https://developer.mozilla.org/en/Link_prefetching_FAQ

[99] Link Prefetching mit HTML5, 14.12.2010
http://www.christianranz.com/post/link-prefetching-mit-html5

[100] ACM Digital Library, 24.09.2010
http://portal.acm.org/

[101] IEEE Xplore Digital Library, 24.09.2010
http://ieeexplore.ieee.org/

[102] ISI Web of Knowledge, 24.09.2010
http://isiknowledge.com/

[103] OvidSP - Inspec, 24.09.2010
http://ovidsp.ovid.com/

[104] Emerald, 24.09.2010
http://www.emeraldinsight.com/

[105] Google, 24.09.2010
http://www.google.ch

[106] Yahoo! YSlow, 17.12.2010
http://developer.yahoo.com/yslow/

[107] Google Page Speed, 17.12.2010
http://code.google.com/intl/de-DE/speed/page-speed/

[108] Firebug v1.5.4, 17.12.2010
http://getfirebug.com/

[109] HttpWatch Pro v7.0.26, 17.12.2010
http://www.httpwatch.com/

[110] dynaTrace AJAX Edition v2.0.0.574, 17.12.2010
http://ajax.dynatrace.com/

[111] Microsoft Visual Round Trip Analyzer v3.0.0250.1024, 17.12.2010
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=119f3477-dced-41e3-a0e7-d8b5cae893a3&displaylang=en

[112] Microsoft Network Monitor v3.4, 17.12.2010
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=983b941d-06cb-4658-b7f6-3088333d062f&displaylang=en

[113] Fiddler Web Debugger v2.3.0, 17.12.2010
http://www.fiddler2.com/

[114] Google Speed Tracer v0.18, 17.12.2010
http://code.google.com/intl/de-CH/webtoolkit/speedtracer/

[115] Wireshark v1.4.2, 17.12.2010
http://www.wireshark.org/

9.10.3. Abbildungen

9.10.4. Tabellen

9.10.5. Formeln