Institut für Kommunikationsnetze und Rechnersysteme
Pfaffenwaldring 47
Universität Stuttgart
D-70569 Stuttgart

Diplomarbeit Nr. 3333

Entwurf eines Überlast-Reglers unter Berücksichtigung des Netzzustandes und möglicher Überlast-Beschränkungen

Dirk Mast

Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. Andreas Kirstädter
Betreuer:	Dipl.-Ing. Mirja Kühlewind, Dipl.-Ing. Frank Feller, Dipl.-Ing. Sebastian Scholz
begonnen am:	2. Mai 2012
beendet am:	1. November 2012
CR-Klassifikation:	C.0, C.2, C.2.3, C.4

Inhaltsverzeichnis

1. Einleitung

Das Internet und das typische Verhalten der Nutzer hat sich in den letzten Jahren stark verändert. Während früher zum Großteil Text und Daten übertragen wurden, dominieren heute zeitkritischer Videoverkehr und Peer-to-Peer Filesharing die Zusammensetzung des Internetverkehrs.

Einen Überblick über die heutige Zusammensetzung des Verkehrs verschafft [Cis11], dargestellt in Abbildung 3.2.

Zusätzlich zum Wandel der Verkehrszusammensetzung durch die unterschiedlichen Dienste kommt ein starkes Ansteigen der Datenmenge hinzu, was exemplarisch an der Verkehrsentwicklung des DE-CIX in Abbildung 1.1 zu sehen ist.

Diese neuen Verkehrsmuster haben weitreichende Folgen und unterschiedliche Anforderungen an die Infrastruktur der Provider und deren Nutzer.

Auf dem Pfad eines Nutzers, der eine Verbindung mit einem Dienst über das Internet aufbaut, liegen typischerweise einer oder mehrere Router. Jeder dieser Router hat einen gewissen Pufferspeicher für Pakete (Routerqueue). Sobald der Router Pakete schneller empfängt, als er sie weiterleiten kann, füllt sich dieser Puffer. Bei einem Überlauf des Puffers kommt es zum Paketverlust, da der Router weitere ankommende Pakete verwerfen muss.

Dieses Verwerfen führt entweder zu komplettem Datenverlust, oder einem erneuten Senden der Daten und bringt damit Verzögerungen mit sich, die für Echtzeitanwendungen oder Streaming problematisch sind. Das erneute Senden von Daten, wie es vom Transmission Control Protocol (TCP) bei Verlust gemacht wird, sorgt dafür, dass der Empfänger garantiert Daten erhält. Allerdings füllt sich dadurch wiederum die Routerqueue und verschlimmert daher möglicherweise die Überlastsituation. Ein Ansteigen der Latenzzeit durch größere Pufferfüllstände ist eine weitere unangenehme Folge des erneuten Sendens.

Häufig wird versucht, die unterschiedlichen Anforderungen von einzelnen Verkehrsarten mit einer gezielten Priorisierung an den Zwischenroutern zu lösen. Hierbei wird Echtzeitverkehr mit einer höheren Priorität versendet, wie zeitunkritischer Verkehr, welcher länger im Routerpuffer verweilen muss. Im Gegensatz zu Verkehrspriorisierungen wird in dieser Diplomarbeit versucht, mit einem Adaptionsregler auf die Überlastsituation zu reagieren, um nicht nur in einer Überlastsituation gezielt manchen Verkehr zu bevorteilen, sondern aktiv den erzeugten Verkehr und damit die Überlastsituation zu reduzieren. Dieser Ansatz hat den Vorteil, dass nur ein Feedback über den Netzzustand benötigt wird, und man nicht auf intelligentere Router angewiesen ist, die Pakete priorisieren können.

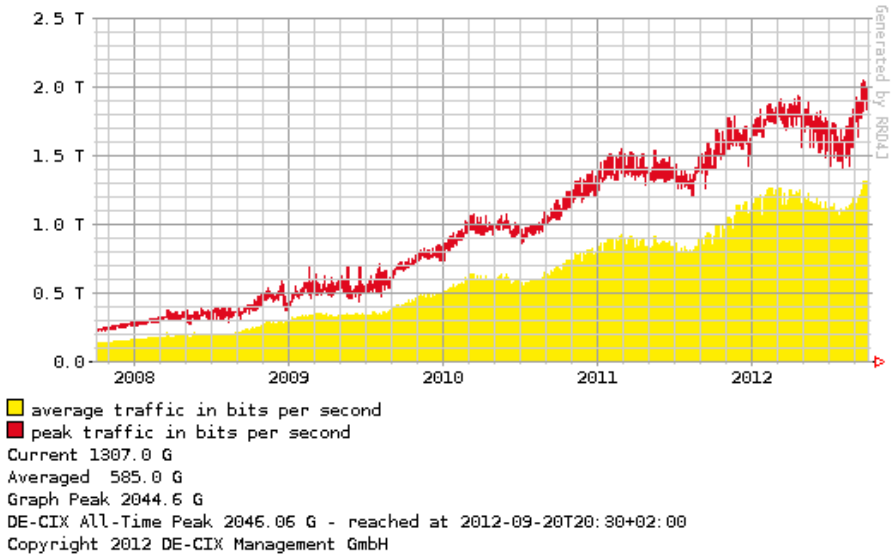


Abbildung 1.1.: Verkehrsentwicklung der letzten 5 Jahre des Internet-Peerers DE-CIX, Quelle [DC]

Dieser vorgeschlagene Adaptionsregler soll die verkehrserzeugenden Dienste adaptieren, so dass sie beispielsweise bei einer Videoübertragung die Bitrate herunterregeln, bis die Überlast wieder vorbei ist, damit es zu keinen weiteren Verlusten aufgrund einer momentan eingeschränkten Bandbreite und vollen Routerpuffern kommt. Hierdurch soll eine bessere und fairere Bandbreitennutzung erzielt werden.

In dieser Arbeit werden die einzelnen Verkehrsarten modelliert, anschließend Adaptionsmechanismen für die Verkehrsarten vorgeschlagen und diese evaluiert.

Gliederung

Die Arbeit ist in die folgenden Abschnitte gegliedert:

Kapitel 2 - Grundlagen: Hier werden für das Verständnis der Arbeit wichtige Konzepte, wie die Überlastkontrolle des *Transmission Control Protocols* eingeführt und Grundlagen des Sendeverhaltens einzelner Verkehrsarten beschrieben. Es werden bisherige Arbeiten vorgestellt, die sich mit der Modellierung dieser Verkehrsarten, sowie teilweise mit deren Adaption befassen.

Kapitel 3 - Modellierung: Der Internetverkehr wird in Verkehrsklassen zusammengefasst. Diese werden repräsentativ modelliert und anschließend werden die erzeugten Verkehrsvolumina der Klassen Bulk, Video, VoIP, Peer-to-Peer, Web an den Cisco Visual Network Index angepasst ([Cis11]), um eine Verkehrsverteilung zu erhalten, wie man sie im Internet erwartet. Daraufhin werden Adaptionsmechanismen für die Verkehrsklassen vorgestellt, die bei einer Überlastsituation angewendet werden können, um diese zu beseitigen.

Kapitel 4 - Bewertung der Adaptionsmechanismen: Dieses Kapitel präsentiert Simulationsergebnisse eines starken Überlastszenarios, bei dem die verschiedenen in Kapitel 3 vorgestellten Adaptionsmechanismen angewendet werden. Nach Beschreibung des Simulationsszenarios werden Kriterien und Metriken vorgestellt, die bei der Evaluation der Ergebnisse helfen. Anschließend werden die Ergebnisse präsentiert und diskutiert.

Kapitel 5 - Zusammenfassung und Ausblick: Die Ergebnisse dieser Arbeit werden noch einmal zusammengefasst und offene Punkte zur Weiterführung des Themas werden angesprochen.

2. Grundlagen

„Die Traffic-Spitze von über 2 Tbit/s markiert zwar einen neuen Höchstwert, aber wir sehen für das Wachstum des Datenverkehrs kein Ende am Horizont.“

(Geschaftsführer DE-CIX)

2.1. TCP Congestion Control

2.1.1. Best Effort

Das Transmission Control Protocol (TCP) ist ein verbindungsorientiertes Protokoll zum zuverlässigen Übertragen von Daten in Netzwerken. TCP beinhaltet eine Überlastkontrolle, die verhindern soll, dass es aufgrund einer stark ausgelasteten Leitung zu Datenverlust kommt. Die Überlastkontrolle von TCP (Congestion Control) regelt das Sendeverhalten und damit effektiv die Bandbreite einzelner Verbindungen (diese werden auch als Flows bezeichnet), da TCP bei Überlast “zurück weicht”. Dieses Verhalten wird durch mehrere Mechanismen wie *Slow Start, Congestion Avoidance, Fast Retransmit* sowie *Fast Recovery* erreicht und wurde in [APS99] spezifiziert und in [APB09] überarbeitet. An dieser Stelle wird eine Grundkenntnis über diese Mechanismen als bekannt vorausgesetzt.

Teilen sich mehrere standardkonforme TCP Flows eine gemeinsame Leitung, so teilen diese laut [Floo0] die Bandbreite fair untereinander auf. Eine ausführliche Analyse für dieses Verhalten findet sich in [CJ89]. Die Autoren zeigen, dass es der *Additive Increase, Multiplicative Decrease* [1] schafft, die Bandbreite zu gleichen Teilen zwischen verschiedenen Flows aufzuteilen.

Es wurde bereits im Jahr 2000, in [Floo0] darauf hingewiesen, das dies nur zu einer Fairness zwischen Flows, nicht jedoch zu einer Fairness zwischen Applikationen führt. Bereits beim Entwurf des Hypertext Transfer Protocol 1.1 wurde erkannt, dass dies zu einem Problem führen kann [FGM+99].

Dort heißt es:

> “Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy.”

[1] Grundlage um das *congestion Window* (limitiert die Senderate) der Überlastkontrolle von TCP zu berechnen

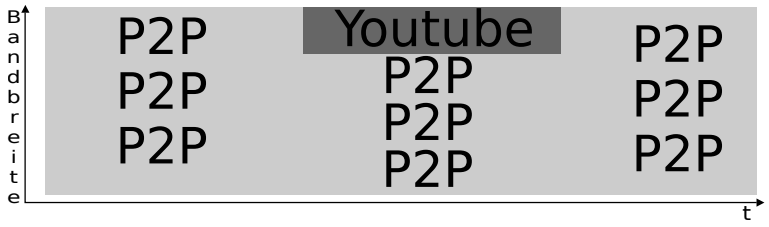


Abbildung 2.1.: Bandbreitenteilung durch TCP

Damals fingen Webbrowser an, das Laden einer Internetseite zu beschleunigen, indem mehrere Verbindungen geöffnet wurden. Heutzutage ist dieses Vorgehen von Webbrowsern aufgrund gestiegener Bandbreite und den verhältnismäßig kleinen Internetseiten nicht mehr so problematisch und daher gängige Praxis um schnellere Ladezeiten zu erreichen. Generell hat sich an der Problemstellung *Flowfairness* oder *Applikationsfairness* nichts geändert.

Mit der Einführung von Peer-to-Peer Applikationen wie Bittorrent und speziell deren immer größeren Verbreitung gewinnt das TCP Fairnessproblem erneut an Bedeutung. Eine einzelne Peer-to-Peer Applikation erzeugt typischerweise eine große Anzahl an Flows, was dazu führt, dass das Verhalten von TCP zwar dafür sorgt, dass sich die Bandbreite aller Flows auf einen gleichen Anteil einpendelt, damit aber unbewusst die Peer-to-Peer Applikation aufgrund ihres größeren Flowanteils stark bevorzugt. Dieses Verhalten ist exemplarisch in Abbildung 2.1 dargestellt.

Less than Best Effort - LEDBAT

Neben zeitkritischen Übertragungen wie Videostreaming gibt es auch Verkehr, der im Hintergrund übertragen werden kann und möglichst den zeitkrischen Verkehr nicht beeinflussen soll. Der Algorithmus *Low Extra Delay Background Transport* wurde entwickelt um dies zu ermöglichen. LEDBAT versucht die volle verfügbare Bandbreite auszuschöpfen, falls keine Überlast vorliegt.

Liegt eine Überlast vor, werden die zu übertragenden Pakete in eine Warteschlange (Queue) auf einem paketvermittelnden Router eingereiht. Durch das Wachsen dieser Warteschlange steigt die Verzögerung, die Pakete brauchen, um übertragen zu werden. Steigt diese Verzögerung, erkennt LEDBAT dies durch Verzögerungsmessungen, die in einer Richtung durchgeführt werden. LEDBAT versucht sich in diesem Fall defensiv zu verhalten und reduziert die Senderate, was einerseits die Verzögerung nicht verschlimmert und andererseits anderen TCP Flows den Vorrang lässt.

Aus diesem Grund wird LEDBAT nachfolgend zur Adaption von Bulk sowie Peer-to-Peer Übertragungen vorgeschlagen.

2.2. Videoverkehr

Youtube - Block Sending Videostreaming hat heutzutage den größten Verkehrsanteil im Internet (siehe [Cis11]) und es wird prognostiziert, dass dieser Anteil weiter wächst. Der wichtigste Videostreamingdienst ist Youtube, weshalb dieser als Modellierungsgrundlage im späteren Verlauf herangezogen wird. Wie sich Youtube beim Übertragen eines Videos verhält ist daher von besonderem Interesse und wurde in [AN11], sowie [GCJM12] behandelt.

Die Autoren von [AN11] prägen den Begriff des *block sending*, da Youtube typischerweise in 64kB Blöcken überträgt, was durch das unterliegende GoogleFS[2] begründet ist. Auf der Clientseite existiert ein Puffer (*Buffer*), der sicherstellt, dass das Video flüssig abgespielt werden kann. Die Idee dabei ist, dass für eine Videoübertragung die Bandbreite mindestens der Videocodecrate entsprechen muss. Um Fluktuationen der Bandbreite im Mittel ausgleichen zu können, sollen beim Empfänger bereits Daten vorgehalten werden, die erst in der Zukunft ausgespielt werden ([CMP08]). Dieser Puffer wird zu Beginn der Übertragung schnellstmöglich gefüllt. Danach wird die Geschwindigkeit stark gedrosselt und es wird nur noch ungefähr mit einer Geschwindigkeit gesendet, die der Bitrate des Videos entspricht. Hierbei wird nicht konstant gesendet, sondern in Blöcken, die häufig einen Abstand größer einer Paketumlaufzeit haben (*Round-Trip-Time, RTT*). Da Nutzer oftmals ein Video nur zum Teil anschauen, wird dadurch verhindert, dass unnötig viele Daten übertragen werden. Allerdings machen die Autoren das *block sending* für große Übertragungsverluste bei DSL Verbindungen verantwortlich, da diese mit einer so großen Häufung von Paketen (*burst*) überlastet sind.

Die Autoren von [PNK08] haben 2008 eine Geschwindigkeitsbegrenzung von 1.25Mbit/s festgestellt, zu diesem Zeitpunkt gab es allerdings noch keine HD Videos bei Youtube.

Das blockweise Sendeverhalten von Youtube wird in [GCJM12] bestätigt. Drei der Autoren arbeiten bei Google und haben Zugriff auf die Server Applikation von Youtube *ustreamer*. Ustreamer nutzt einen Token-Bucket Algorithmus, welcher einen Sendeplan für die Blöcke des Videos erstellt. Wie von [AN11] beobachtet, werden die ersten 30-40s des Videos wie eine TCP Bulk Übertragung gesendet, anschließend beginnt das Verzögern der zu übertragenden Blöcke. Tritt aus irgendeinem Grund ein Pufferunterlauf beim Client ein, wird erneut wie bei einer TCP Bulk Übertragung gesendet. Es wird immer versucht, einen Überschuss von 25% Videodaten pro Videosekunde zu senden. Die Autoren fahren fort und bestätigen das Problem, das durch das burstartige Sendeverhalten auftritt, da es periodische Queuespitzen auf dem Pfad erzeugt, die auch andere zeitkritische Anwendungen negativ beeinflussen. Auf eine kleinere Blockgröße zu wechseln halten die Autoren nicht für die richtige Lösung, da dies für das verwendete Dateisystem einen großen Input/Output-Anstieg bedeutet.

Als Lösung wird vorgeschlagen, ein Geschwindigkeitslimit über TCP zu erreichen, wozu das *congestion window* mit *Ausspielrate* * *RTT* begrenzt wird. In den Ergebnissen zeigt sich, dass dies zwar für zwei getestete Datacenter (Westeuropa und Indien) die Verluste durch die Burstübertragung stark reduziert, sowie die *durchschnittliche RTT* reduziert, aber keinen

[2] Google File System

Einfluss auf die Pufferunterläufe beim Videoausspielen hat, die zum Anhalten der Videos führen.

Akamai HD - Adaptive Video Streaming Der Dienst *Akamai HD* des Content Delivery Network (CDN) Anbieters Akamai verfolgt den Ansatz des *Stream Switching*. Jedes auszuliefernde Video liegt in fünf Qualitätsstufen ([CM10a]) vor. Abhängig von der momentan tatsächlich verfügbaren Bandbreite zwischen Client und Server wird automatisch zwischen diesen Qualitätsstufen umgeschaltet. Damit soll, abhängig von der aktuellen Netzauslastung, erreicht werden, einen Pufferunterlauf beim Client zu vermeiden - oder bei geringer Netzlast eine bessere Qualität zu zeigen.

Zwischen dem Client und dem ausliefernden Server gibt es eine Kommunikation über HTTP, welche dazu verwendet wird, Feedback über die aktuelle Bandbreite auszutauschen. Hierzu wird eine *Round-Trip-Time (RTT)* Messung vorgenommen. Aufgrund dieser Messung wird die Bandbreite geschätzt und dem Server mitgeteilt, der die entsprechende Qualitätsstufe des Videos aussendet. Messungen zeigten, dass es ungefähr *150s* dauert, bis zur maximalen Qualitätsstufe umgeschaltet wird. Bei einem abrupten Bandbreiteneinbruch ist mit ungefähr *14s* Wartezeit bis zur niedrigsten Qualitätsstufe zu rechnen ([CM10a]). Die *14s* sind besonders kritisch, da dies zu einem Pufferunterlauf beim Client führen kann, wodurch das Video anhält.

Neben der Möglichkeit, die aktuell verfügbare Bandbreite über eine RTT Messung auf Applikationsebene abzuschätzen, liefert unter Unix Systemen die *get-sockopt API* das aktuelle *congestion window* cwnd, die *Maximum Segment Size* MSS, sowie einen Schätzwert für die RTT. Hiermit kann der Durchsatz abgeschätzt[3] werden ([MSMO97], [KKH10]).

Eine Optimierung des Verhaltens des *Quality Adaption Controller* von Akamai wird in [CMP11] versucht. Die Autoren schaffen es hier aufgrund besserer Bandbreitenschätzung das Video schneller zur maximalen Qualitätsstufe umzuschalten (nun 30s). Leider gelingt es auch hier nicht, spontan auf Netzüberlast zu reagieren, ohne dass es zu einem Pufferunterlauf kommt.

Neben Akamai wird die Technik *Stream Switching* auch von Apple, Microsoft, Adobe, sowie Skype verwendet.

Stream Switching lässt sich auf verschiedene Arten erreichen. Einerseits der oben vorgestellte Ansatz von Akamai, bei welchem das Video bereits in allen Qualitätsstufen vorcodiert beim Anbieter liegt. Dadurch wird zwar mehr Festplattenplatz beim CDN benötigt, dafür kann man die Daten ohne viele CPU Operationen ausliefern. Problematisch sind Liveübertragungen, da dabei typischerweise die Zeit fehlt, das Video im Vorfeld in die verschiedenen Qualitätsstufen zu codieren. Eine andere Möglichkeit erfordert es nur, die maximale Qualität auf dem Server zu speichern. Während das Video ausgeliefert wird, kann dieses aufgrund der Struktur des Videocodecs leicht neu codiert werden, um eine schlechtere Videoqualität zu erzeugen. Dies benötigt weniger Speicherplatz, dafür allerdings Server mit mehr Rechenleistung.

[3] mit $\frac{cwnd * MSS}{RTT}$

Der Videocodec *H.264/SVC*[4] verfolgt einen ähnlichen Ansatz. Das Video besteht aus Fragmenten mit unterschiedlich vielen Informationen. Die Idee ist, die Videodaten bei der Übertragung so umzuordnen (*priority streaming*), dass der Empfänger die Hauptinformationen (I-Frames) von verschiedenen Videozeitpunkten erhält. Danach werden die verfeinerten Bildinformationen (P und schliesslich B Frames) gesendet. Der Empfänger kann, falls er diese erhält, das Video in bestmöglicher Qualität ausspielen. Erreichen ihn diese unwichtigeren Fragmente nicht - oder zu spät, kann das Video aufgrund der zuvor erreichten wichtigeren Fragmente (I-Frames) dennoch ausgespielt werden, wenn auch in schlechterer Qualität. Dies wird detailliert in [KKH10] untersucht. Mit *H.264/SVC* ist es auf diese Weise möglich, 64 diskrete Adaptionsschritte durchzuführen, wodurch sehr feine Qualitätsunterschiede möglich sind, was dem Nutzer einen flüssigeren Übergang geben kann, als beim "harten Umschalten" zwischen den fünf Qualitätsstufen die bei Akamai HD zum Einsatz kommen.

Es soll erwähnt sein, dass man in *Sandvine*[5] der Meinung ist, dass ein sich von der Qualität her veränderndes Videosignal zu weniger Nutzerzufriedenheit führt, wie ein durchgehend schlechtes.

2.3. VoIP-Verkehr

Telefonie erfährt einen Wandel von der typischen leitungsvermittelnden zur paketvermittelnden Übertragung via IP. Ein sehr populärer Dienst hierfür ist *Skype*. Die technische Grundlage für Voice-over-IP ist das Digitalisieren von Sprache, was mit einem *Pulse-Code-Modulation* Verfahren erfolgt. Im späteren Verlauf wird der Verkehr auf Basis des *G.711 Codec* [cit88] modelliert. *Skype* verwendet zur eigentlichen Gesprächsübertragung kein TCP sondern UDP (User Datagram Protocol). Dieses Protokoll hat im Unterschied zu TCP keine Überlastkontrolle sowie keine erneuten Übertragungen bei Paketverlust. Das erneute Senden von Paketen macht bei einem Telefongespräch aufgrund der zeitkrischen Ankunft der Pakete nur begrenzt Sinn. In dieser Arbeit wird dennoch für VoIP TCP verwendet, auch wenn dies nicht optimal ist. Ein Versuch, das Sendemodell von Skype mathematisch nachzubilden findet sich in [CM10b].

2.4. Peer-to-Peer Verkehr

In den letzten Jahren haben verschiedene Peer-to-Peer Systeme mit unterschiedlicher Funktionsweise Bedeutung erlangt. Eine Gemeinsamkeit aller Systeme ist die Verbindung zwischen einzelnen Internetteilnehmern, bei denen jeder Teilnehmer sowohl die traditionelle Client (*Leecher*), wie auch die Serverrolle (*Seeder*) annimmt. Pro Applikation sind das oft mehr als 100 Verbindungen ([BMM+08]). Dies ist eine große Herausforderung für die Infrastruktur des Internets, welches von den Providern zum Endnutzer oft asymmetrisch ausgelegt ist (große Downstreambandbreite, geringe Upstreambandbreite).

[4] H.264 - Scalable Video Codec

[5] `http://www.sandvine.com/news/global_broadband_trends.asp`

Zum Zeitpunkt dieser Diplomarbeit ist das Peer-to-Peer System mit der größten Verbreitung BitTorrent (vgl. [Cis11], sowie Sandvine[6]).

BitTorrent ist ein Peer-to-Peer System, welches vor allem für den Austausch großer Dateien verwendet wird, was die Last auf viele Nutzer verteilt, anstatt auf einzelne Serveranbieter. Im Unterschied zu bisherigen Systemen fehlt BitTorrent eine Suchfunktion für Dateien. Suchinformationen müssen beispielsweise über das Internet publiziert werden. Hierfür gibt es Dienste, die eine Liste von *torrents* führen. Dies sind kleine Dateien, die beschreiben wo einzelne Stücke der zu verteilenden Datei zu finden sind. Zudem werden hier noch Prüfsummen für diese Stücke bereitgestellt.

Die sehr hohe Verbindungsanzahl der Nutzer ist problematisch für die *TCP Congestion Control*, was auch von den BitTorrententwicklern adressiert wird. Als Lösungsversuch wird ein sogenanntes *choke* Verfahren im Protokoll spezifiziert[7], das eine Untermenge aus den verfügbaren Verbindungen auswählt, über die dann tatsächlich übertragen wird.

Da die Übertragungen typischerweise nicht zeitkritisch sind, wurde das Übertragungsprotokoll *µTP* [8] eingeführt, das auf TCP LEDBAT (vergleiche 2.1.1) basiert. Eine Evaluation des Protokolls findet sich in [RTV10].

Von *David Erman* wurde über mehrere Jahre hinweg in verschiedensten Veröffentlichungen versucht, den Verkehr von BitTorrent zu verstehen und modellieren. Das später eingeführte Modell stützt sich stark auf seine Ergebnisse in [ESSGP08].

2.5. Webverkehr

Typischer Webverkehr über das *HTTP-Protokoll* ist der Standardverkehr des traditionellen Internets. Für Grundlagen wird auf die Spezifikation [FGM+99] verwiesen. Für diese Diplomarbeit ist weniger das Übertragungsverfahren interessant, als der Verkehr, der dabei entsteht. Wurde früher hauptsächlich Text in HTML Form übertragen, finden sich heute häufig eingebettete Objekte auf Internetseiten.

> "In this specification, the term "object" is used to describe the things that people want to place in HTML documents; other commonly used terms for these things are: applets, plug-ins, media handlers, etc." [9]

Das *object*-Tag in HTML erlaubt das generische Einbetten von Inhalten aller Art.

Gerade durch diese eingebetteten Objekte ist der Verkehr interessant: neben kurzen Übertragungen (weniger als 10kB, auch als *Mice* von [BMM+08] bezeichnet) finden auch größere Übertragungen statt (mehr als 5Mb, sogenannte *Elephants*).

[6] http://www.sandvine.com/news/global_broadband_trends.asp
[7] http://www.bittorrent.org/beps/bep_0003.html
[8] http://www.utorrent.com/help/documentation/utp
[9] http://www.w3.org/TR/REC-html40/struct/objects.html

In [BMM+08] wird festgestellt, dass die Anzahl der *TCP Flows* deutlich höher als die Anzahl der unterschiedlichen Webhosts ist. Genauer heißt dies, dass Webbrowser mehr als eine Verbindung zu einer Internetseite aufbauen um die Geschwindigkeit zu optimieren. Dies steht im Widerspruch zur in 2.1.1 zitierten Empfehlung aus [FGM+99]. Hierbei wird ausgenutzt, dass die *TCP Congestion Control* zwischen einzelnen *Flows* die Bandbreite gleichmässig aufteilt, nicht jedoch zwischen Applikationen.

3. Modellierung

3.1. Verkehrsklassen

Zur Bewertung der Adaptionsmechanismen ist ein sinnvolles Simulationsmodell zu verwenden, das die Realität möglichst genau abbildet. Hierzu wird Verkehr, wie er heute typischerweise auftritt, in Klassen zusammengefasst. Diese Verkehrsklassen werden modelliert TODO WOMIT. Anschließend werden, die durch die Modelle erzeugten Verkehrsvolumen an denen des Internet zu orientiert. Ein Bild über die Verkehrszusammensetzung im Internet verschaffen Studien von Cisco [Cis11], Akamai[1], sowie Sandvine[2].

Diese Arbeit gliedert die Verkehrsklassen ähnlich wie [Cis11] in *Bulk, Video, VoIP, Peer-to-Peer* sowie *Webverkehr*. Die anderen Studien haben zum größten Teil ähnliche Untergliederungen, die Datenvolumen weichen zum Teil etwas ab.

- *Bulkverkehr* modelliert größere Dateiübertragungen, wie FTP, oder HTTP Downloads die zeitunkritisch sind.
- *Videoverkehr* modelliert Videoübertragungen nach dem Vorbild Youtube, da dies heute der größte Videostreamanbieter ist.
- *Voice Over IP Verkehr* modelliert Telefongespräche zwischen zwei Nutzern basierend auf dem Codec G.711.
- *Peer-to-Peer Verkehr* modelliert das Verhalten eines BitTorrent Nutzers, der eine größere Videodatei an mehrere Nutzer verteilt.
- *Webverkehr* modelliert das Surfverhalten eines Nutzers, der in gewissen Abständen Internetseiten mit eingebetteten Objekten besucht.

In der Studie von [Cis11] beinhaltet die Klasse Webverkehr die Klasse Bulkverkehr. Diese werden aber grundsätzlich unterschiedlich modelliert und werden daher hier getrennt modelliert. Außerdem existiert noch eine zusätzliche Klasse *Online Gaming*, welche allerdings nur einen Anteil von weit unter einem Prozent am Verkehrsvolumen ausmacht und hier daher keine Beachtung findet.

Selbstverständlich finden sich für die ausgewählten und modellierten Verkehrsklassen in der Literatur verschiedene Ansätze zur Modellierung, sowie unterschiedliche verwendete

[1] `http://www.akamai.com/stateoftheinternet/`
[2] `http://www.sandvine.com/news/global_broadband_trends.asp`

Verteilungsfunktionen. Oftmals liegt dies daran, dass die Autoren einen bestimmten Verkehrsverlauf aufgezeichnet haben und anschließend anhand dieser mitgeschnittenen Daten geprüft haben, welche Verteilungsfunktionen zutreffen. Im Folgenden wurde daher beispielsweise ein BitTorrent Modell von [ESSGP08] gegenüber dem von [BMM+08] bevorzugt, da es über viele Jahre entworfen, ausgewertet und angepasst wurde, während das abgelehnte Modell nur über einen kurzen Zeitraum entstand.

Die in den folgenden Abschnitten verwendeten Definitionen von Verteilungsfunktionen stammen aus dem *IKR Simulation Library 2.7 Reference Guide* [3].

3.1.1. Bulkverkehr

Die Verkehrsklasse Bulkverkehr versucht große Dateiübertragungen zu modellieren, wie beispielsweise das Herunterladen eines CD-Images oder anderer Nutzdaten. Der Verkehr zeichnet sich durch ein hohes Volumen aus, wobei generell von Daten ausgegangen wird, bei denen der Erhalt im Unterschied zum Echtzeitverkehr nicht zeitkritisch ist. Modelliert werden muss eine Dateigrösse, sowie eine Inter-Arrival-Time für die einzelnen Übertragungen. Die Verteilungen der Klasse stammen nicht aus Drittliteratur.

Dateigröße Für die Modellierung der zu versendenden Dateigröße wird eine abgeschnittene Paretoverteilung verwendet.

Eine Paretoverteilung ist gegeben durch

$$P(T \leq t) = F(t) = 1 - (\frac{k}{t})^{\alpha} \quad für \quad t \geq k \quad mit \quad \alpha, \; sowie \; k > 0. \tag{3.1}$$

Hierbei bezeichnet k das Minimum und α einen Wert um die Gestalt der Funktion anzupassen. Ein großes α bewirkt, dass große Funktionswerte seltener vorkommen.

Verwendet wird $\alpha = 3$ und ein Minimum $k = 314572800$ (300Mb). Die Verteilung wird bei 400Mb abgeschnitten, damit das Modell ungefähr ähnlich große Dateiübertragungen erzeugt und keine die Statistik verzerrenden Ausreisser.

Bulk - Inter Arrival Time Die Zwischenankunftszeit der einzelnen Dateiübertragungen (IAT) wird mit einer Poissonverteilung modelliert. Die Poissonverteilung ist gegeben durch die Verteilungsfunktion

$$P(X = i) = \frac{(\lambda t)^i}{i!} * e^{(-\lambda t)}. \tag{3.2}$$

Im Modell wird diese mit dem Mittelwert $\lambda * t = 100$ in Sekunden initialisiert.

[3] `http://www.ikr.uni-stuttgart.de/INDSimLib/Resources/Documentation/ReferenceGuide-2.7.pdf`

3.1.2. Videoverkehr

Dieses Modell versucht das Sendeverhalten von Youtube nachzubilden. Hierfür wird das in 2.2 vorgestellte Verfahren *block sending* implementiert. Dazu werden 64kB Blöcke mit Videoinformationen so lange, so schnell wie möglich gesendet, bis auf der Clientseite ein Puffer gefüllt ist. Ab diesem Zeitpunkt werden die folgenden Blöcke verzögert, um den Puffer stabil gefüllt zu halten. Dazu muss ein Feedback vom Client verfügbar sein. Dies kann direkt der Pufferfüllstand sein, oder die insgesamt empfangenen Daten in Verbindung mit dem Zeitpunkt des ersten Ausspielens.

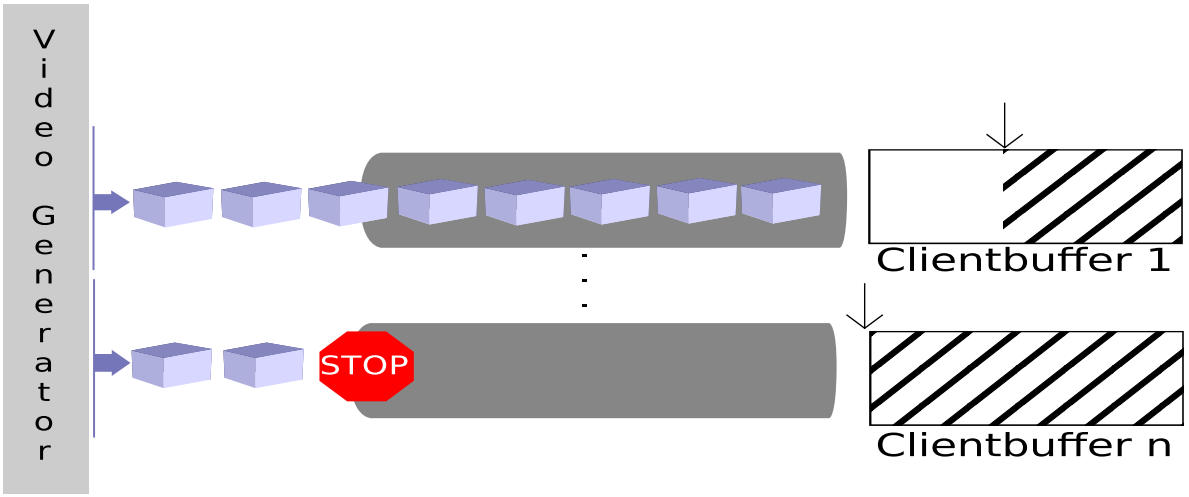


Abbildung 3.1.: Sendeverhalten des Videomodell

In Abbildung 3.1 ist zu sehen, wie bei *Client 1* der Puffer noch weiter gefüllt werden kann, bei *Client n* aber aufgrund des vollen Puffers das weitere Senden verzögert wird.

Für die Modellierung des Videoverkehrs werden zwei Verteilungen benötigt. Zum Einen muss die Videogröße modelliert werden und zum Anderen wird eine Inter-Arrival Time zwischen verschiedenen Videos benötigt.

Videogröße Für die Größenverteilung wird eine Paretoverteilung verwendet, die sich aufgrund ihrer "heavy-tailed" Eigenschaft anbietet. Die Idee hierzu stammt aus [MKHS10].

Es wird $\alpha = 2.76$ ([MKHS10]), sowie $k = 70000000$ (Beobachtungswert bei Videos) verwendet.

Video Inter-Arrival Time Die Video Inter-Arrival-Time lässt sich mit einer Poissonverteilung modellieren. In [MKHS10] wurde hierzu Videoverkehr beobachtet und nach dem Entfernen von Ausreissern zeigte sich, dass eine Poissonverteilung gut zur Modellierung des Ankunftsprozesses geeignet ist.

Ein Parameter für die Poissonverteilung wird in [MKHS10] nicht direkt angegeben, mit diesem lässt sich das gewünschte Verkehrsvolumen pro Zeit beeinflussen. Im Modell dieser Diplomarbeit wird ein Durchschnittswert $m = \lambda * t = 85$ in Sekunden verwendet, da dieser das gewünschtes Verkehrsvolumen erzeugt.

3.1.3. VoIP Verkehr

Das Modell erzeugt einen Telefongespräch auf Basis eines G.711 PCM [cit88] Stroms mit 64kbit/s. Hierzu werden alle 20ms Pakete mit 160b Payload versendet, wie in [Cis06] beschrieben. Es werden On-Off Verteilungen modelliert, die das aktive Sprechen, bzw. das Zuhören simulieren. Die Verteilungen für die beiden Gesprächsphasen, sowie deren gewählten Parameter stammen aus [Bie06]. In der Implementierung ist es möglich, mehrere Gespräche gleichzeitig zu instanziieren, um ein bestimmtes Verkehrsvolumen zu erzeugen. Ein Gespräch wird hierbei so implementiert, dass es die komplette Simulationszeit geführt wird. Dadurch ist der Overhead der durch TCP verursacht wird, nicht ganz so schlimm. Dennoch ist die Verwendung von UDP vorzuziehen.

Eine Hyperexponentialverteilung ist durch die Verteilungsfunktion

$$P(T \leq t) = F(t) = 1 - \sum_{i=1}^{k} p_i * e^{(-\lambda_i t)} \tag{3.3}$$

gegeben.

Parameter sind die Ordnung $k > 0$, Raten λ_i, sowie die Verzweigungswahrscheinlichkeiten p_i.

On-Phase Für die On-Periode wird eine Hyperexponentialverteilung der Ordnung 2 verwendet. Die Parameter sind: $\lambda_1 = 0.85$, $\lambda_2 = 0.4$, $p_1 = 0.63$, $p_2 = 1 - 0.63$.

Off-Phase Die Off-Periode wird mit einer Hyperexponentialverteilung der Ordnung 3 modelliert. $\lambda_1 = 1.83$, $\lambda_2 = 0.25$, $\lambda_3 = 19.68$, $p_1 = 0.5$, $p_2 = 0.2$, $p_3 = 0.3$.

3.1.4. Peer-to-Peer Verkehr

Als Modellierungsgrundlage wird das Peer-to-Peer Programm BitTorrent verwendet. Es wird das Sendeverhalten eines *Seeders* modelliert, der eine Videodatei an verschiedene *Peers* versendet. Während der Laufzeit kommen *Peers* hinzu, oder beenden das Herunterladen. Dieser Zeitraum wird als *Session* bezeichnet. Ein *Peer* lädt nicht den ganzen Film kontinuierlich herunter, sondern es findet eine Unterteilung in *Chunks* (oder *Pieces*) statt, welche zur tatsächlichen Übertragung erneut in *Subpieces* unterteilt werden. Im Modell wird eine Größe von 256kB pro *Chunk* ([ESSGP08]) verwendet, sowie 32768b als Größe pro *Subpiece* ([EIP10]).

Für die Modellierung des Sendeverhaltens werden drei Verteilungen verwendet. Es wird die *Sessiondauer* in Sekunden, die *Sessiongröße* in Bytes und die *Session Inter-Arrival-Time* in Sekunden erzeugt.

Aus dem Verhältnis der *Sessiondauer* und der *Sessiongröße* lassen sich aufgrund der bekannten *Chunkgröße* äquidistante Sendeabstände für die einzelnen Teilübertragungen der Session bestimmen.

Eine Lognormal-Verteilung sei beschrieben durch die Dichtefunktion

$$P(T = t) = f(t) = \frac{1}{\sqrt{2\pi}\sigma t} * exp(-\frac{(ln(t) - \mu)^2}{2\sigma^2}) \quad für \quad t > 0. \tag{3.4}$$

Sessiongröße Für die *Sessiongröße* wird eine Lognormal-Verteilung mit den Parametern $\mu = 13.89$ und $\sigma = 2.43$ verwendet.

Session Inter-Arrival-Time Für die *Session Inter-Arrival-Time* nutzt man eine Hyperexponentialverteilung der Ordnung 2 mit Parametern $\lambda_1 = 0.1792$, $\lambda_2 = 2.0132$, $p = 0.96666$.

Sessiondauer Die *Sessiondauer* wird mit einer Paretoverteilung mit Parametern $\alpha = 0.47$ und $k = 1149$ modelliert.

Diese Verteilungen, sowie ihre Paremetrisierungen wurden in [ESSGP08] nach Verkehrsmessungen erstellt.

3.1.5. Webverkehr

Webverkehr versucht das Surfverhalten eines (oder mehrerer) Nutzers nachzubilden. Dies zeichnet sich dadurch aus, dass ein Nutzer eine Internetseite aufruft, welche eine variable Anzahl eingebetteter Objekte beinhaltet. Nach einer bestimmten Zeit ruft der Nutzer eine andere Seite auf, oder folgt einem Link auf der aktuellen Seite, wodurch sich der Prozess wiederholt. Die Kenngrößen für das Modell sind also die *Größe der Internetseite*, die *Anzahl eingebetterer Objekte*, deren *Größe*, sowie eine *Inter Arrival Time* zwischen den einzelnen Webseiten. Dies entspricht im Prinzip der Lesedauer des Nutzers.

Das Modell basiert auf einer Studie von Intel [JJL07], die den Web-Verkehr eines Nutzers untersucht.

Eine Gammaverteilung sei gegeben durch die Dichtefunktion

$$P(T = t) = f(t) = \frac{\beta^{-\alpha} * t^{\alpha-1} * exp(-t/\beta)}{\Gamma(\alpha)} \quad für \quad t, \alpha, \beta > 0. \tag{3.5}$$

$\Gamma(x)$ bezeichnet hierbei die Gammafunktion.

Größe der Internetseite Die *Größe der Internetseite* wird durch eine abgeschnittene Lognormal-Verteilung mit Parametern $\mu = 7.90272$, $\sigma = 1.7643$, sowie einer oberen Grenze von 2Mb modelliert.

Anzahl eingebetterer Objekte Die *Anzahl eingebetterer Objekte* wird durch eine Gammaverteilung mit $\alpha = 0.141385$, $\beta = 40.3257$ bestimmt.

Größe der eingebetteten Objekte Die *Größe der eingebetteten Objekte* wird durch eine abgeschnittene Lognormalverteilung mit Parametern $\mu = 7.51384$ sowie $\sigma = 2.17454$ und einer oberen Grenze von 6Mb modelliert.

IAT Die *Inter Arrival Time* zwischen den Webseiten wird durch eine Lognormal-Verteilung mit Parametern $\mu = 0.281044$ und $\sigma = 2.64962$ erzeugt.

3.1.6. Anpassung an Verkehrsvolumen des Cisco Visual Network Index

Die Verkehrsverteilungen generieren ein gewisses Verkehrsvolumen, welches typischerweise dem entspricht, was die Autoren der jeweiligen Veröffentlichungen in ihren eigenen Verkehrsmitschnitten gemessen haben. Da dies nicht zwangsläufig die selben Volumina sind, wie sie im Cisco Visual Network Index [Cis11] vorkommen, werden die Verteilungen für diese Arbeit angepasst.

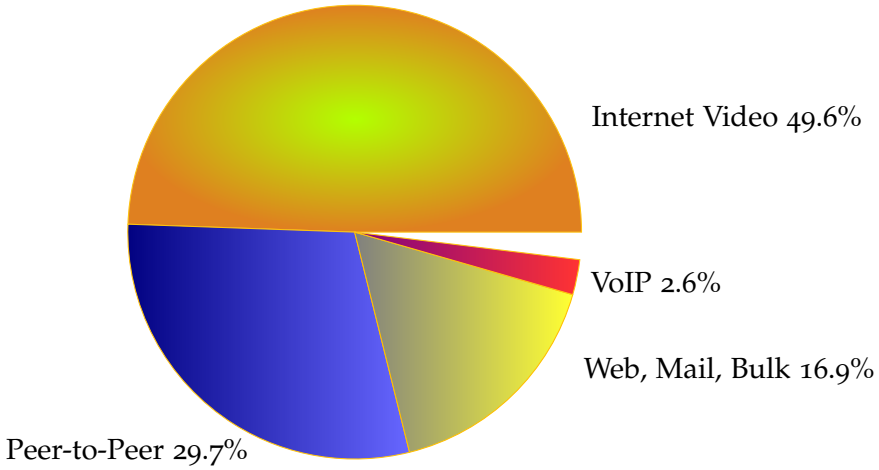


Abbildung 3.2.: Daten: Vorhersage 2012 aus Cisco Visual Networking Index: Forecast and Methodology, 2011-2016

Normalerweise geschieht dies durch eine Anpassung der jeweils generierten Inter Arrival Time (*Bulk*, *Peer-to-Peer*, *Video*) oder durch eine Anpassung der Nutzerzahl (*VoIP*, *Web*), sowie einer maximalen Sessionzahl bei *Peer-to-Peer*.

Im Detail sind die Anpassungen

- Web Verkürzung der IAT um den Faktor 72
- VoIP Instanziierung von 70 Verbindungen
- P2P Limitierung auf 20 Verbindungen, Verdoppelung der IAT
- Verkürzung der Video IAT um den Faktor 16

Diese Anpassungen wurden schrittweise durchgeführt, bis eine zufriedenstellende Entsprechung statt fand.

3.2. Adaptionsmechanismen der Verkehrsklassen

Um die Netzlast zu reduzieren, wird im Folgenden auf verschiedene Möglichkeiten pro Verkehrsklasse eingangen, die alle das selbe Ziel haben: Im Falle einer Überlast weniger Verkehr senden, dadurch eine Überlastsituation aufzulösen. Dennoch sollem dem Nutzer Daten geliefert werden.

3.2.1. Adaption des Bulkverkehrs

Da die Verkehrsklasse des Bulkverkehrs zeitunkritisch ist, bietet sich die Verwendung von LEDBAT an, siehe 2.1.1. Dadurch wird erreicht, dass sich der Bulkverkehr sehr defensiv verhält und zeitkritischen Verkehrsklassen den Vorzug lässt, falls solche eine Übertragung durchführen. In der Praxis kann dies beispielsweise ein Software-Update betreffen, welches dann im Hintergrund heruntergeladen und installiert wird, ohne den Nutzer bei seinem Surfverhalten zu stören.

3.2.2. Adaption des Videoverkehrs

Der Videoverkehr ist genau wie der VoIP Verkehr eine zeitkritische Anwendung. Kommt es beim Client zu einem Pufferunterlauf stockt das Video, was die *Quality of Experience* des Nutzers sehr negativ beeinflusst. Um ein komplettes Aussetzen zu vermeiden, wird daher *Stream Switching* verwendet, wodurch dem Nutzer eine niedrigere (oder höhere) Bitrate abhängig von der Last und damit der Verzögerung ausgespielt wird. Hierbei wird das Konzept des Streaminganbieters Akamai HD (Kapitel 2.2) aufgegriffen und damit das Youtube basierende Modell erweitert.

Es kann eine variable Anzahl von Videoleveln verwendet werden - in dieser Arbeit wurde mit fünf Leveln gearbeitet, wie sie auch bei Akamai HD zum Einsatz kommen.
Ohne Adaption entspricht ein *Block* der Größe 64kB Videoinformationen von 0.5s. Nun wird abhängig von der Auslastung dieses Verhältnis um jeweils 1s angepasst, im Intervall [0.5; 4.5]. Dabei ergeben sich Bitraten von $1024 kbit/s$ bis $113 kbit/s$.

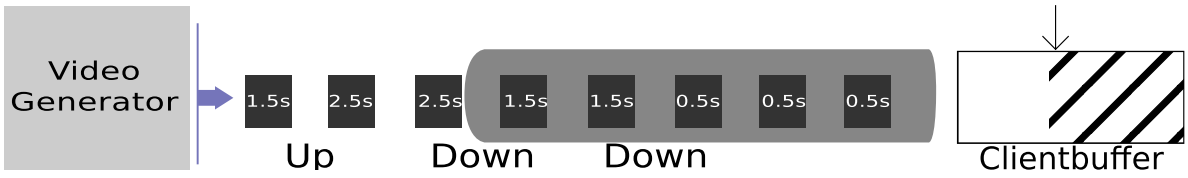


Abbildung 3.3.: Adaptiertes Sendeverhalten des Videomodells

Abbildung 3.3 zeigt, wie aufgrund einer vorhandenen Überlastsituation die pro 64kB Block enthaltenen Videoinformationen erst von 0.5s auf 1.5s und schliesslich 2.5s ansteigen und dann aufgrund einer Aufhebung der Überlast wieder auf 1.5s sinken.

3.2.3. Adaption des VoIP Verkehrs

Genau wie beim Videoverkehr kann hier ein *Stream Switching* verwendet werden, also das "Umschalten" des verwendeten Codecs, abhängig der aktuellen Netzlast. Dies wird auch beim VoIP Telefoniedienst Skype eingesetzt. (Siehe [CM10b]). Bei Skype kommt ausserdem noch eine *Forward Error Control* zum Einsatz, welche den zu übertragenen Paketen Redundanzinformationen hinzufügt. Die Paketgröße steigt hierbei auf das Doppelte, dafür können Fehler erkannt und korrigiert werden. Dies ist vor allem bei Verbindungen interessant, die einzelne Bitfehler induzieren, weniger jedoch für den in dieser Arbeit untersuchten Fall, bei dem ganze Pakete aufgrund von Überlast verworfen werden.

Im hier verwendeten Verkehrsmodell existieren zwei *Level*: einerseits eine Codecrate von 160b/20ms sowie bei Überlast 80b/20ms.

3.2.4. Adaption des Peer-to-Peer Verkehrs

Eine Adaption des Peer-to-Peer Verkehrs erscheint besonders sinnvoll, aufgrund des enormen Anteils am gesamten Verkehrsvolumen. Zum Einen bietet sich LEDBAT (Kapitel 2.1.1) an (was auch in der Realität verwendet wird, Kapitel 2.4), da der Peer-to-Peer Dateiaustausch typischerweise nicht zeitkritisch ist. Eine Ausnahme bildet das zeitkritische Videostreaming über Peer-to-Peer (z.B. *PPStream*[4]), da dies aber in der Verkehrsklasse nicht modelliert wird, spielt es keine Rolle. Zum Anderen kann eine dynamische Anpassung der erlaubten Sessionzahl an die Verkehrslast vorgenommen werden. Das plötzliche Trennen einer Peer-to-Peer Session stellt im Allgemeinen kein Problem für die Anwendung dar, da solche Abbrüche ständig durch das Ausschalten der Anwendung vorkommen. Generell kann man solche zu trennenden Sessions nach speziellen Merkmalen auswählen (z.B. Sessiondauer, übertragenes Volumen) oder zufällig, wie es in dieser Diplomarbeit gemacht wird.

[4] `http://www.pps.tv/`

3.2.5. Adaption des Webverkehrs

Eine Adaption des Webverkehrs könnte durch eine Kompression durchgeführt werden, was aber serverseitig erfolgen muss. In [FGM+99] wird dies bereits in Abschnitt *3.5* vorgesehen und hier daher nicht mehr behandelt, vor allem aufgrund der Einschränkung, dass dies typischerweise nicht An- oder Ausgeschaltet wird, sondern unterstützt wird, oder nicht.

Die mit einer Webseite mitgelieferten eingebetteten Objekte enthalten oft Werbeelemente, Bilder oder Zusatzinformationen, ohne die der Inhalt einer Internetseite dennoch verständlich ist. Verwirft man zufällig eingebettete Objekte, kann immer noch ein Teilinformationsgehalt der Internetseite transportiert werden. Hiermit kann der zu sendende Verkehr reduziert werden - aber nur dann, wenn das verworfenee eingebettete Objekt für den Nutzer nicht notwendig war. Daher ist diese Adaption relativ kritisch zu betrachten, denn im Unterschied zu den anderen Adaptionen fehlen hierdurch Informationen, die auch nicht in reduzierter Form übertragen werden. Eine vorstellbare Lösung wären eingebettete Objekte in verschiedenen Qualitätsstufen einzuteilen, ähnlich wie *Stream Switching*.

4. Bewertung der Adaptionsmechanismen

> Latency matters. Amazon found every 100ms of latency cost them 1% in sales. Google found an extra .5 seconds in search page generation time dropped traffic by 20%.
>
> *(Gigaspaces)*

4.1. Simulationsszenario

Damit die in Kapitel 3.2 vorgeschlagenen Adaptionen evaluiert werden können, wurde eine Simulation durchgeführt. Der Simulationsaufbau wurde mit *IKR Simlib*[1] auf Javabasis vorgenommen. Auf der Basis von *Simlib* wurde ein Framework von *Frank Feller, IKR Uni Stuttgart* gebaut, was die Grundlage für die Implementierungen der Simulationen dieser Arbeit war.

In Abbildung 4.1 ist der Simulationsaufbau schematisch zu sehen. Auf der linken Seite sind Verkehrsgeneratoren dargestellt, welche jeweils nach den Modellen aus Kapitel 3 Verkehr generieren. Dieser Verkehr wird von Clients an Server übertragen. Client und Server bilden ein Endpunktpaar. Die Clients werden durch einen sogenannten *Access Link* mit einem gemeinsamen *Bottlenecklink* verbunden, welcher statistisch ausgewertet wird. Nach dem *Bottlenecklink* werden die Pakete von einem De-Multiplexer an die entsprechenden Endpunkte weitergeleitet. Außer dem *Bottlenecklink* existieren keine weiteren Beschränkungen, insbesondere nicht bei den *Access Links*. Entsprechend dem dargestellten Hinkanal existiert auch ein symmetrischer Rückkanal, damit die TCP Antwortpakete (*ACK*) nicht verloren gehen. Dieser Rückkanal ist aber nicht Bestandteil der Auswertungen.

- Die Round-Trip-Time des Szenarios beträgt 100ms.
- Die Queuegröße entspricht dem *Bandwidth-Delay-Product*[2].
- Die Bandbreite des Bottlenecklink wird in unterschiedlichen Simulationen variiert
- Jede Simulation ist untergliedert in eine *transiente Phase*, sowie 10 *Batches*, die zur statistischen Auswertung herangezogen werden.

Das erzeugte Verkehrsvolumen wurde an den *Cisco Visual Network Index* angepasst, um vergleichbare Ergebnisse zu erhalten.

[1] `http://www.ikr.uni-stuttgart.de/INDSimLib/`
[2] Bottlenecklinkkapazität * Bottlenecklinkverzögerung

Feedback und Überlastregelung Von der Queue, die sich vor dem Bottlenecklink befindet, gibt es ein *Feedback* über den aktuellen Füllstand an einen *Überlastregler*. Dieser Überlastregler berechnet über die jeweils letzten zehn Füllstände einen Durchschnitt (*moving-average*), um die Netzauslastung zu erfassen. In einem echtem Anwendungsszenario haben die Anwendungen auf den Endgeräten typischerweise kein direktes Feedback über die Netzauslastung. Aus diesem Grund wird das *moving-average* jeweils immer exklusive des aktuellen Wertes berechnet, wodurch versucht wird, die verzögerte Sicht zu simulieren. In der Realität kann hier beispielsweise mit dem Setzen des *Explicit Congestion Notification* Bits ([RFB01]) dieses Feedback erfolgen, anstatt des Mitteilens des genauen Füllstands. Da nach dem Setzen des ECN-Bits das Paket allerdings diese Information erst noch zum Ziel und von dort zurück zur Quelle transportiert werden muss, kommt dieses Feedback verzögert an. Der Überlastregler adaptiert die Verkehrsgeneratoren bei einem Füllstand von 30%. Das Ziel ist hierbei, die Überlast zu beenden indem weniger Verkehr gesendet wird.

Eine erneute kurze Zusammenfassung der Mechanismen aus Abschnitt 3.2:

- Web-Verkehr - Anzahl der eingebetteten Objekte verändern
- P2P Verkehr - Einsatz von LEDBAT, Anzahl der Sessions verändern
- Bulk-Verkehr - Einsatz von LEDBAT
- Video-Verkehr - Streamswitching
- VoIP-Verkehr - Streamswitching

Die Adaptionen finden jeweils abhängig der Netzlast (größer, oder kleiner 30%) in beide Richtungen statt. Man kann *negative*, sowie *positive* Adaptionen unterscheiden. *Negative* Adaptionen werden bei großer Netzlast durchgeführt und senken beispielsweise die erlaubte Sessionanzahl oder reduzieren das Videolevel. Das kommende Verkehrsvolumen soll reduziert werden, damit eine Überlast nicht verschlimmert sondern aufgehoben wird. Im Gegenzug wird bei geringer Netzlast eine *positive* Adaption durchgeführt, also beispielsweise die Bitrate erhöht, damit die gegebene Bandbreite optimal ausgenutzt wird. Im Falle, dass für die Verbindung volumenabhänige Kosten entstehen, kann man natürlich auf die *positive* Adaption verzichten.

4.1.1. Bewertungskriterien und Metriken

Als Ergebnisse der Simulation lassen sich Paketverluste pro Klasse betrachten. Diese liefern einen Eindruck darüber, wie stark das Netz überlastet ist. Sind die Verluste sehr hoch, so handelt es sich nicht nur um Verluste, wie sie durch die *TCP Congestion Control* auch im normalen Netzzustand auftreten. Hohe Verluste bedeuten erneutes Senden der Daten, was erneut zur Überlast führen kann. Die Bandbreite wird zwar nach wie vor ausgenutzt, die Übertragungsrate bleibt hoch, aber der *Goodput*[3] sinkt stark. Die verfügbare Bandbreite wird also nicht mehr sinnvoll ausgenutzt und Daten kommen verzögert an. Durch hohe Queueauslastungen steigt die *Round-Trip-Time*.

[3]die tatsächlichen übertragenen und empfangenen Informationen

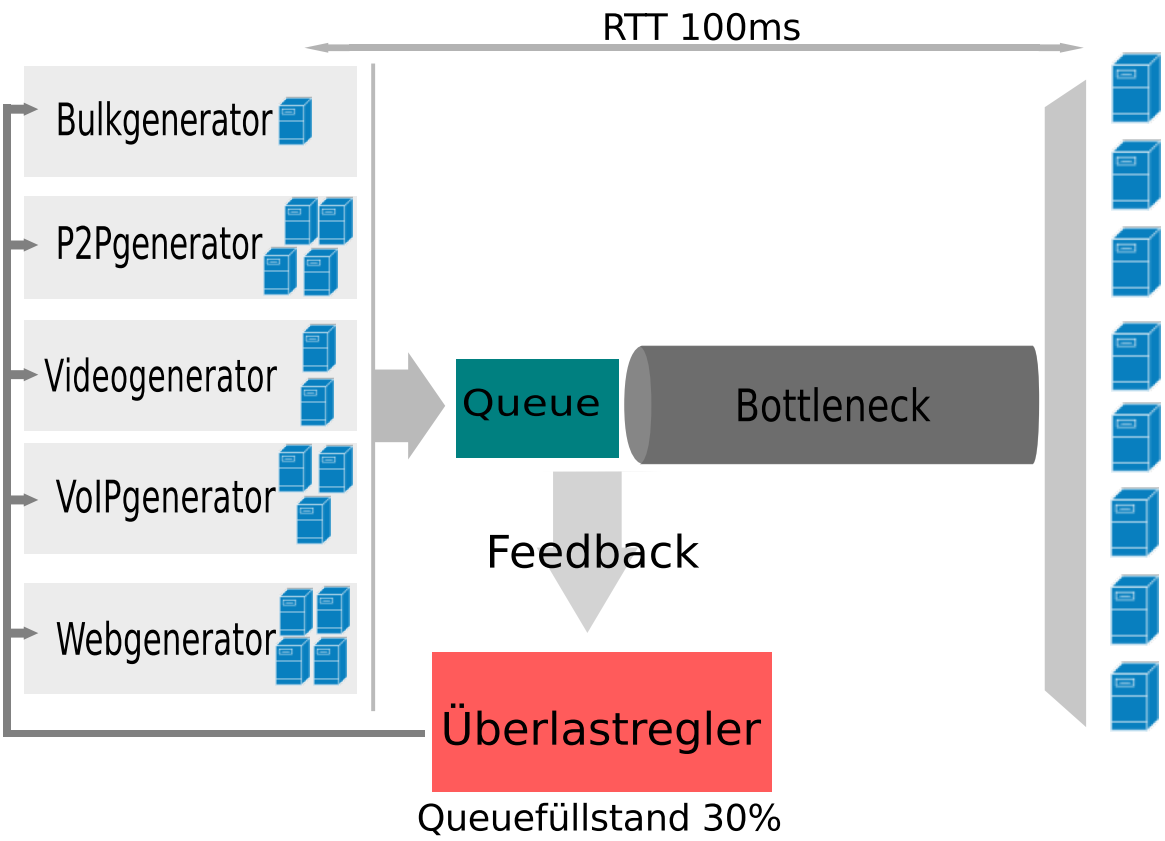


Abbildung 4.1.: Simulationsaufbau

Allein den Paketverlust pro Klasse zu untersuchen gibt aber nur eine sehr eingeschränkte Sicht auf das Szenario. Daher wurde das Simulationsszenario erweitert, so dass Videoübertragungen nicht nur auf Paketverluste untersucht werden. Der Empfänger verfügt über einen Ausspielpuffer, der von empfangenen Daten gefüllt wird. Ist dieser Puffer erstmalig gefüllt, so beginnt der Empfänger mit dem Ausspielen des Videos. Der Empfänger arbeitet hierbei in Ausspielschritten von einer ganzen Sekunde. Jedes mal, wenn der Pufferfüllstand zum Ausspielzeitpunkt leer ist, kommt es zu einem Pufferunterlauf (*Underrun*). Auf diese Weise erhält man bessere Einsicht, was Paketverlust für einen Anwender tatsächlich bedeutet.

Außerdem wird die tatsächliche Last auf dem *Bottlenecklink* angegeben, welcher möglichst hoch ausgelastet sein sollte, sowie die mittlere Wartezeit in der Bottleneckqueue.

Sandvine[4] führt in ihrem Bericht die Metrik der *Round-Trip-Time*, sowie die *Video Quality of Experience* als relevant auf. Von der alleinigen Analyse der Verlustrate raten sie ab. Da keine expliziten *Round-Trip-Time* Statistiken erfasst werden, wird die durch die unterschiedlich stark gefüllte Queue induzierte Verzögerung herangezogen.

4.2. Simulationsergebnisse

Auf den folgenden Seiten werden detaillierte Ergebnisse der Simulation vorgestellt. Hierbei wird jeweils eine Verkehrsklasse adaptiert. Es wird das Verkehrsvolumen pro Klasse vorgestellt, in Verbindung mit dem ursprünglichen Volumen ohne Adaption. Außerdem werden die Paketverluste pro Klasse in Prozent grafisch dargestellt, ebenfalls im Vergleich zur ursprünglichen Situation.

Volumenveränderung Es zeigt sich, dass sich aufgrund der Adaptionen das ursprüngliche Volumenverhältnis der Verkehrsklassen ändert. Dies ist zu erwarten, da die Adaptionen die adaptierten Klassen dazu veranlassen, weniger Verkehr zu erzeugen oder sich defensiver zu verhalten. Hierdurch können im Gegenzug andere Klassen die dadurch verfügbar gewordene Bandbreite ausnutzen. Aufgrund dessen wurden noch Messungen durchgeführt, bei denen versucht wird, das Volumen der adaptieren Klasse identisch zur nicht adaptierten Simulation zu halten. Der Verkehr wird dadurch auf ein bestimmtes Verkehrsklassenvolumen normiert, was die Simulationszeit beeinflusst. Durch diese Normierung ist nicht zu erkennen, wie das Verkehrsvolumen durch die Adaption vom Standardvolumen (Cisco VNI) abweicht. Dafür kann man das Verhalten der nicht adaptierten Verkehrsklassen besser betrachten. Verlustbetrachtungen pro Klasse können dann allerdings nicht mehr auf die selbe Weise betrachtet werden. Dies liegt daran, dass nun beispielsweise bei der Verkehrsklasse Video zwei identische Verkehrsvolumina einer stark unterschiedlichen Menge an Videodauer entsprechen. Eine Möglichkeit ist es, nicht nur die Pufferunterläufe zu zählen, sondern ebenso die erfolgreich ausgespielten Videosegmente und anschließend diese beiden Werte in ein Verhältnis zu setzen. Die nicht normierten Simulationsergebnisse sind alle mit der selben Simulationszeit entstanden. 600s für die Batches, sowie 30s für die transiente Phase.

[4] `http://www.sandvine.com/news/global_broadband_trends.asp`

Ergebnisse bei Simulation mit gleicher Dauer

Auswertung bei 20Mbit/s Bottleneck In Abbildung 4.2 werden die Messergebnisse aus sieben Simulationen für einen einfacheren Überblick zusammengestellt. Zu sehen sind im oberen Schaubild die verschiedenen Verkehrsvolumina, die aufgrund der verschiedenen Adaptionsmöglichkeiten entstanden sind. Das untere Schaubild zeigt die entstandenen Verluste pro Verkehrsklasse.

Auf den ersten Blick fällt die entstandene und erwartete Verschiebung der Verkehrsvolumina auf. Durch die jeweilige Adaption einer Klasse sendet diese Klasse weniger Daten und somit können die anderen Verkehrsklassen diese freigewordene Bandbreite nutzen.

Statistiken zur Linkauslastung finden sich in Tabelle 4.2. Tabelle 4.1 fasst die Pufferunterläufe zusammen.

Adaption des Videoverkehrs Detaillierte Ergebnisse sind in A.0.3 zu finden. Es ist zu sehen, dass durch die Adaption des Videoverkehrs (Anpassung der Videolevel durch Streamswitching) das Videovolumen stark sinkt und der prozentuale Anteil der Verluste ebenso verkleinert wird. Die Linkauslastung ist vergleichbar zur nicht adaptierten Simulation, die Auslastung der Queue vor dem Bottleneck sinkt jedoch um ca 13%. Dies erklärt die gemessenen besseren absoluten Verlustraten. Es können keine Videopufferunterläufe mehr beobachtet werden (vgl. 4.1), wodurch diese Adaption als sehr sinnvoll betrachtet werden kann, da sie einen direkten Einfluss auf die *Quality of Experience* des Nutzers hat.

Adaption des Peer-to-Peer Verkehrs Durch seinen sehr großen Verkehrsanteil wurde mit einer starken Verbesserung der Ergebnisse durch Adaption gerechnet. Dies ist zum Teil eingetreten: Das Peer-to-Peer Volumen sinkt und die zugehörigen prozentualen Verluste sinken enorm. Bis auf die Klasse Web kann von den einzelnen anderen Klassen mehr Verkehr gesendet werden. Die gemessenen Pufferunterläufe bei der Videoübertragung sinken im Vergleich zum nicht adaptierten Szenario.

Die Linkauslastung sowie die Queueauslastung sind leicht gesunken im Vergleich zum nicht adaptierten Fall.

Insgesamt betrachtet wäre hier eine noch stärkere Verbesserung der Messergebnisse erwartet worden. Eine Auswertung mit 40Mbit/s Bottlenecklink zeigt ähnliche Ergebnisse. Betrachtet man jedoch die Pufferunterläufe in Tabelle 4.5 mit einem 80Mbit/s Bottlenecklink, so zeigt sich hier eine starke Verbesserung.

Adaption des Webverkehrs Die Messergebnisse der Webadaption sind schwerer zu erklären. Erwartungsgemäß sinkt das Webverkehrsvolumen durch das Verwerfen einzelner eingebetteter Objekte, vor allem, da diese im Verhältnis zur eigentlichen Webseite größer sind. Die prozentualen Verluste der Klasse steigen aber. Dies kann vielleicht darauf zurück geführt werden, dass nun häufiger kleinere Übertragungen stattfinden. Die Verluste der anderen

Klassen sinken leicht. Durch die Erhöhung des Videovolumens ist die leicht gestiegene Zahl der Pufferunterläufe zu erklären. Die Adaption insgesamt muss kritisch betrachtet werden, vor allem durch das Fehlen mancher eingebetteten Objekte für den Endnutzer.

Adaption des VoIP Verkehrs Durch den sehr kleinen Anteil am Gesamtverkehr konnten durch eine Adaption des VoIP Verkehrs keine starken Verbesserungen erwartet werden. Möglicherweise hätte eine Auswertung über die Verbindungsqualität ähnlich dem Videoverkehr stärkere Verbesserungen gezeigt. Hier lässt sich nur feststellen, dass die VoIP Verluste gesunken, dafür die anderen Verluste leicht gestiegen sind. Auf die Videoübertragungen sowie die Link- und Queueauslastung hatte die Adaption praktisch keinen Einfluss.

Adaption des Bulk Verkehrs Die Adaption des Bulkverkehrs liefert bei einem 20 Mbit/s Bottlenecklink unbefriedigende Ergebnisse. Obwohl sich der Bulkverkehr nun defensiver verhält, nehmen die Verluste in allen Klassen zu. Das Videovolumen nimmt leicht zu, ebenso aber die Pufferunterläufe bei der Videoausspielung. Der Link, sowie die Queue bleiben stark ausgelastet. Erhöht man die Bandbreite des Bottlenecklinks auf 40 Mbit/s sehen die Ergebnisse anders aus (Tabelle 4.4, sowie A.0.8). Hier zeigt sich eine kleinere Verlustrate, weniger Pufferunterläufe und ein niedrigeres Volumen an Bulkverkehr. Dies wurde auch schon bei 20 Mbit/s erwartet und kann hier nicht eindeutig geklärt werden.

Adaption aller Klassen Durch die Adaption aller Verkehrsklassen sinken fast alle Verkehrsvolumina leicht. Die gemessenen Verluste sinken sehr stark. Das übertragene Volumen des Bulkverkehrs nimmt leicht zu, was sich in diesem Szenario nicht eindeutig erklären lässt. Die Auslastung der Queue, welche sich vor dem Bottleneck befindet, sinkt deutlich, die Linkauslastung fällt auf ungefähr 85%. Wie auch bei der alleinigen Videoadaption treten hier keine Pufferunterläufe bei der Videoübertragung mehr auf. Durch die deutlich geringere durchschnittliche Queueauslastung sinkt die Zeit für die Übertragung deutlich. Dies bewirkt eine kleine Round-Trip-Time und ist für zeitkritische Anwendungen wie beispielsweise Videostreaming oder auch Online-Gaming sehr interessant.

Da sich die Ergebnisse eines 20 Mbit/s Bottlenecklinks nicht systematisch von denen eines 40 Mbit/s Bottlenecklinks unterscheiden, wurde verzichtet, letztere komplett im Appendix grafisch darzustellen. Lediglich die Volumina des Bulkverkehrs weichen hier ab und werden daher beinhaltet. Eine Gesamtübersicht über die Ergebnisse findet sich in Abbildung 4.3.

Man kann festhalten, dass generell der Ansatz, Verkehr zu Adaptieren große Verbesserungen bringt, wenn auch nicht bei allen Adaptionen zum gleichen Teil. Besonders hervorzuheben ist die Adaption des Videoverkehrs, die Pufferunterläufe komplett beseitigt. Ist man daran interessiert, die absoluten Verkehrsverluste zu reduzieren und damit allgemein mehr *Goodput* zu erhalten, ist die Adaption des Peer-to-Peer Verkehrs eine gute Möglichkeit hierzu.

Ergebnisse bei Simulation mit normierten Volumen bei 20 Mbit/s Bottlenecklink

Neben der Simulation begrenzt durch eine bestimmte Zeit, wurden Messungen durchgeführt, die das Volumen einer einzelnen Verkehrsklasse normieren. Dies bedeutet, dass als Abbruchbedingung das Erreichen eines festgelegten Volumens pro Klasse verwendet wurde. Damit erhält man dasselbe Verkehrsvolumen in dieser Klasse für den adaptierten sowie den nicht adaptierten Fall.

Interessant wäre vielleicht noch eine Untersuchung mit größerem Bottlenecklink gewesen, dies war zeitlich allerdings nicht mehr möglich.

Die Ergebnisse finden sich in A.0.9. Eine tabellarische Übersicht über die Verluste gibt Tabelle 4.6.

Normierter Bulkverkehr Durch die Adaption des Bulkverkehrs kann nun mehr Verkehr aus allen anderen Klassen übertragen werden. Die Paketverluste steigen, allerdings nicht so stark wie das übertragene Volumen, siehe Abschnitt A.0.9. Tabelle 4.6 zeigt, dass fast die doppelte Menge an Videoinformationen ausgespielt wurde, die Pufferunterläufe konnten hier aber nicht verbessert werden.

Normierter VoIP Verkehr Durch Adaption des VoIP Verkehrs stieg das übertragene Volumen der anderen Klassen leicht an, siehe Abschnitt A.0.9. Allerdings stiegen die Verluste mit Ausnahme der Klasse Bulk und VoIP ebenso leicht an. Es wurde mehr Videoinformation ausgepielt, die Pufferunterläufe stiegen aber, hier kann keine Verbesserung festgestellt werden.

Normierter Videoverkehr Vor Adaption des Videoverkehrs führten fast die Hälfte der Ausspielversuche zu einem Pufferunterlauf. Nach Adaption wurde dies auf Null reduziert. Sowie mit demselben Volumen konnte die dreifache Dauer an Videoinformationen ausgespielt werden. Abschnitt A.0.9 zeigt erfreuliche Ergebnisse. Die übertragenen Volumina stiegen durch die Adaption stark an und die prozentualen Verluste pro Klasse konnten durchgehend gesenkt werden.

Normierter Webverkehr Die Adaption des Webverkehrs hat zur Folge, dass in den anderen Verkehrsklassen deutlich mehr Volumen übertragen wird, bis dieselbe Datenmenge in der Klasse Web übertragen ist. Dies war zu erwarten, da das Verkehrsvolumen durch Verwurf mancher eingebetteter Objekte stark sinkt. Die Paketverluste stiegen erfreulicherweise nicht in der selben Größenordnung wie das Volumen. Eine Auswertung der Pufferunterläufe ist schwer, Tabelle 4.6 zeigt, dass vor der Adaption nahezu keine Videoinformationen übertragen wurden, weshalb ein Vergleich nicht sinnvoll erscheint.

Volumen und Verluste bei 20 Mbit/s Bottleneck

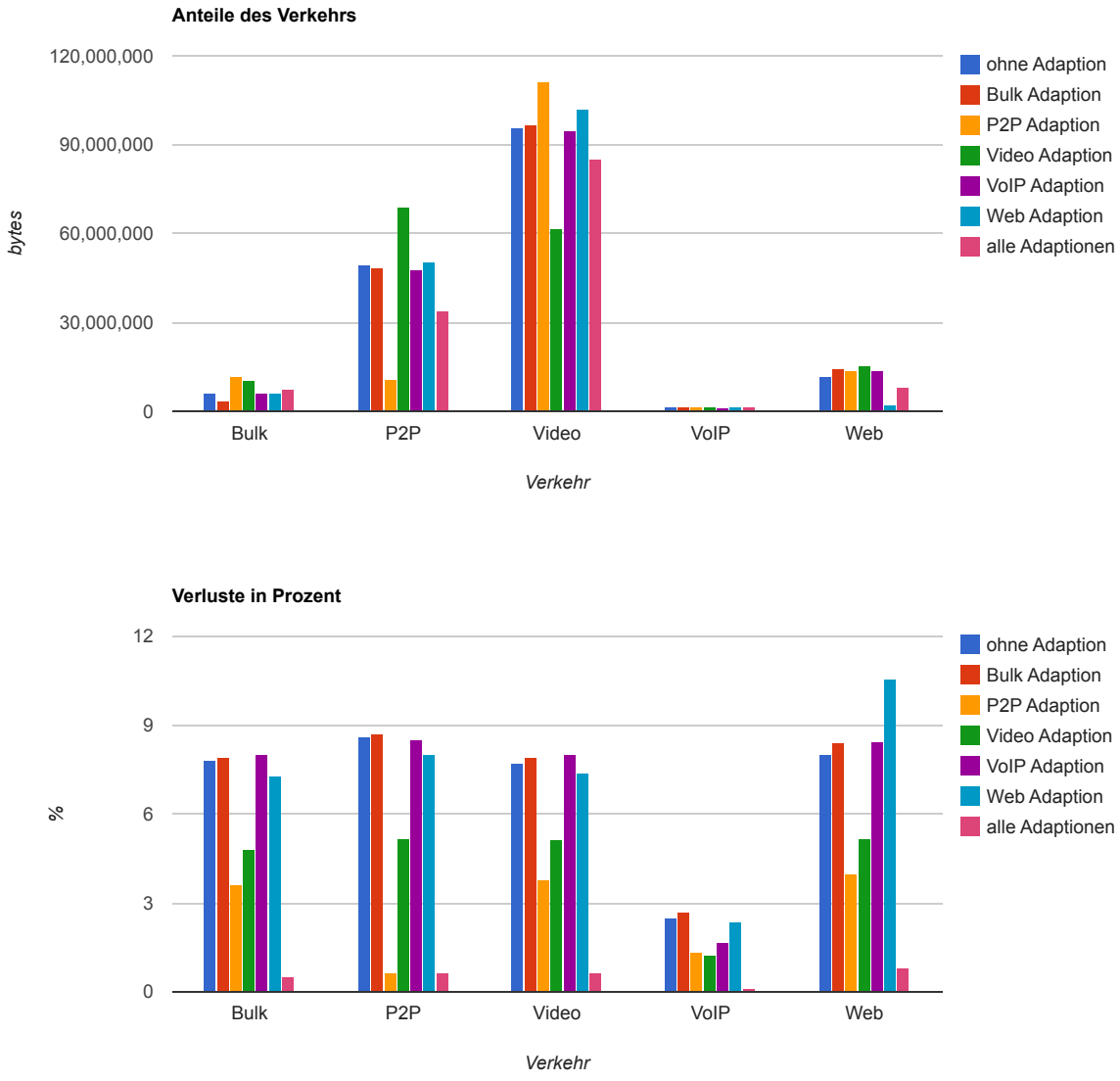


Abbildung 4.2.: Auswertung eines 20 Mbit/s Bottlenecklinks

Volumen und Verluste bei 40 Mbit/s Bottleneck

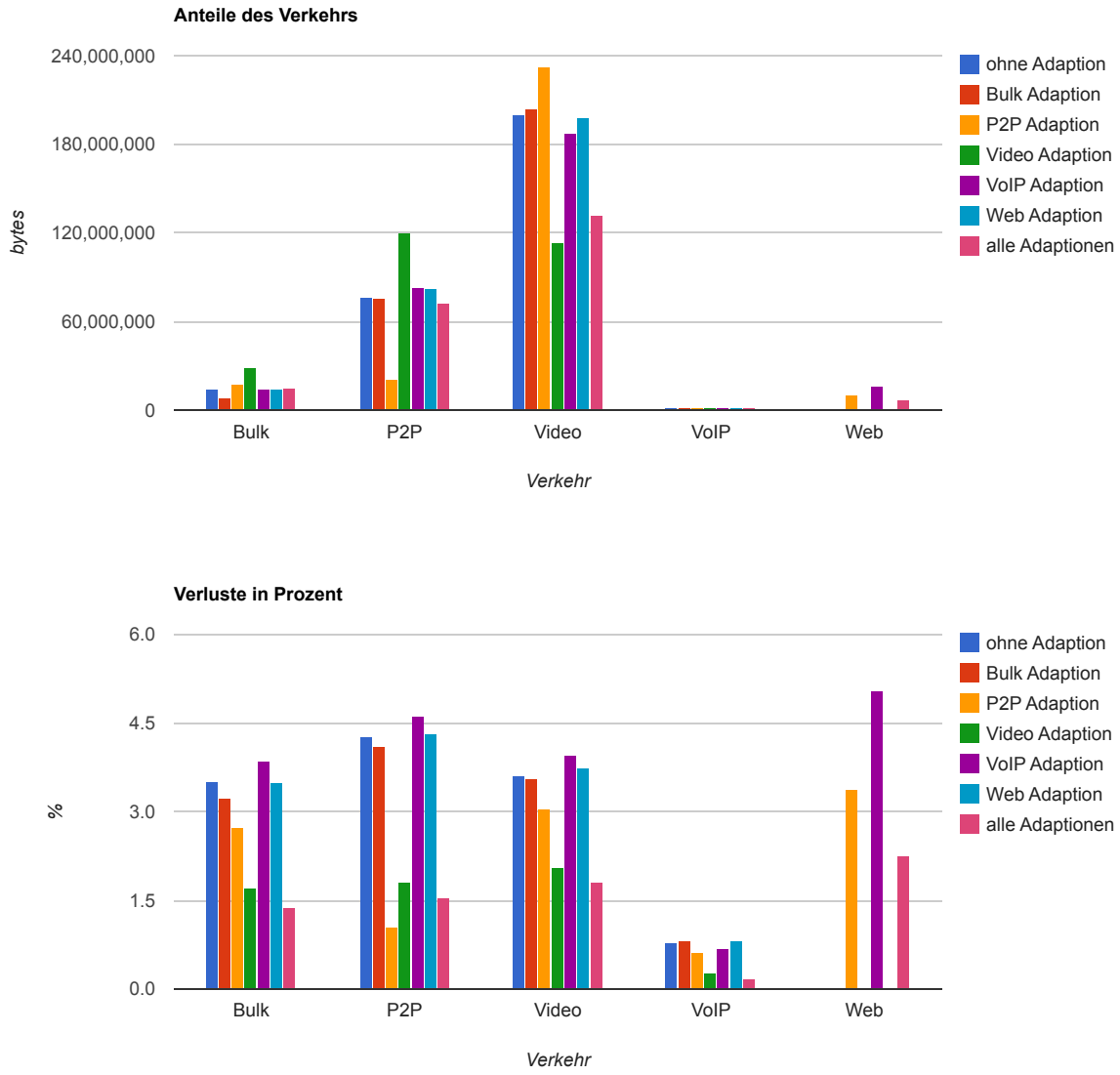


Abbildung 4.3.: Auswertung eines 40 Mbit/s Bottlenecklinks

Volumen und Verluste bei 80 Mbit/s Bottleneck

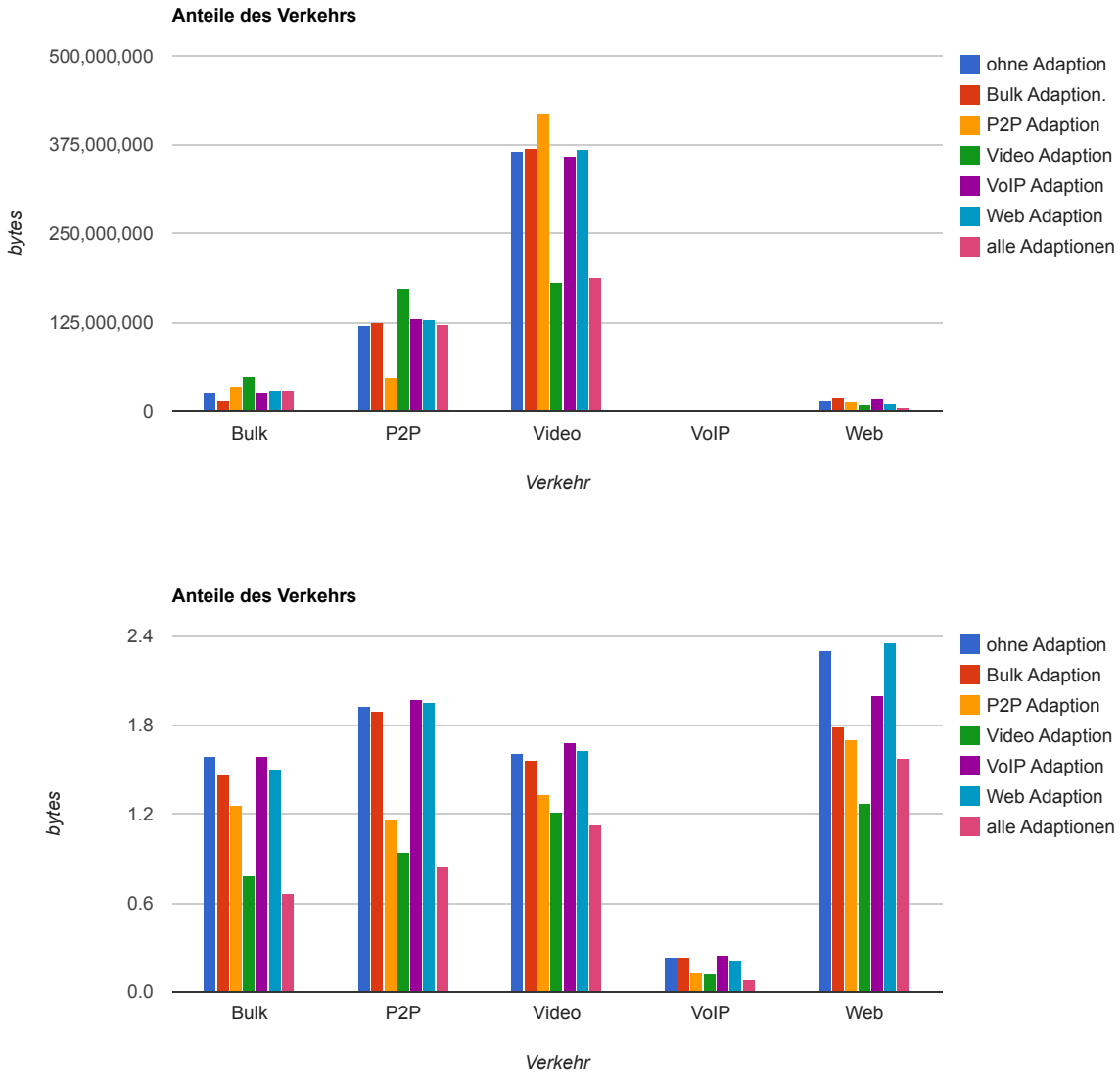


Abbildung 4.4.: Auswertung eines 80 Mbit/s Bottlenecklinks

Adaption	Pufferunterläufe in s	Puffertreffer in s
Ohne Adaption	3004.1	3167.2
mit Bulk Adaption	3116.1	3275.0
mit P2P Adaption	2882.6	3214.7
mit Video Adaption	0.0	3195.9
mit VoIP Adaption	3037.4	3195.3
mit Web Adaption	3036.7	3206.7
mit allen Adaptionen	0.0	3241.8

Tabelle 4.1.: beobachtete Videopufferunterläufe bei 20Mbit/s Bottleneck

Adaption	Queueauslastung in %	Linkauslastung in %
Ohne Adaption	83.96	99.3
mit Bulk Adaption	84.54	99.3
mit P2P Adaption	67.98	94.83
mit Video Adaption	70.95	98.9
mit VoIP Adaption	83.21	98.5
mit Web Adaption	81.71	98.1
mit allen Adaptionen	34.02	89.40

Tabelle 4.2.: Auslastung abhängig der Adaption, bei 20 Mbit/s Bottleneck

Adaption	Queueauslastung in %	Linkauslastung in %
Ohne Adaption	65.97	92.99
mit Bulk Adaption	63.95	92.13
mit P2P Adaption	60.33	90.62
mit Video Adaption	33.53	85.73
mit VoIP Adaption	68.44	95.87
mit Web Adaption	66.58	94.10
mit allen Adaptionen	23.17	74.07

Tabelle 4.3.: Auslastung abhängig der Adaption, bei 40 Mbit/s Bottleneck

Adaption	Pufferunterläufe in s	Puffertreffer in s
Ohne Adaption	2739.4	3204.2
mit Bulk Adaption	2677.1	3196.0
mit P2P Adaption	2422.5	3179.6
mit Video Adaption	0.0	3224.5
mit VoIP Adaption	2774.2	3215.5
mit Web Adaption	2743.5	3207.0
mit allen Adaptionen	0.0	3211.4

Tabelle 4.4.: beobachtete Videopufferunterläufe bei 40Mbit/s Bottleneck

Adaption	Pufferunterläufe in s	Puffertreffer in s
Ohne Adaption	1483.0	3137.5
mit Bulk Adaption	1399.9	3105.6
mit P2P Adaption	965.6	3189.6
mit Video Adaption	0.0	3201.1
mit VoIP Adaption	1554.7	3164.1
mit Web Adaption	1587.4	3230.8
mit allen Adaptionen	0.0	3249.3

Tabelle 4.5.: beobachtete Videopufferunterläufe bei 80Mbit/s Bottleneck

Normierte Verkehrsklasse	Pufferunterläufe in s ohne Adaption	Puffertreffer in s ohne Adaption	Pufferunterläufe in s mit Adaption	Puffertreffer in s mit Adaption
Bulk	6004.5	6207.3	10327.3	10571.8
Video	47.9	103.4	0.0	332.5
VoIP	500.4	594.3	852.4	955.5
Web	0.0	7.3	247.3	329.3

Tabelle 4.6.: Pufferstatistiken bei normierten Verkehrsklassen mit 20 Mbit/s Bottleneck

5. Zusammenfassung und Ausblick

Zusammenfassung Diese Arbeit wurde motiviert durch die wachsende Belastung für die Infrastruktur und den Folgen für den Endanwender aufgrund des stark ansteigenden Verkehrsvolumen im Internet. Es wurde zu Beginn festgestellt, dass die Bandbreitenverwaltung des *Transmission Control Protocols* nur zu einer Fairness zwischen einzelnen Verbindungen, nicht jedoch zu einer Fairness zwischen Anwendungen führt. Dies ist besonders problematisch für Videostreaming und andere Echtzeitanwendungen, die durch Peer-to-Peer Verkehr benachteiligt werden. Das Ziel der Arbeit war es, bei Auftreten einer Überlastsituation diese aktiv aufzuheben, anstatt sie weiter zu verschlimmern. Hierfür wurden Adaptionsmechanismen untersucht, welche abhängig der Netzlast versuchten, die aktuell verfügbare Bandbreite sinnvoll auszunutzen. Der Verkehr einzelner Anwendungen wurde in Klassen zusammengefasst. Anschließend wurde das Sendeverhalten dieser Klassen modelliert. Um die Vergleichbarkeit dieser Modelle zu steigern, wurden die enstehenden Verkehrsvolumina an die des *Cisco Visual Network Index* angepasst. Die vorgeschlagenen Adaptionsmechanismen wurden implementiert, simuliert und die Ergebnisse in dieser Arbeit präsentiert.

Es zeigten sich bei der Adaption des Videoverkehrs exzellente Ergebnisse, was Verlustraten und Pufferunterläufe betrifft. Diese Verbesserung ist für einen Endnutzer direkt feststellbar. Eine Adaption des Peer-to-Peer Verkehrs konnte die allgemeinen Verlustraten stark senken und führte zu einer besseren Latenzzeit. Weniger große Verbesserungen konnten bei einer Adaption des VoIP- sowie Webverkehrs festgestellt werden. Durch den geringen Verkehrsanteil am Gesamtvolumen war dies aber auch nicht zu erwarten. Eine Adaption des Bulkverkehrs lieferte bei 20 Mbit/s eine Verschlechterung der Ergebnisse, dies wurde so nicht erwartet und konnte nicht erklärt werden. Bei einem Bottlenecklink von 40 Mbit/s zeigten sich aber positive Ergebnisse.

Ausblick Mit dieser Arbeit wurden Grundsteine für die Auswertung der Verkehrsadaption gelegt. Teilweise wurden Problemstellungen für die Simulation vereinfacht, aufgrund der begrenzten Bearbeitungszeit. Ein wichtiger zu beachtender Punkt ist das Feedback zur Überlastsituation, welches den Adaptionsregler in einem echten Szenario erst verzögert erreicht. In dieser Diplomarbeit wurde jeweils das neueste erhaltene Feedback nicht in die Berechnungen miteinbezogen, um die Verzögerung von mindestens einer Paketumlaufzeit, wie sie beispielsweise bei Explicit Congestion Notification auftritt, zu simulieren. Dies kann die verzögerte Sicht aber nur emulieren, nicht genau abbilden, weshalb an diesem Punkt weitere Untersuchungen mit einem echten Feedback notwendig sind. Die Messergebnisse der Adaption des Bulkverkehrs bei einem kleineren Bottlenecklink widersprachen den Erwartungen. Daher sollte hier das Szenario gegebenenfalls genauer untersucht werden.

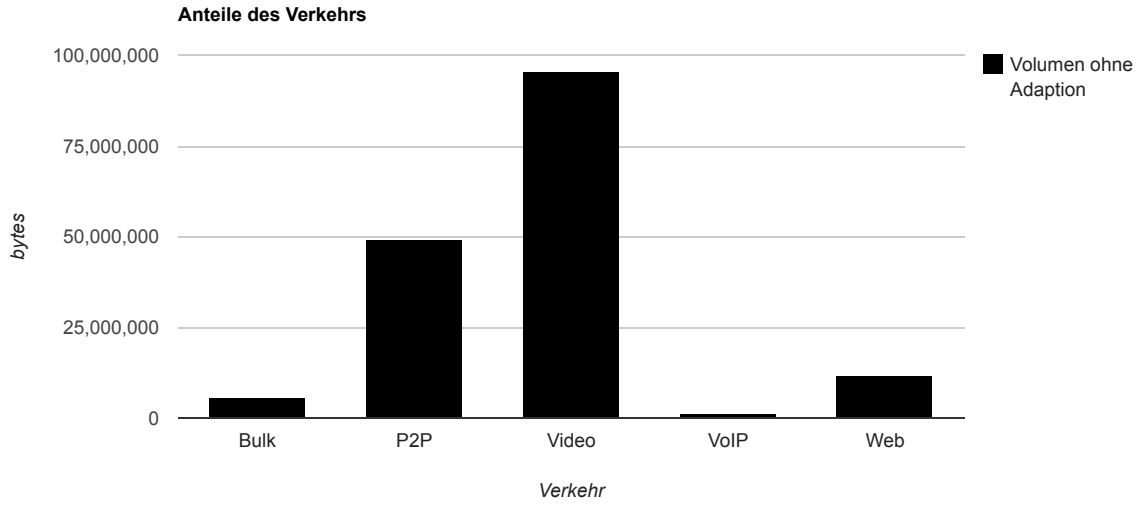
Mit der Modellierung der Verkehrsklassen wurde ein wichtiger Schritt für künftige Untersuchungen gemacht. Diese künftigen Untersuchungen können einen größeren Fokus auf die Auswertung der Verkehrsmodelle legen, insbesondere gilt es, die Adaptionsmechanismen mit weiteren Metriken genauer zu verstehen, hier sei beispielsweise auf die Auswertung der VoIP Verbindungsgüte hingewiesen.

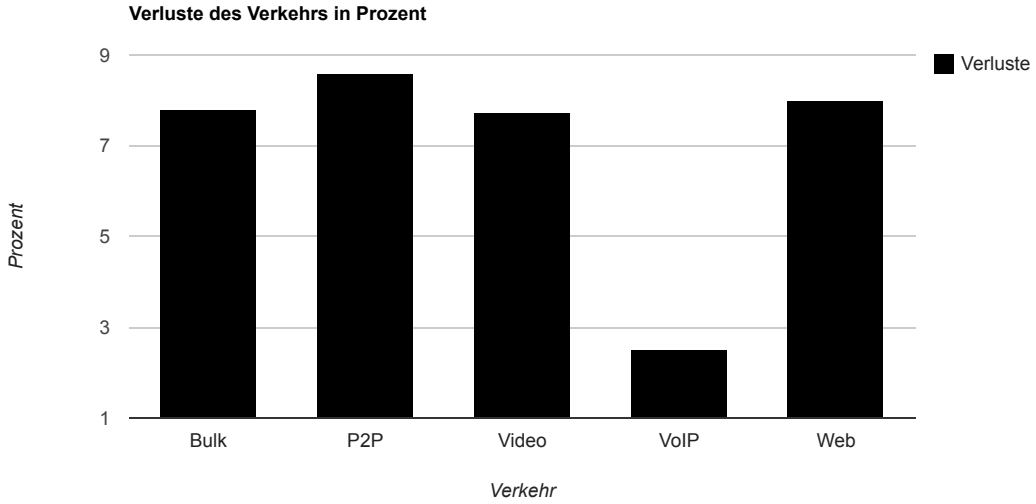
Als Nebenprodukt der Adaptionsauswertung entstanden die Modelle des Verkehrsklassen, sowie deren Anpassung an die im Internet typischen Volumina. Von diesen Modellen können weitere Simulationen profitieren.

A. Appendix

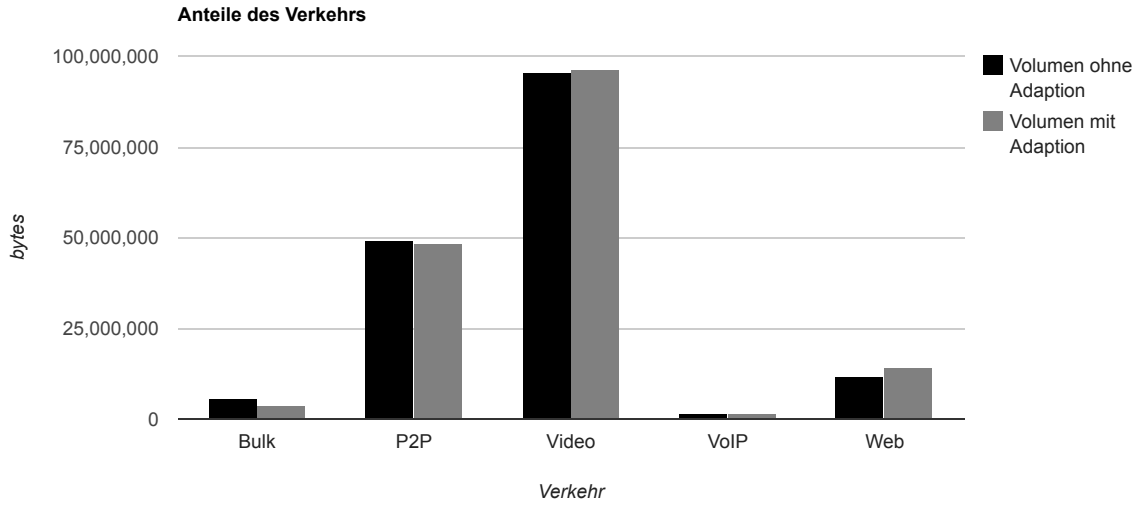
A.0.1. Ergebnisse ohne Adaption

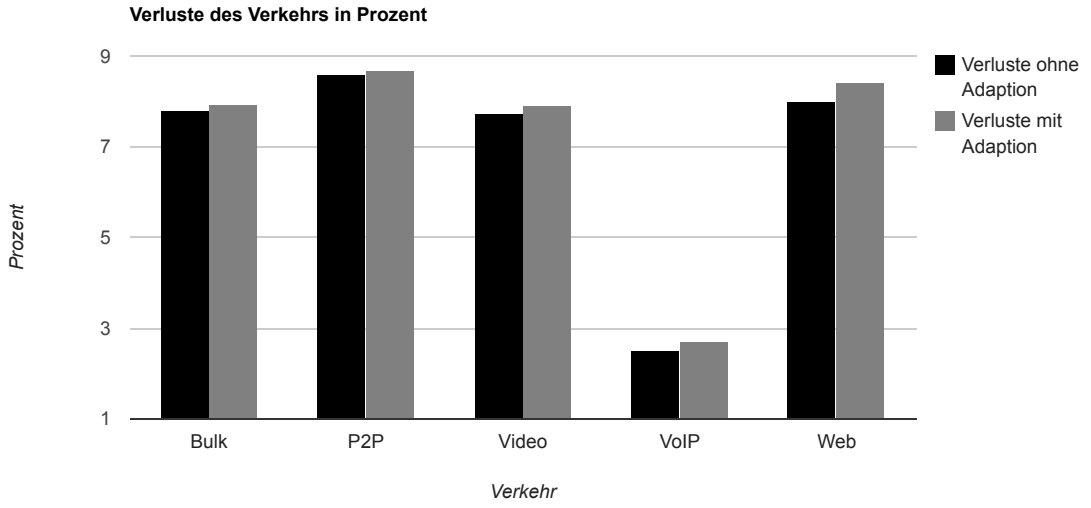
20 Mbit/s





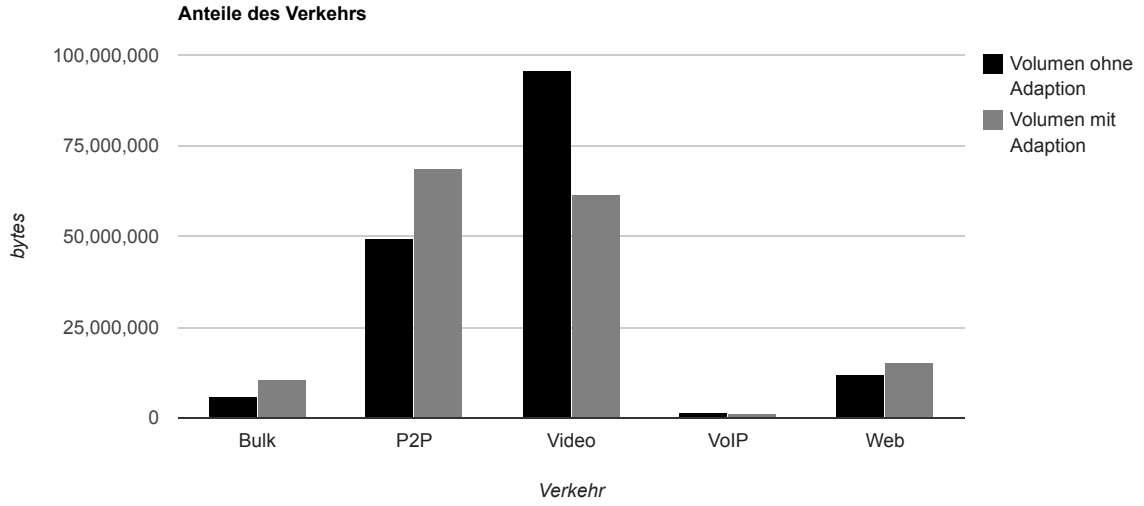
A.0.2. Ergebnisse mit Adaption des Bulkverkehrs

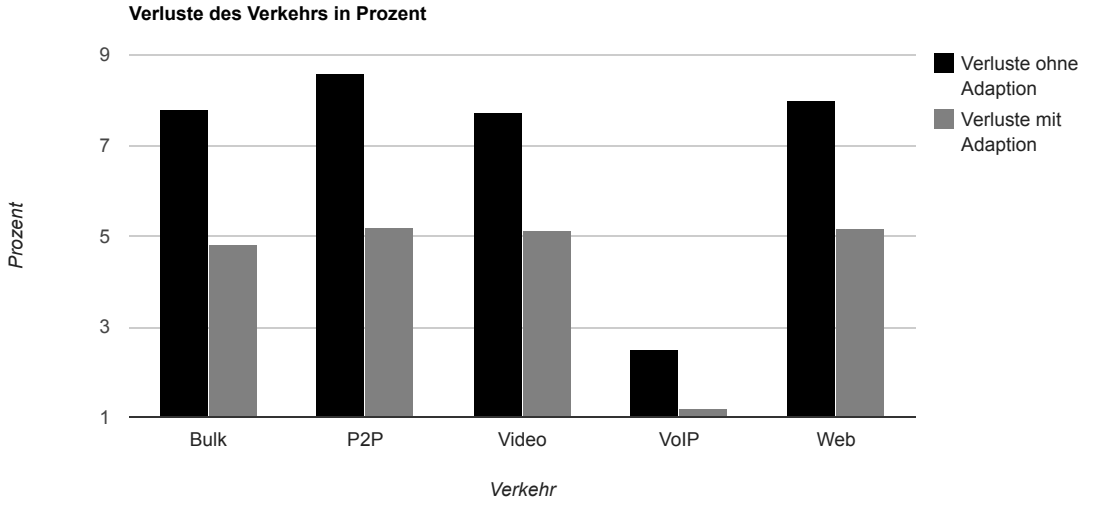




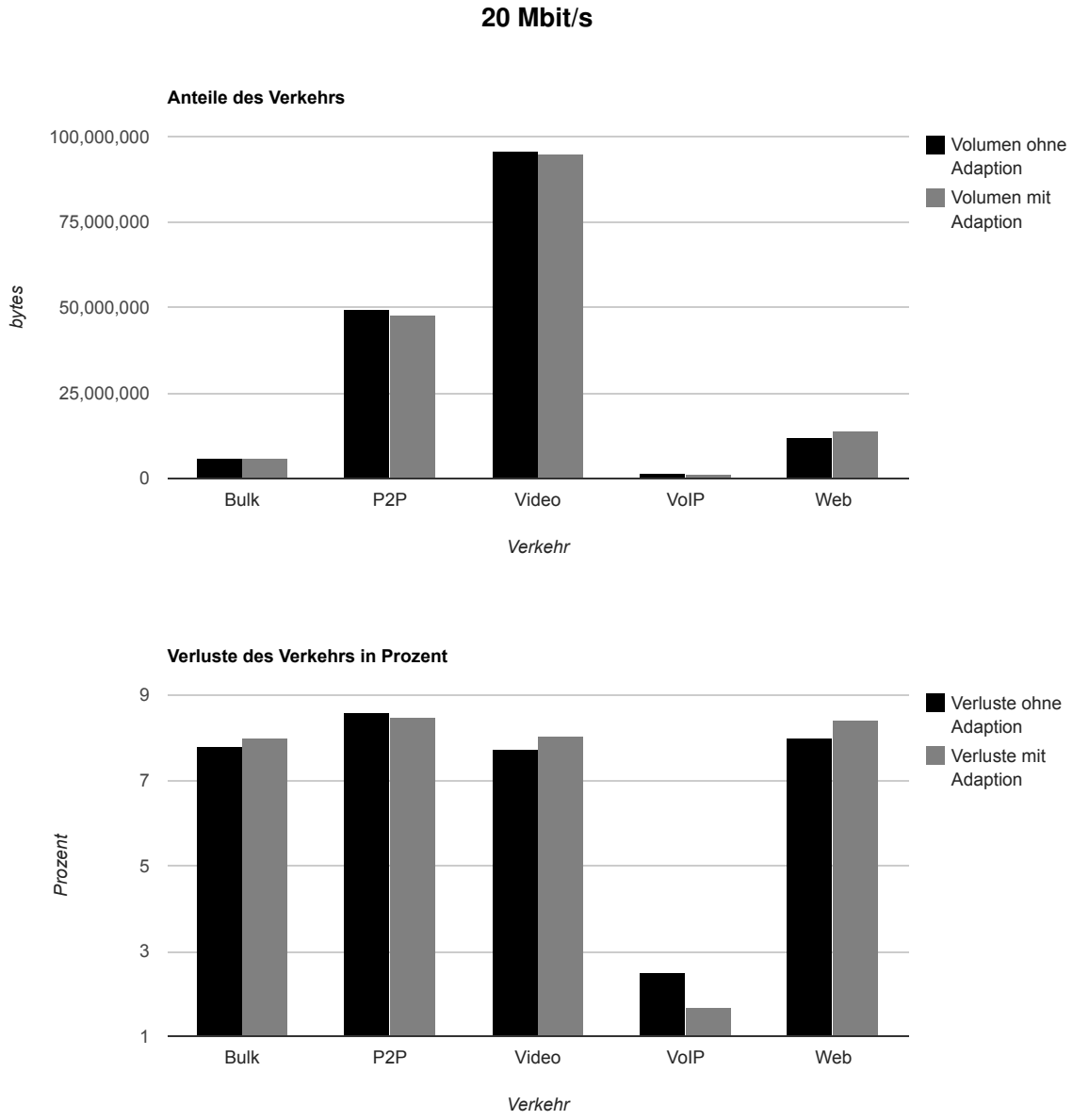
A.0.3. Ergebnisse mit Adaption des Videoverkehrs

20 Mbit/s



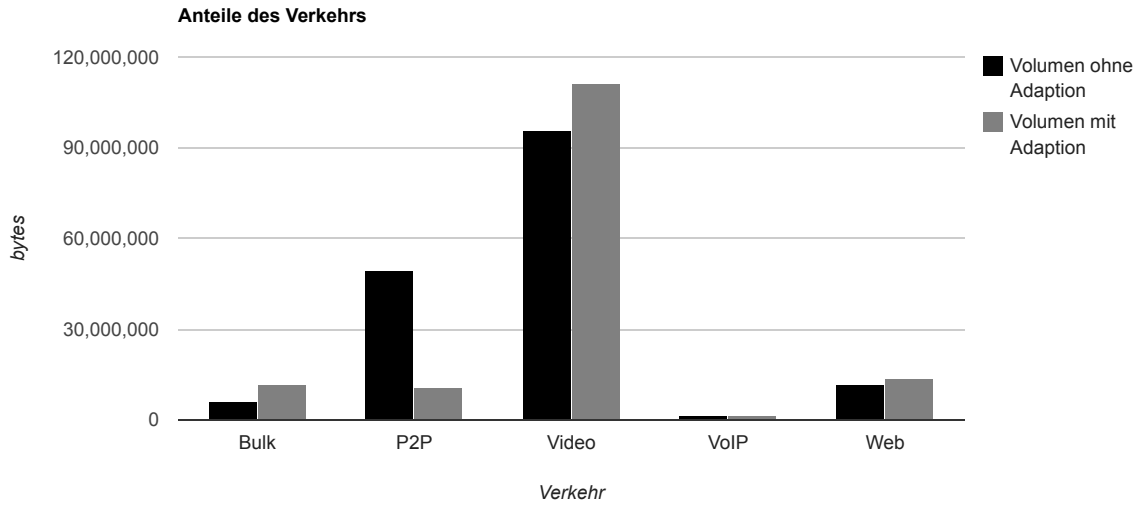


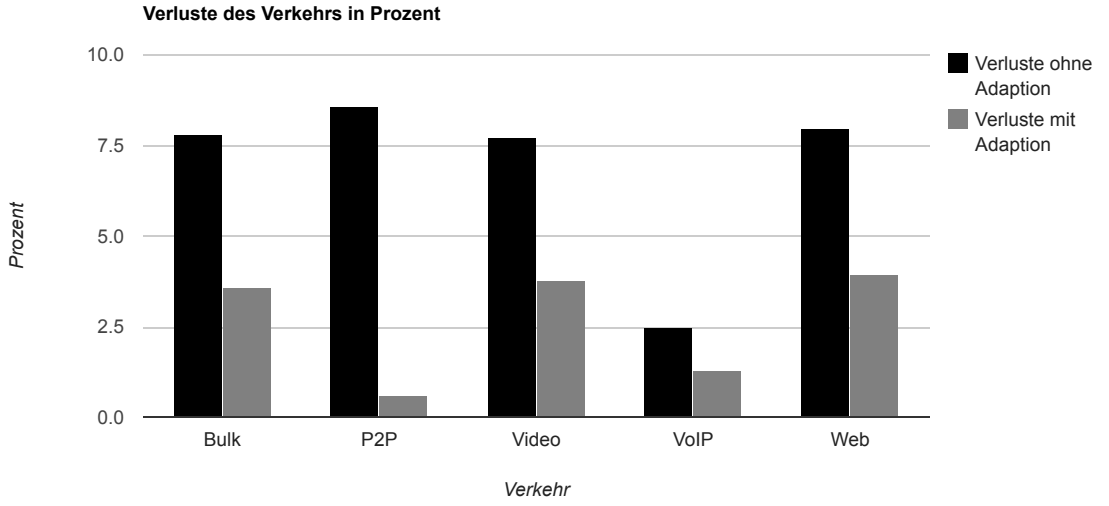
A.0.4. Ergebnisse mit Adaption des VoIP Verkehrs



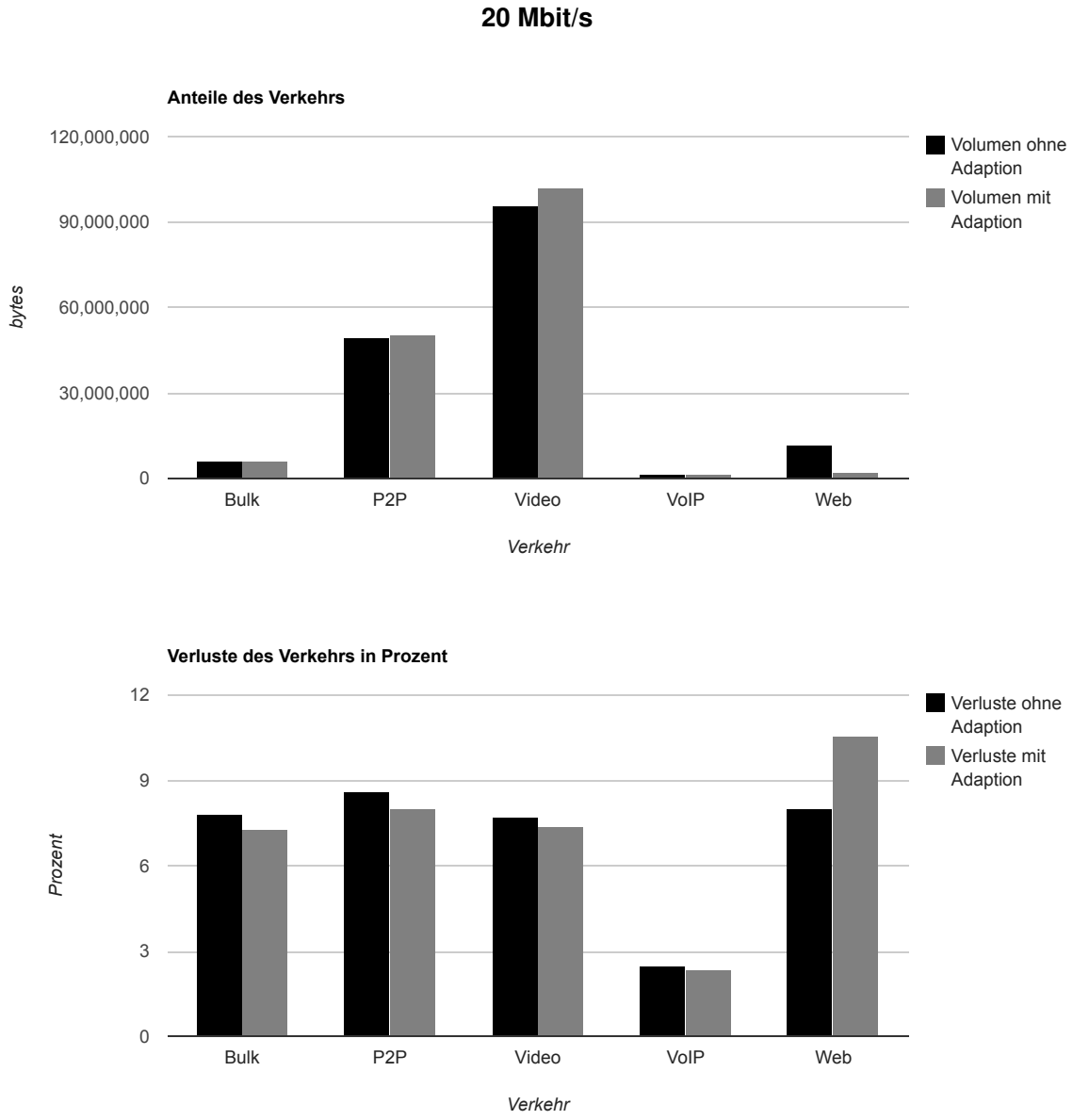
A.0.5. Ergebnisse mit Adaption des Peer-to-Peer Verkehrs

20 Mbit/s



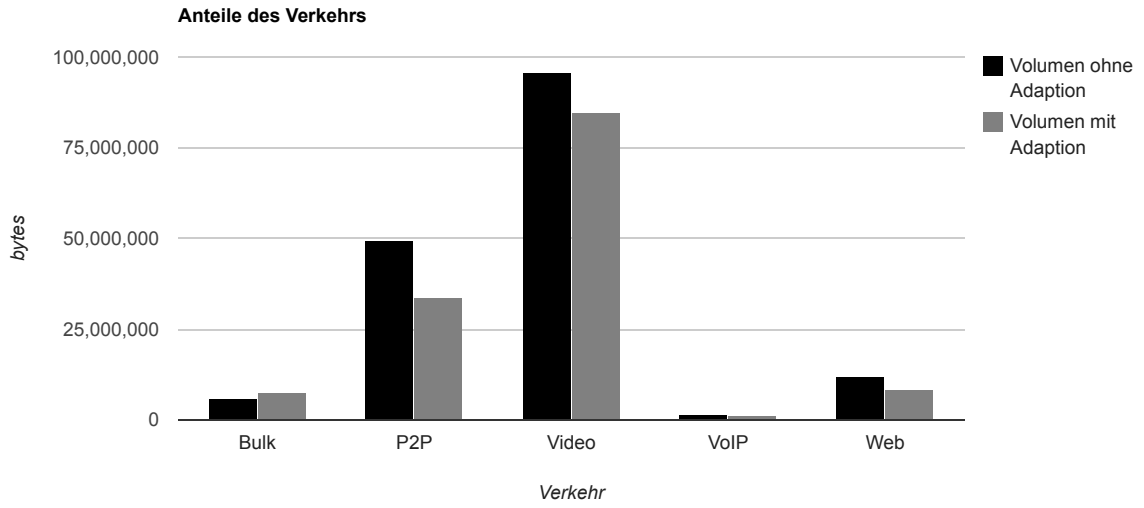


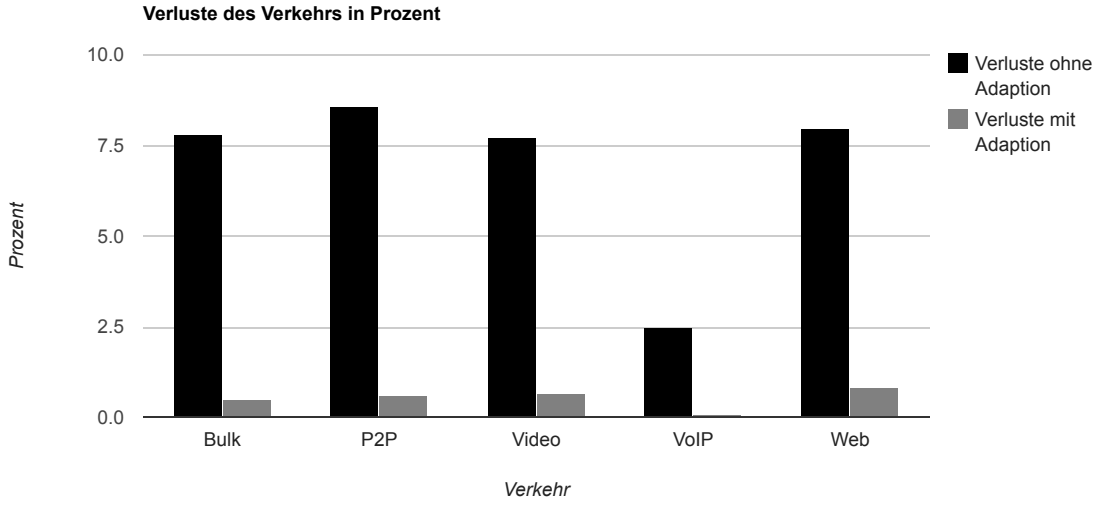
A.0.6. Ergebnisse mit Adaption des Webverkehrs



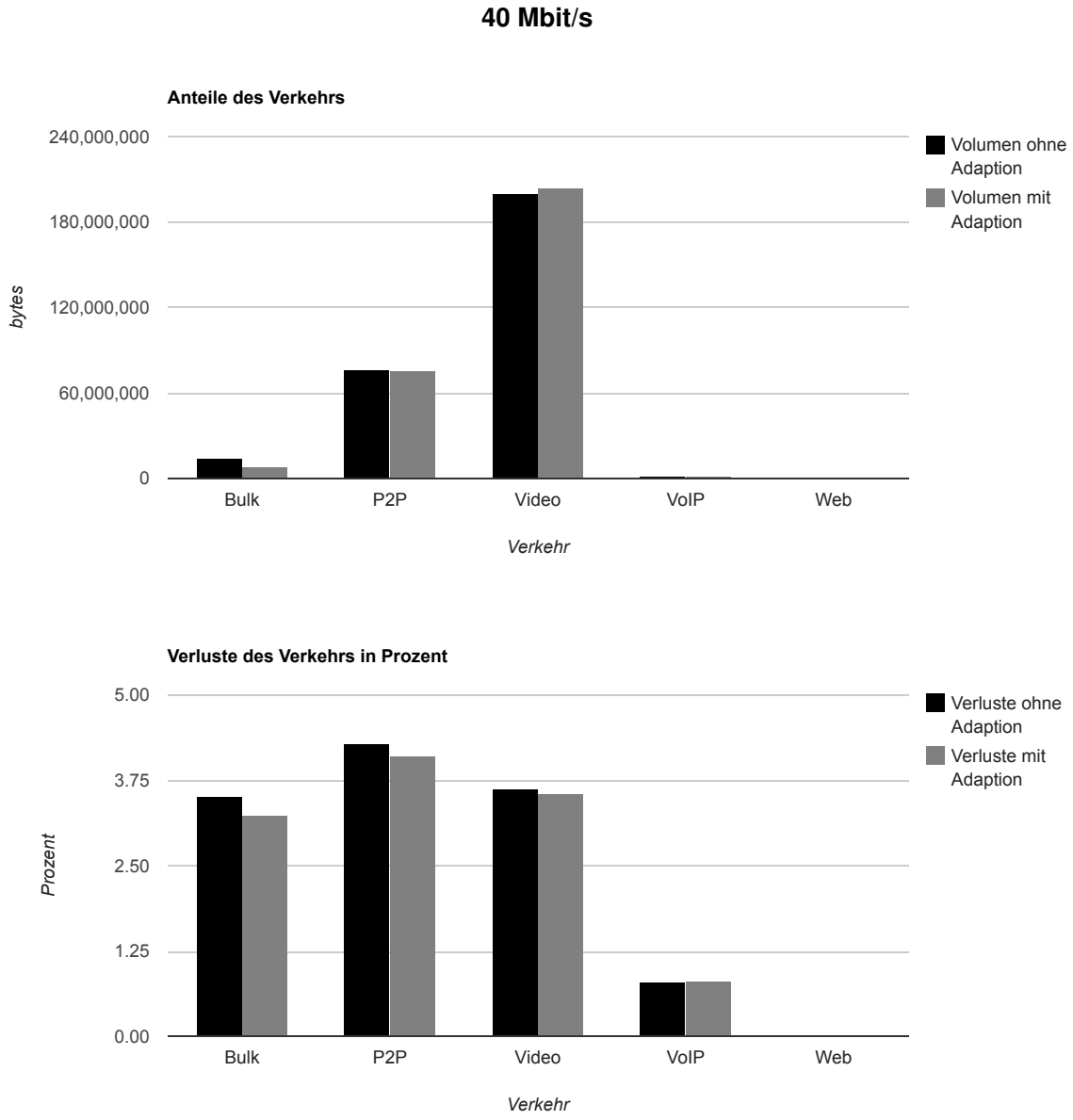
A.0.7. Ergebnisse mit Adaption aller Verkehrsklassen

20 Mbit/s



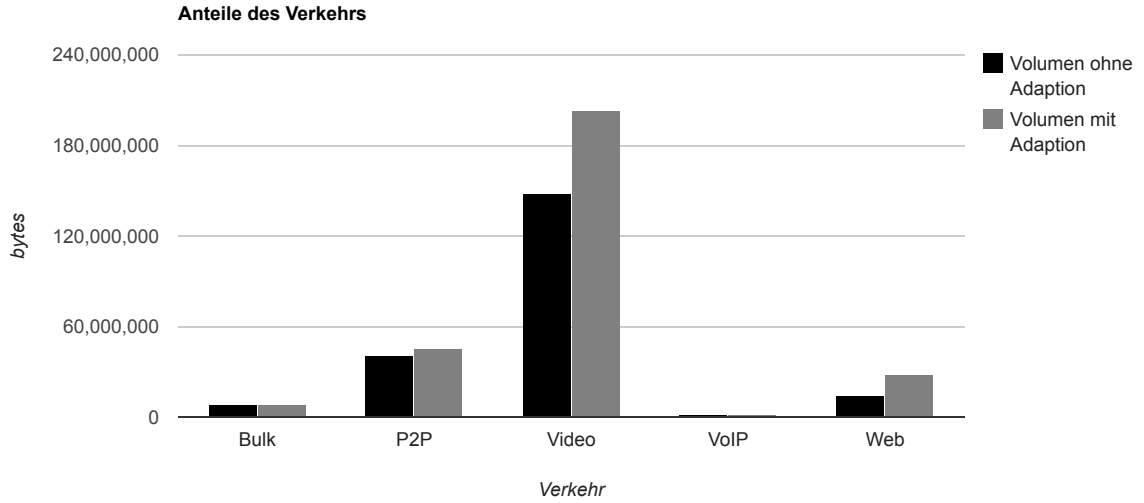


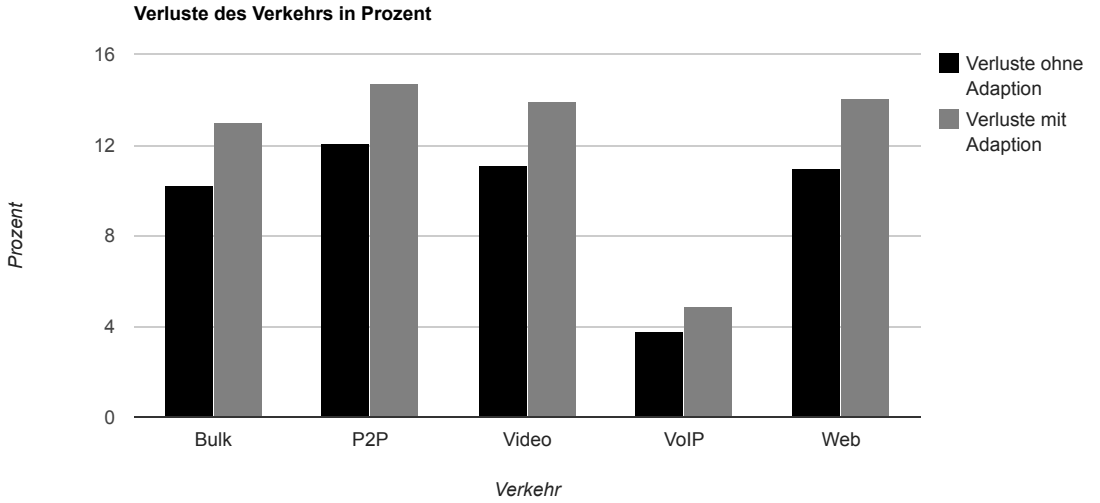
A.0.8. Ergebnisse mit Adaption des Bulkverkehrs



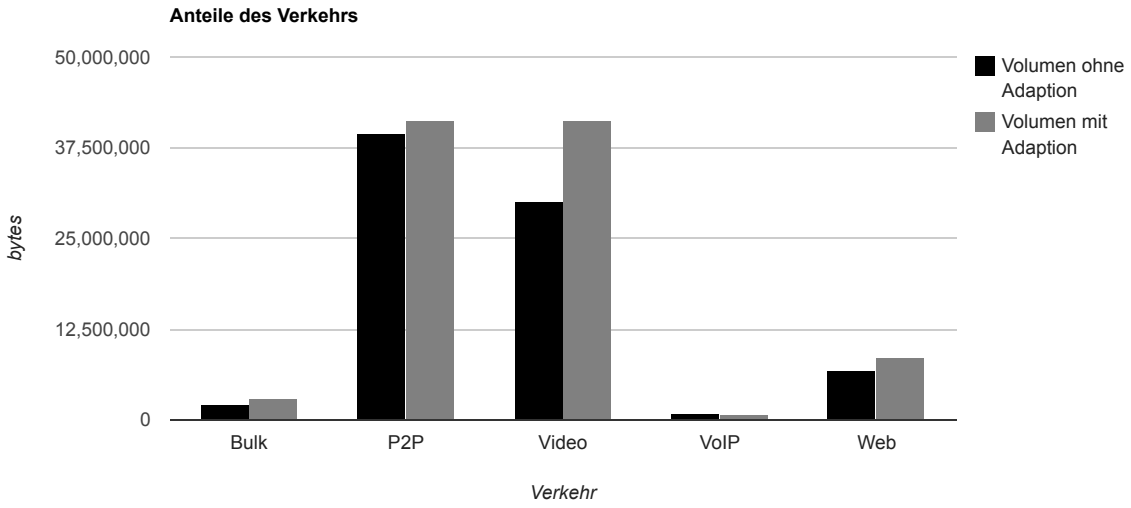
A.0.9. Ergebnisse mit normiertem Verkehr bei 20Mbit/s Bottleneck

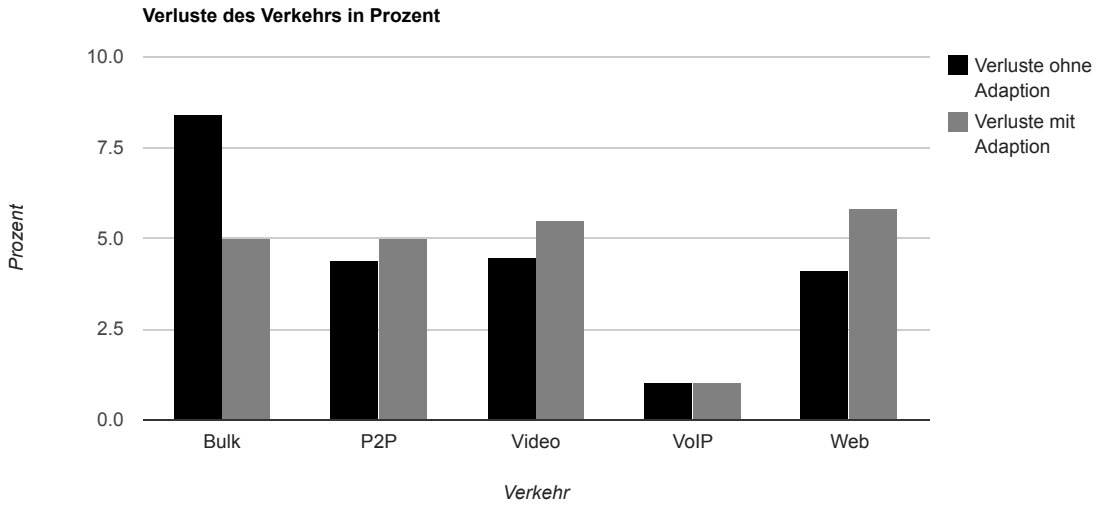
Normierter Bulkverkehr



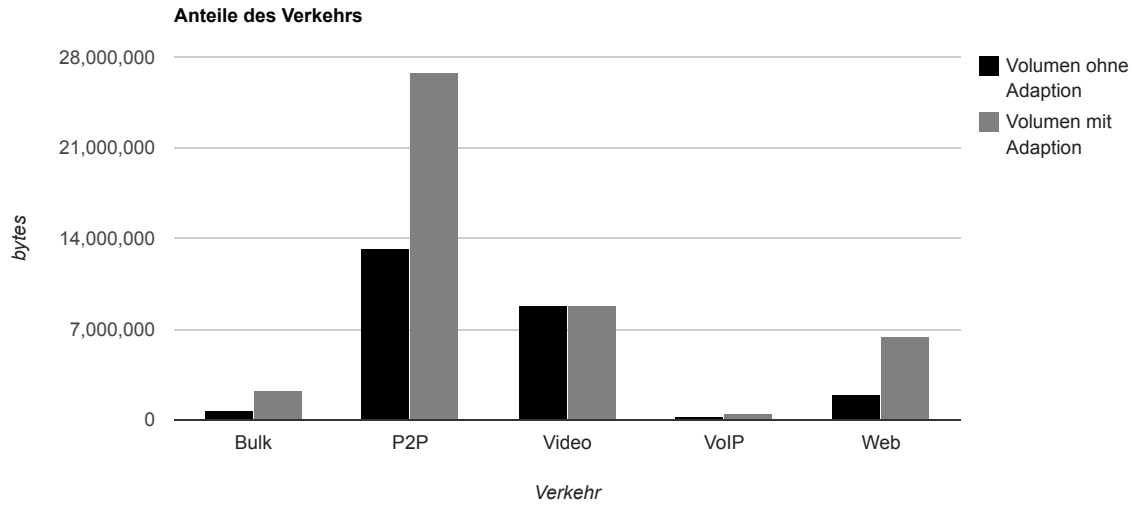


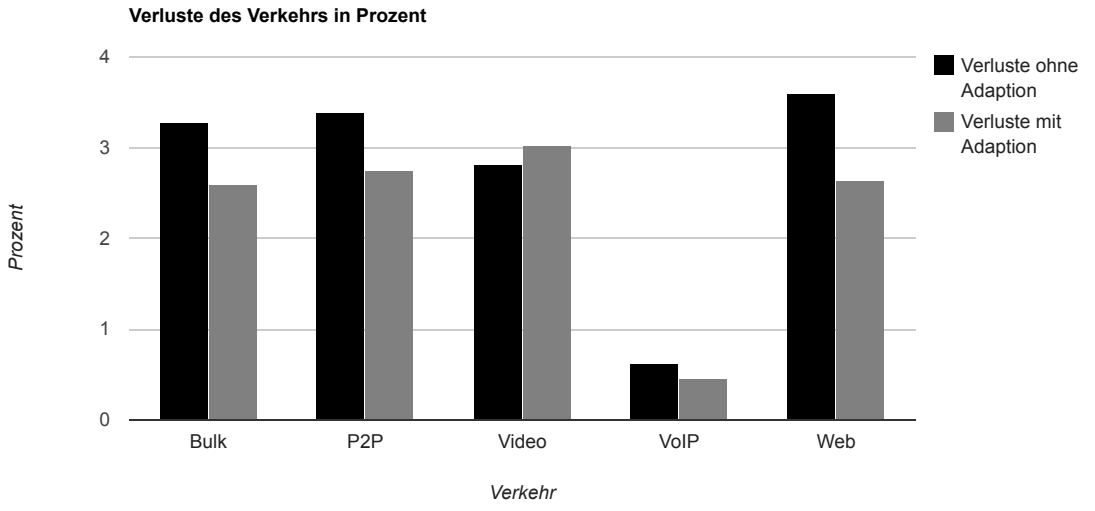
Normierter VoIP Verkehr



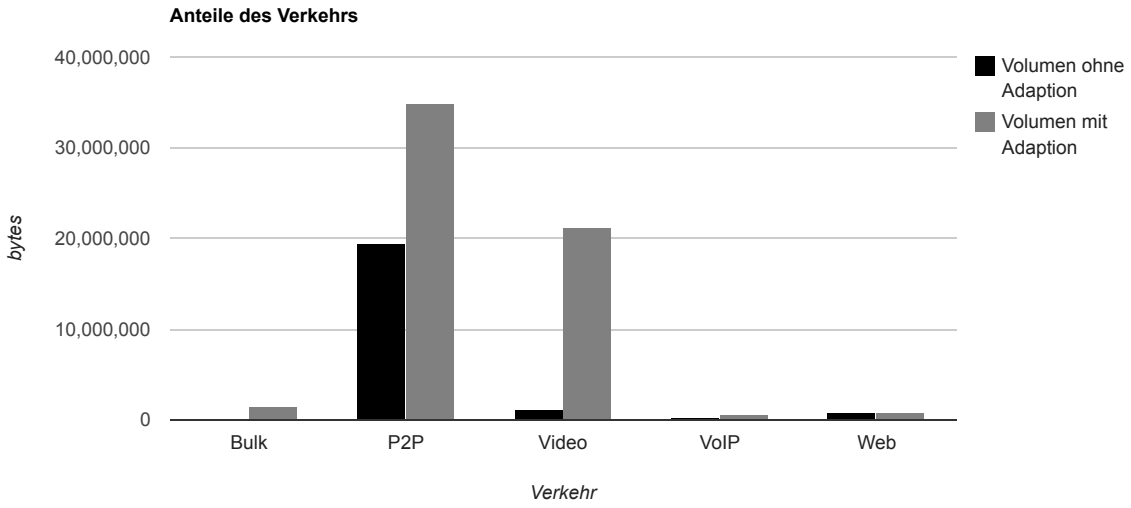


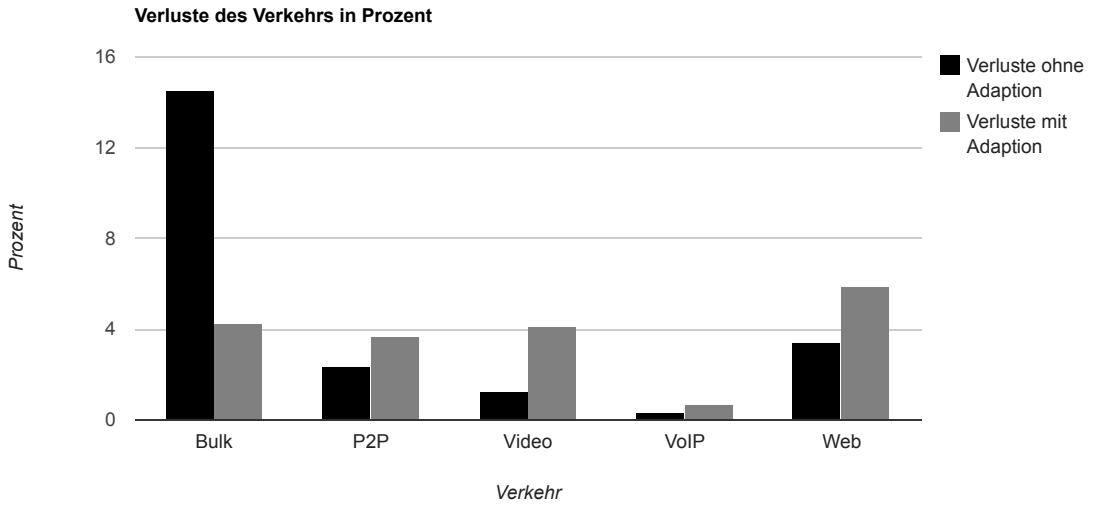
Normierter Videoverkehr





Normierter Webverkehr





Literaturverzeichnis

[AN11] Alcock, Shane ; Nelson, Richard: Application flow control in YouTube video streams. In: *Computer Communication Review* 41 (2011), Nr. 2, 24-30. http://dblp.uni-trier.de/db/journals/ccr/ccr41.html#AlcockN11

[APB09] Allman, M. ; Paxson, V. ; Blanton, E.: *TCP Congestion Control*. RFC 5681 (Draft Standard). http://www.ietf.org/rfc/rfc5681.txt. Version: September 2009 (Request for Comments)

[APS99] Allman, M. ; Paxson, V. ; Stevens, W.: *RFC 2581: TCP Congestion Control*. 1999

[Bie06] Biernacki, Arkadiusz: VoIP Source Model based on the Hyperexponential Distribution. (2006)

[BMM+08] Basher, Naimul ; Mahanti, Aniket ; Mahanti, Anirban ; Williamson, Carey ; Arlitt, Martin F.: A comparative analysis of web and peer-to-peer traffic. In: Huai, Jinpeng (Hrsg.) ; Chen, Robin (Hrsg.) ; Hon, Hsiao-Wuen (Hrsg.) ; Liu, Yunhao (Hrsg.) ; Ma, Wei-Ying (Hrsg.) ; Tomkins, Andrew (Hrsg.) ; Zhang, Xiaodong (Hrsg.): *WWW*, ACM, 2008. – ISBN 978–1–60558–085–2, 287-296

[Cis06] Cisco: Voice Over IP - Per Call Bandwidth Consumption. (2006). http://www.cisco.com/application/pdf/paws/7934/bwidth_consume.pdf

[Cis11] Cisco: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015. (2011), February. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf

[cit88] *ITU-T Recommendation G.711. Pulse Code Modulation (PCM) of Voice Frequencies*. November 1988

[CJ89] Chiu, Dah-Ming ; Jain, Raj: Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. In: *Comput. Netw. ISDN Syst.* 17 (1989), Juni, Nr. 1, 1–14. http://dx.doi.org/10.1016/0169-7552(89)90019-6. – DOI 10.1016/0169–7552(89)90019–6. – ISSN 0169–7552

[CM10a] Cicco, Luca D. ; Mascolo, Saverio: An Experimental Investigation of the Akamai Adaptive Video Streaming. In: *Lecture Notes in Computer Science* 6389 (2010), 447-464. http://dblp.uni-trier.de/db/conf/usab/usab2010.html#CiccoM10

[CM10b] Cicco, Luca D. ; Mascolo, Saverio: A Mathematical Model of the Skype VoIP Congestion Control Algorithm. In: *IEEE Trans. Automat. Contr.* 55 (2010), Nr. 3, 790-795. http://dblp.uni-trier.de/db/journals/tac/tac55.html#CiccoM10

[CMP08] Cicco, Luca D. ; Mascolo, Saverio ; Palmisano, Vittorio: Skype video responsiveness to bandwidth variations. In: Griwodz, Carsten (Hrsg.) ; Wolf, Lars C. (Hrsg.): *NOSSDAV*, ACM, 2008, 81-86

[CMP11] Cicco, Luca D. ; Mascolo, Saverio ; Palmisano, Vittorio: Feedback control for adaptive live video streaming. In: Begen, Ali C. (Hrsg.) ; Mayer-Patel, Ketan (Hrsg.): *MMSys*, ACM, 2011. – ISBN 978–1–4503–0518–1, 145-156

[DC] DE-CIX: *Verkehrsentwicklung der letzten 5 Jahre*. http://www.de-cix.net/about/statistics/, . – Zugriff: 30.09.2012

[EIP10] Erman, David ; Ilie, Dragos ; Popescu, Adrian: Measuring and modeling the BitTorrent content distribution system. In: *Comput. Commun.* 33 (2010), November, S22–S29. http://dx.doi.org/10.1016/j.comcom.2010.04.036. – DOI 10.1016/j.comcom.2010.04.036. – ISSN 0140–3664

[ESSGP08] Erman, David ; Saavedra, Daniel ; Sánchez González, JoséÁ. ; Popescu, Adrian: Validating BitTorrent models. In: *Telecommunication Systems* 39 (2008), 103-116. http://dx.doi.org/10.1007/s11235-008-9115-z. – DOI 10.1007/s11235–008–9115–z. – ISSN 1018–4864

[FGM+99] Fielding, R. ; Gettys, J. ; Mogul, J. ; Frystyk, H. ; Masinter, L. ; Leach, P. ; Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*. United States, 1999

[Floo0] Floyd, S.: *Congestion Control Principles*. RFC 2914 (Best Current Practice). http://www.ietf.org/rfc/rfc2914.txt. Version: September 2000 (Request for Comments)

[GCJM12] Ghobadi, Monia ; Cheng, Yuchung ; Jain, Ankur ; Mathis, Matt: Trickle: Rate Limiting YouTube Video Streaming. In: *Proceedings of the USENIX Annual Technical Conference*, 2012, S. 6

[JJL07] Jeongeun Julie Lee, Intel: A new traffic model for current user web browsing behavior. (2007). http://blogs.intel.com/wp-content/mt-content/com/research/HTTP%20Traffic%20Model_v1%201%20white%20paper.pdf

[KKH10] Kuschnig, Robert ; Kofler, Ingo ; Hellwagner, Hermann: An evaluation of TCP-based rate-control algorithms for adaptive internet streaming of H.264/SVC. In: *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. New York, NY, USA : ACM, 2010 (MMSys '10). – ISBN 978–1–60558–914–5, 157–168

[MKHS10] Mori, Tatsuya ; Kawahara, Ryoichi ; Hasegawa, Haruhisa ; Shimogawa, Shinsuke: Characterizing Traffic Flows Originating from Large-Scale Video Sharing Services. In: Ricciato, Fabio (Hrsg.) ; Mellia, Marco (Hrsg.) ; Biersack, Ernst W. (Hrsg.): *TMA* Bd. 6003, Springer, 2010 (Lecture Notes in Computer Science). – ISBN 978–3–642–12364–1, 17-31

[MSMO97] Mathis, Matthew ; Semke, Jeffrey ; Mahdavi, Jamshid ; Ott, Teunis: *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm*. 1997

[PNK08] Plissonneau, Louis ; Najjary, Taoufik En ; Keller, Guillaume Urvoy: Revisiting web traffic from a DSL provider perspective : the case of YouTube. In: *ITC-SS 2008, 19th ITC Specialist Seminar 2008 on Network Usage and Traffic, October 8-9, 2008, Berlin, Germany*. Berlin, GERMANY, 10 2008

[RFB01] Ramakrishnan, K. ; Floyd, S. ; Black, D.: *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168 (Proposed Standard). `http://www.ietf.org/rfc/rfc3168.txt`. Version: September 2001 (Request for Comments)

[RTV10] Rossi, Dario ; Testa, Claudio ; Valenti, Silvio: Yes, We LEDBAT: Playing with the New BitTorrent Congestion Control Algorithm. In: Krishnamurthy, Arvind (Hrsg.) ; Plattner, Bernhard (Hrsg.): *PAM* Bd. 6032, Springer, 2010 (Lecture Notes in Computer Science). – ISBN 978–3–642–12333–7, 31-40

[SHIK11] Shalunov, S. ; Hazel, G. ; Iyengar, M. ; Kühlewind, M.: Low Extra Delay Background Transport (LEDBAT): draft-ietf-ledbat-congestion-04.txt / Submitted as Internet-Draft to the Internet Engineering Task Force (IETF). 2011. – Forschungsbericht

Erklärung

Mir ist bekannt und ich erkenne an, dass ich an den Ergebnissen meiner Diplomarbeit keine eigenständigen Verwertungsrechte habe. Eine Verwertung der Arbeit einschließlich der Ausarbeitung darf nur nach Zustimmung durch das Institut für Kommunikationsnetze und Rechnersysteme erfolgen.
Ich erkläre weiterhin, dass ich diese Arbeit selbständig verfasst und keine anderen als die in meiner Ausarbeitung angegebenen Hilfsmittel für die Durchführung der Arbeit verwendet habe.

(Dirk Mast)