Adaptives Bandbreitenmanagement in hybriden Peer-unterstützten Video-on-Demand Systemen

Masterarbeit

Markus Günther, B.Sc.
KOM-D-0407

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Adaptives Bandbreitenmanagement in hybriden Peer-unterstützten Video-on-Demand Systemen
Masterarbeit
KOM-D-0407

Eingereicht von Markus Günther, B.Sc.
Tag der Einreichung: 9. Dezember 2010

Gutachter: Prof. Dr.-Ing. Ralf Steinmetz
Betreuer: Dipl.-Inf. Konstantin Pussep

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 9. Dezember 2010 Markus Günther, B.Sc.

Zusammenfassung

Das Betrachten von Video-Inhalten über Video-on-Demand Dienste erfreut sich einer wachsenden Beliebtheit unter den Benutzern. Jedoch erzeugen insbesondere Videos in hoher Qualität (SD- bis HD-Standard) eine hohe Last auf den Servern von Content Providern. Eine Peer-to-Peer basierte Verteilstrategie kann hier Abhilfe schaffen, führt jedoch ein enormes Kostenproblem für einen ISP durch erhöhtes Verkehrsvolumen auf Inter-ISP-Verbindungen ein. Existierende Mechanismen aus den Bereichen des Traffic Shaping, Hardware Provisioning und Locality Aware Peer Selection adressieren die Reduktion dieses kostenintensiven Datenverkehrs. Dies schränkt jedoch potentiell die Verfügbarkeit von Daten ein und beeinträchtigt damit die Performanz eines solchen Dienstes, was weder im Interesse des Content Providers, noch des Benutzers liegt.

Die vorliegende Arbeit versucht, durch adaptives Bandbreitenmanagement die Interessen aller Beteiligten zu wahren. Die Basis hierfür stellt eine hybride Verteilstrategie: Der primäre Distributionspfad ist durch ein P2P-System realisiert, während ein sekundärer Distributionspfad über adaptive Server im Falle eines Performanz-Verlusts als Ausweichlösung dient. Die adaptiven Server basieren auf der sog. Supporter-Strategie.

Der Beitrag der Arbeit ist zweierlei. Zunächst erfolgt die prototypische Implementierung und Evaluation der Supporter-Strategie. Die Ergebnisse dieser Evaluation zeigen, dass die Supporter-Strategie eine Kompromisslösung zwischen Server-Last und Leistung erzielt. Sie ist insbesondere für Content Provider interessant, die den Peer-Bedarf a priori nicht kennen oder deren Server-Auslastung über die Zeit starken Schwankungen unterzogen ist. Diese Resultate greift die Arbeit auf, um den Einsatz der Supporter-Strategie innerhalb der hybriden Systemarchitektur zu rechtfertigen. Darauf aufbauend erfolgt die Erarbeitung eines Konzepts zum adaptiven Bandbreitenmanagement. Die Idee hinter dem Mechanismus ist, das Zugangsprofil ausgewählter Peers zu erhöhen. Die Ergebnisse der Arbeit zeigen, dass sich durch eine geschickte Auswahl von sich besonders günstig verhaltenden Peers eine deutliche Besserung des Systemzustands erreichen lässt. Dies wirkt sich nicht nur positiv auf das Inter-ISP-Datenvolumen aus, sondern resultiert auch in einer Reduktion der Last auf Servern des Content Providers. Ein Benutzer erfährt durch Einsatz des Ansatzes – je nach Konfiguration und Szenario – eine gleichbleibende, wenn nicht sogar bessere Dienstgüte in Form reduzierter Abspielverzögerungen. Die Arbeit stellt somit eine ganzheitliche Lösung für Peer-to-Peer basiertes Video-on-Demand in einer hybriden Systemarchitektur vor, in der alle beteiligten Interessengruppen einen Gewinn erzielen.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Problembeschreibung . . . 2
- 1.2 Forschungsfrage und wissenschaftlicher Beitrag . . . 2
- 1.3 Struktur der Arbeit . . . 3

2 Grundlagen **5**
- 2.1 Verteilung von Daten im Internet . . . 5
  - 2.1.1 Content Delivery Netzwerke . . . 5
  - 2.1.2 Peer-to-Peer Netzwerke . . . 6
- 2.2 BitTorrent . . . 7
  - 2.2.1 Funktionsweise . . . 7
  - 2.2.2 Verwaltung der Nachbarschaftsliste . . . 9
  - 2.2.3 Arbeitsweise des Choke-Algorithmus . . . 9
- 2.3 Peer-to-Peer gestütztes Video Streaming . . . 10
  - 2.3.1 Baum-basierte Topologie . . . 11
  - 2.3.2 Mesh-basierte Topologie . . . 13
  - 2.3.3 Give-to-Get . . . 14
- 2.4 Adaptive Server-Allokationsstrategien . . . 17
  - 2.4.1 Supporter-Strategie . . . 17
  - 2.4.2 Funktionsweise der Supporter-Strategie . . . 18
- 2.5 Kombination von Give-to-Get und adaptiver Server-Allokation . . . 19

3 Verwandte Arbeiten **21**
- 3.1 Traffic Shaping . . . 21
- 3.2 Bereitstellung dedizierter Hardware . . . 22
  - 3.2.1 Cache-gestützte Lösungen . . . 22
  - 3.2.2 Gateway Peers . . . 23
- 3.3 Locality Aware Peer Selection . . . 24
  - 3.3.1 Informationsserver . . . 25
  - 3.3.2 Client-basierte Ansätze . . . 26
- 3.4 Diskussion . . . 26

4 Adaptives Bandbreitenmanagement **27**
- 4.1 Ansatz . . . 28
  - 4.1.1 Interaktion der Komponenten . . . 28
  - 4.1.2 Beeinflussung des Zugangsprofils und einhergehende Herausforderungen . . . 29
- 4.2 Datenerhebung . . . 30
  - 4.2.1 Informationsquelle . . . 30
  - 4.2.2 Statistiken über den P2P-Datenstrom eines Benutzers . . . 31
- 4.3 Inferenz . . . 31
  - 4.3.1 Identifikationsmetriken . . . 31
  - 4.3.2 Berechnung des Gütewerts . . . 34
  - 4.3.3 Selektion der besten Peers . . . 34
- 4.4 Bandbreitenzuteilung . . . 34
- 4.5 Realisierung auf Seiten des ISP . . . 35
- 4.6 Einschränkungen im Rahmen der Arbeit . . . 35

5 Implementierung **37**
- 5.1 Implementierung des adaptiven Bandbreitenmanagements . . . 37
  - 5.1.1 Modulübersicht . . . 37

5.1.2 Ablauf einer Bewertungsrunde . . . 38
5.2 Implementierung der Supporter Strategie . . . 39
5.2.1 Kommunikation . . . 39
5.2.2 Erweiterung des NextShare Client um die Supporter-Strategie . . . 40
5.3 Implementierung eines Video-on-Demand Client-Emulators . . . 45
5.3.1 Notwendige Erweiterungen der NextShare Code-Basis . . . 45
5.3.2 Zusammenspiel mit der Überwachungskomponente der Supporter-Strategie . . . 47

6 Methodik der Evaluation **49**
6.1 Evaluationsplattform . . . 49
6.2 Topologie und Systemarchitektur . . . 49
6.3 Testdurchführung . . . 51
6.4 Parameter . . . 51
6.4.1 Szenarien-bezogene Parameter . . . 51
6.4.2 Schwarm-bezogene Parameter . . . 52
6.4.3 Client-bezogene Parameter . . . 53
6.4.4 Supporter-bezogene Parameter . . . 54
6.4.5 Bandbreitenmanagement-bezogene Parameter . . . 54
6.5 Ausgabemetriken . . . 55
6.6 Verlässlichkeit und Generalisierbarkeit der Ergebnisse . . . 56

7 Evaluation der Supporter-Strategie **57**
7.1 Einfluss der Ankunftsrate auf Ausgabemetriken . . . 58
7.1.1 Maximale Anzahl versorgbarer Peers . . . 58
7.1.2 Einfluss auf die Stalling-Verzögerung . . . 59
7.1.3 Einfluss auf die Startup-Verzögerung . . . 60
7.1.4 Einfluss auf die Server- und Peer-Last . . . 62
7.1.5 Einfluss auf den Signalisierungs-Overhead . . . 63
7.1.6 Diskussion . . . 64
7.2 Entwicklung von Zustandsallokationen an der Überwachungskomponente . . . 64
7.2.1 Vergleich der Strategien bei einer Ankunftsrate von 12 Peers/Min. . . . 65
7.2.2 Vergleich der Strategien bei einer Ankunftsrate von 24 Peers/Min. . . . 66
7.2.3 Diskussion . . . 67
7.3 Basisvergleich . . . 68
7.3.1 Vergleich bei einer Ankunftsrate von 12 Peers/Min. . . . 68
7.3.2 Vergleich bei einer Ankunftsrate von 24 Peers/Min. . . . 68
7.3.3 Diskussion . . . 69
7.4 Vergleich mit *Adaptive Server Allocation for Peer-Assisted Video-on-Demand (Pussep et al.)* . . . 70

8 Evaluation des adaptiven Bandbreitenmanagements **73**
8.1 Güte der Identifikationsmetriken . . . 73
8.1.1 Untersuchung der Auslastung der Upload-Kapazität $U(t)$ . . . 74
8.1.2 Untersuchung des Verhältnisses zwischen Upload und Download $S(t)$ . . . 76
8.1.3 Untersuchung der Lokalitätsmetrik $L(t)$ und $L'(t)$ . . . 77
8.1.4 Kombinationsmöglichkeiten . . . 79
8.1.5 Diskussion . . . 81
8.2 Genereller Einfluss des Bandbreitenmanagement . . . 81
8.2.1 Einfluss auf Abspielverzögerungen . . . 82
8.2.2 Einfluss auf die Server-Last . . . 84
8.2.3 Einfluss auf das Transfervolumen nach Verkehrsklasse . . . 84
8.2.4 Diskussion . . . 85
8.3 Einfluss des Bandbreitenmanagements im Falle eines Early Adopters . . . 86
8.3.1 Vergleich der Abspielverzögerungen . . . 86
8.3.2 Einfluss auf die Server-Last . . . 87
8.3.3 Vergleich des Transfervolumens nach Verkehrsklasse . . . 87
8.3.4 Diskussion . . . 88

9 Diskussion **89**
9.1 Zusammenfassung . . . 89
9.2 Zukünftige Arbeiten . . . 90
9.2.1 Praxistauglichkeit der Implementierung der Supporter-Strategie . . . 90
9.2.2 Adaptives Bandbreitenmanagement . . . 91

A Studie zur Anwendbarkeit von Linux Traffic Control **93**
A.1 Linux Traffic Control . . . 93
A.2 Traffic Control Next Generation . . . 94
A.3 Evaluationsszenarien . . . 94
A.3.1 Ratenlimitierung per autonomen System . . . 95
A.3.2 Ratenlimitierung per Verbindung zwischen zwei Endsystemen . . . 97
A.3.3 Ratenlimitierung per Applikation . . . 98
A.3.4 Aufteilung von Intra- und Inter-ISP-Datenverkehr mit htb . . . 100
A.4 Probleme mit `tc` in einem G-Lab Testbed . . . 102
A.5 Diskussion . . . 103

B Limitierende Faktoren bei der Datenerhebung **105**

C UML-Diagramme **107**

D Client-seitige Statistiken **109**

Literaturverzeichnis **115**

Abbildungsverzeichnis

2.1 Darstellung der Beziehungen zwischen CDN- und P2P-Overlays und der Netzschicht 6
2.2 Temporale Abfolge der Aktionen in einem typischen BitTorrent-Szenario 8
2.3 Qualitative Darstellung einer baum-basierten Topologie 11
2.4 Qualitative Darstellung einer mesh-basierten Topologie 13
2.5 Kategorisierung von Chunks nach unterschiedlichen Prioritäten bei G2G 14
2.6 Zustandsdiagramm der Peer-Zustände auf Seiten des Überwachungsmechanismus 18
2.7 Darstellung der hybriden Systemarchitektur eines Peer-to-Peer gestützten Video-on-Demand Systems 19

4.1 Interaktion der Komponenten des adaptiven Bandbreitenmanagements 28
4.2 G2G-Peers holen Feedback von den Nachbarn ihrer Nachbarn ein 29
4.3 Verlauf der Kurven nach Lokalitätsmetrik $L(t)$ und $L'(t)$ 33
4.4 Vereinfachte Darstellung einer beispielhaften ISP-Plattform mit Zugangsbereich 36

5.1 Klassendiagramm des adaptiven Bandbreitenmanagements (Grobansicht) 37
5.2 Ablauf einer Bewertungsrunde mit anschließender Zuteilung der neuen Zugangsprofile 38
5.3 Klassendiagramm des Moduls `SupporterMonitor` (Grobansicht) 41
5.4 Visualisierung der internen Abläufe bei der Verarbeitung client-seitiger Nachrichten 42
5.5 Zustellung aktualisierter Peer-Listen an korrespondierende Supporter 43
5.6 Klassendiagramm des Moduls `Supporter` (Detailansicht) 44
5.7 Klassendiagramm des Moduls `ClientEmulation.CallbackHandlers` 45

6.1 Vereinfachte Darstellung der virtuellen Topologie 50

7.1 Einfluss der Ankunftsrate auf die Stalling-Verzögerung, betrachtet über alle Server-Strategien 60
7.2 Einfluss der Ankunftsrate auf die Startup-Verzögerung, betrachtet über alle Server-Strategien 61
7.3 Einfluss der Ankunftsrate auf die Startup-Verzögerung (nur 95%-Perzentil) 62
7.4 Einfluss der Ankunftsrate auf die Server- und Peer-Last 63
7.5 Einfluss der Ankunftsrate auf das Volumen Supporter-bezogener Kommunikation 63
7.6 Entwicklung der Zustandsallokationen bei einer Ankunftsrate von 12 Peers/Min. 65
7.7 Entwicklung der Zustandsallokationen bei einer Ankunftsrate von 24 Peers/Min. 67
7.8 Vergleich der Strategien untereinander bei einer Ankunftsrate von 12 Peers/Min. 69
7.9 Vergleich der Strategien untereinander bei einer Ankunftsrate von 24 Peers/Min. 69
7.10 Vergleich des Supporters (Testbed) mit den Ergebnissen aus [10] (Simulation) 71

8.1 Analyse der Identifikationsmetrik Auslastung der Upload-Kapazität $U(t)$ 75
8.2 Analyse der Identifikationsmetrik Upload-Download-Verhältnis $S(t)$ 76
8.3 Vergleich der Peer-Bewertungen anhand der Lokalitätsmetrik $L(t)$ und $L'(t)$ 78
8.4 Analyse der Identifikationsgüte der Lokalitätsmetrik $L(t)$ und $L'(t)$ 79
8.5 Verteilung von Gütewerten in Abhängigkeit div. Identifikationsmetriken 80
8.6 Einfluss der Reihenfolge bei Kombination mehrerer Identifikationsmetriken 80
8.7 CDF zu den Stalling-Verzögerungen der Peers und ihren Startup-Verzögerungen 83
8.8 Abspielverzögerungen auf dem Median und dem 95%-Perzentil im Early Adopter Szenario 87

A.1 Funktionsweise von Linux Traffic Control 93
A.2 Ratenlimitierung per autonomen System mit stochastischer Fairness 96
A.3 Ratenlimitierung per autonomen System ohne stochastische Fairness 97
A.4 Ratenlimitierung per Verbindung zwischen zwei Endsystemen 98
A.5 Ratenlimitierung per Applikation 99
A.6 Aufteilung von Intra- und Inter-ISP-Datenverkehr mit htb und Token Borrowing 101
A.7 Zuweisung eines anonymen Ports bei Verbindungsherstellung 103

C.1 Klassendiagramm des adaptiven Bandbreitenmanagements (Detailansicht) 107

C.2 Klassendiagramm des Moduls `SupporterMonitor` (Detailansicht) 108

Tabellenverzeichnis

5.1 Übersicht über die HTTP-Ressourcen der Tracker-Protokollerweiterung 40

6.1 Tabellarische Darstellung verwendeter Video-Testdateien 53

7.1 Grundkonfiguration für die Testszenarien der Supporter-Strategie 57
7.2 Abweichungen zur Grundkonfiguration des Supporter-Szenarios 58
7.3 Auswertung der maximalen Anzahl versogbarer Peers in Abhängigkeit der Ankunftsrate 58
7.4 Abweichungen zur Grundkonfiguration des Supporter-Szenarios 64
7.5 Abweichungen zur Grundkonfiguration des Supporter-Szenarios 68
7.6 Direkter Vergleich der Server-Strategien bei einer Ankunftsrate von 12 Peers/Min. 68
7.7 Direkter Vergleich der Server-Strategien bei einer Ankunftsrate von 24 Peers/Min. 70
7.8 Reduktionsfaktoren bzgl. Stalling der statischen Server im Vergleich mit der Supporter-Strategie . . . 70
7.9 Berechnung der gewichteten mittleren Größe einer Nachricht der Supporter-Implementierung 72

8.1 Grundkonfiguration der Testszenarien zu den Identifikationsmetriken 74
8.2 Anteil individueller Gruppen an der Gesamtzahl mit durchschnittlichem Gütewert 76
8.3 Anteil fehlerhaft identifizierter Peers bei Selektion von 33% und 69% nach Verweilzeit 77
8.4 Abweichungen zur Grundkonfiguration des Identifikationsmetriken-Szenarios 77
8.5 Grundkonfiguration der Testszenarien zum adaptiven Bandbreitenmanagement 82
8.6 Vergleich der absoluten Abspielverzögerungen für jede Konfiguration 83
8.7 Absolute Server-Last und prozentuale Einsparung bzgl. Server-Last im Vgl. zu Basisfall A 84
8.8 Absolutes Datenvolumen und prozentuale Einsparung bzgl. des Datenvolumens im Vgl. zu Basisfall A 85
8.9 Abweichungen zur Grundkonfiguration des Early Adopter Szenarios 86
8.10 Absolute Server-Last und prozentuale Einsparung bzgl. der Server-Last im Vgl. zu Basisfall A und BM 87
8.11 Absolutes Datenvolumen und prozentualer Vergleich über alle ISPs 88

D.1 Übersicht über allgemeine Statistiken, die client-seitig aggregiert werden 109
D.2 Übersicht über Statistiken bzgl. der Wiedergabe des Videos 109
D.3 Übersicht über Statistiken bzgl. bereits heruntergeladener oder fehlender Datenblöcke 109
D.4 Übersicht über client-seitige Statistiken bezüglich der aktuellen aktiven Nachbarschaft des Peers . . . 110

1 Einleitung

Das Betrachten von Videoinhalten über das Internet hat in jüngster Zeit einen deutlichen Popularitätsgewinn erfahren. So stellt bspw. Cisco [1] fest, dass zum zweiten Quartal des Jahres 2010 der Datenverkehr für vom Endbenutzer angeforderte Videoinhalte bei ca. 33% gemessen an dessen Gesamtdatenverkehr liegt. Cisco wagt in derselben Studie die Prognose, dass dieses Verhältnis zunächst auf 40% (Ende 2010) und langfristig sogar auf ca. 90% (2014) ansteigen wird. Dieser Trend spiegelt sich auch in den steigenden Benutzerzahlen prominenter Videoanbieter, wie bspw. YouTube, Amazon Video-on-Demand und T-Home wieder. Dieses rasante Wachstum schlägt sich ebenfalls in dem Datenvolumen, welches durch Video-Streaming-Anbieter generiert wird, nieder. Die wachsende Anzahl an Benutzer ist allerdings nicht der einzige, beisteuernde Faktor. Video-Qualitätsstandards befinden sich derzeit in einer Phase des Umbruchs, wodurch die individuellen Ansprüche der Benutzer zusammen mit den technischen Möglichkeiten wachsen. Für Benutzer gehört es inzwischen zum Standard, Videos in hoher Qualität – angefangen von 480p (Standard Definition) bis hin zu 1080p (High Definition) – betrachten zu können. Dabei soll die Erlebnisgüte beim Betrachten eines Videos über das Internet der Erfahrung, die der Benutzer mit herkömmlichen TV-Empfängern hat, in nichts nachstehen. Die Erfüllung dieser Anforderungen stellt bereits für Videos in Standard Definition eine technische und infrastrukturelle Herausforderung dar, da für die unterbrechungsfreie Wiedergabe ein konstanter Datenstrom von ca. 1,5 Mb/s[1] erforderlich ist [2].

Die wachsenden Anforderungen stellen beteiligte Interessengruppen vor eine Reihe von ernstzunehmenden Problemen. Content Provider (dt. Anbieter von Inhalten) sind in erster Linie bestrebt, angebotene Inhalte möglichst schnell und unkompliziert an interessierte Nutzer zu senden. Um dies zu realisieren, nutzen einige Content Provider sogenannte Content Delivery Networks (CDN, dt. Distributoren für digitale Inhalte), welche die nötige Server-Hardware und Infrastruktur bereitstellen, um eine Vielzahl von Benutzern mit den entsprechenden Daten versorgen zu können. Der Distributionsweg über ein Content Delivery Network ist gerade für junge Content Provider interessant, die nicht über die notwendigen finanziellen Mittel verfügen, um für die erforderliche Infrastruktur selbst zu sorgen. Aber auch etablierte Anbieter, wie bspw. Google, nutzen den Dienst von Content Delivery Networks, um sich auf ihre Kernkompetenzen beschränken zu können [3].

Content Delivery Networks, deren Infrastruktur typischerweise auf dem klassischen Client-/Server-Modell basiert, sehen sich durch das erhöhte Datenaufkommen mit dem Problem einer steigenden Server-Last konfrontiert. Client-/Server-Architekturen skalieren zwar prinzipiell sehr gut mit den wachsenden Anforderungen, allerdings ist diese Skalierfähigkeit nur durch zusätzliche Server-Hardware sicherzustellen. Dies sorgt für ein gewisses Maß an finanziellem Druck, da Content Delivery Networks durchaus daran interessiert sind, die Ansprüche ihrer Kunden, den Content Providern, zufrieden zu stellen.

Mit dem gleichen Problem sieht sich allerdings auch der Internet Service Provider (ISP, dt. Netzanbieter) des Endbenutzers konfrontiert. Durch dessen Infrastruktur werden die Daten, die der Konsument angefordert hat, an das Endgerät geliefert. Ungleich dem Content Provider bzw. dem Content Delivery Network hat der ISP jedoch keinerlei Anreiz, in weitere, kostenintensive Hardware zu investieren, da er aus der Weiterleitung Bandbreiten-lastiger Videoinhalte keine direkten finanziellen Vorteile erzielt. Aus diesem Grund hat man sich jüngst mit der Frage beschäftigt, in welcher Art und Weise eine geschäftliche Kollaboration zwischen Content Provider und Internet Service Provider aussehen könnte [2]. Mögliche Geschäftsmodelle sind jedoch als kritisch zu betrachten. Der ISP hätte durch ein entsprechendes Geschäftsmodell sicherlich den Anreiz, seine Infrastruktur hinreichend auszubauen und besäße über die Einnahmen auch die monetäre Kraft, um in den technologischen Ausbau zu investieren. Kritisch ist allerdings, dass durch ein Bezahlmodell für anfallenden Datenverkehr diejenigen Endbenutzer potentiell benachteiligt werden, die für einen speziellen Dienst nicht bezahlen. Man fürchtet daher eine Verletzung der Netz-Neutralität [4], da nicht länger gewährleistet ist, dass *jeder* Benutzer an die Daten gelangen kann, die ihm innerhalb eines *offenen* Internet zugänglich sein sollen.

In Ermangelung eines Geschäftsmodells limitieren einige ISPs ungewollten Datenverkehr, um dadurch entstehende Kosten einzusparen [2]. Eine derartige Regulation des Datenverkehrs steht ebenfalls im Konflikt mit dem Prinzip der Netz-Neutralität und liegt weder im Sinne von Content Providern, noch von Endbenutzern. Content Provider

[1] Die konkreten Anforderungen schwanken in Abhängigkeit des Kodierverfahrens und der gewünschten Qualität.

befürchten, dass derartige Methoden speziell zu Stoßzeiten verhindern, dass ihre Konsumenten die gewünschten Inhalte abrufen können. Ein Problem, wofür der Endbenutzer aufgrund mangelnder Transparenz bezüglich der Distributionswege zunächst den Content Provider in die Verantwortung zieht. Es liegt allerdings auch nicht im Sinne der Endbenutzer, denn diese erwarten für den Dienst, den sie schließlich bezahlen, eine verlässliche und unterbrechungsfreie Wiedergabe von Inhalten, so sie diese anfordern.

1.1 Problembeschreibung

Eine Lösung dieses Problems kann nur entlang mehrerer Fronten erarbeitet werden, da die verschiedenen beteiligten Gruppen an der Optimierung teilweise konträr zueinander stehenden Zielen interessiert sind. Für Content Provider bzw. CDNs erscheint der Wechsel zu einem verteilten Ansatz mittels *Peer-to-Peer Computing* von Bedeutung. Ein derartiger Ansatz stellt eine Kompromisslösung zwischen Kosten und Nutzen dar und genießt zudem eine höhere Robustheit gegenüber Ausfällen in der Netzwerkarchitektur. Die Entwicklung im Bereich des Peer-to-Peer Computing hat durchaus gezeigt, dass dieses Kommunikationsmittel ein probates Mittel ist, um kostengünstig skalierungsfähige verteilte Systeme zu erschaffen. In einem Peer-to-Peer System (P2P-System) wird die Verantwortung, die im traditionellen Client-/Server-Modell der Server übernimmt, auf die einzelnen Benutzer des Systems verteilt [5]. Ein sogenannter Peer agiert also nicht nur als Konsument, sondern gleichzeitig auch als Anbieter. Um Daten innerhalb eines P2P-Systems zwischen den einzelnen Benutzern zu tauschen, genügt es, Kommunikationspfade zu Peers aufzubauen, die den gewünschten Inhalt besitzen und anbieten können. Potentiell hat also jeder Peer Zugriff auf einen verteilten Datenbestand, der im Grunde genommen nur durch die Anzahl der Benutzer des Systems und der Speicherkapazität ihrer Hardware beschränkt ist.

Eine weitere entscheidende Eigenschaft P2P-basierter Systeme leitet sich ebenfalls aus der Tatsache ab, dass ein Peer gleichzeitig als Anbieter und Konsument agiert: Das System skaliert inhärent mit der Anzahl seiner Benutzer. Gerade dieser Umstand lässt P2P-Systeme für Content Provider äußerst attraktiv erscheinen. Ein Anbieter hat die Möglichkeit, durch eine P2P-basierte Lösung seine Inhalte kostengünstig anzubieten, da die Notwendigkeit für eine umfangreiche Server-Landschaft, die für erhebliche Kosten sowohl in der Anschaffung als auch der Unterhaltung sorgen würde, nicht länger gegeben ist.

Für Content Provider und Content Delivery Networks scheinen P2P-Systeme durchaus eine Gewinnsituation herbeizuführen. Bei genauerer Betrachtung stellt man allerdings fest, dass das ursprüngliche Kostenproblem nicht eliminiert, sondern lediglich an eine andere Instanz verlagert wird: Den Internet Service Provider [6]. Die Verlagerung der Kosten erklärt sich durch die Art und Weise, wie gängige P2P-Systeme arbeiten. Um eine besonders hohe Robustheit und gute Effizienz bei der Datenübertragung zu haben, werden Peering-Beziehungen zufällig gewählt. Ein Peer bevorzugt bei klassischen P2P-Systemen hierbei solche Verbindungen, welche die Auslastung seiner Bandbreite maximieren. Das Resultat dessen ist, dass Peers in vielen Fällen Verbindungen zu anderen Teilnehmern aufrecht erhalten, die sich in anderen ISP-Domänen[2] befinden. Diese sogenannten Inter-ISP-Verbindungen stellen einen erheblichen Kostenfaktor für den ISP dar. Das Problem verstärkt sich, denn der ISP hat keinen Anreiz, diese Verbindungen überhaupt zuzulassen.

Dieses Problem kann auf Seiten des P2P-Systems gelöst werden, wenn lokalitätsfördernde Mechanismen wie Biased Neighbour Selection [7] oder Biased Unchoking [8] zum Einsatz kommen. Durch diese Techniken fördert man Peering-Beziehungen, die sich vornehmlich im lokalen ISP-Netzwerk befinden. Derartige Mechanismen führen allerdings zu weiteren Problemen, da sie das grundsätzliche Problem nur im Kontext des Netzanbieters betrachten.

1.2 Forschungsfrage und wissenschaftlicher Beitrag

Das Ziel der vorliegenden Arbeit ist es, Mechanismen in einer hybriden Systemarchitektur zu erproben, von der letzten Endes alle beteiligten Interessengruppen profitieren. Der primäre Distributionsweg basiert auf einer P2P-basierten Verteilstrategie, in der lokalitätsfördernde Mechanismen eingesetzt werden. Dies sorgt für eine potentielle

[2] Das Internet besteht aus einer Vielzahl sogenannter autonomer Systeme (AS). Die Verwaltung eines AS unterliegt in der Regel einem Netzanbieter. Große Netzanbieter können mehrere AS verwalten, während kleine Netzanbieter sich ein AS teilen können. Im Kontext der Arbeit erfolgt die Annahme, dass der Verwaltungsbereich eines Netzanbieters immer exakt ein AS umfasst. Die Begriffe autonomes System und ISP-Domäne haben eine synonyme Bedeutung.

Reduktion des Inter-ISP-Datenverkehrs, kann allerdings die Performanz des Schwarms reduzieren bzw. die Diversität der Daten im ISP-Netzwerk einschränken. Diesem Problem wird durch den Einsatz eines Content Delivery Networks in Form adaptiver Server entgegengewirkt. Im Falle eines Performanz-Verlusts existiert somit ein sekundärer Distributionsweg, der als Ausweichlösung dienen kann.

Die Arbeit diskutiert den Entwurf eines ISP-seitigen Mechanismus zur selektiven Steuerung der Zugangsprofile der Peers im ISP-Netzwerk. Peers, die sich im Sinne des ISPs vorteilhaft verhalten, sollen ein höheres Zugangsprofil zugesprochen bekommen. Das Verfahren arbeitet adaptiv, so dass in regelmäßigen Intervallen eine erneute Zuweisung von Zugangsprofilen erfolgt. Es ist zu erwarten, dass dieses adaptive Bandbreitenmanagement die Reduktion von Inter-ISP-Datenverkehr – insbesondere in Kollaboration mit lokalitätsfördernden Mechanismen – ermöglicht. Des Weiteren ist zu erwarten, dass durch die Promotion ausgewählter Peers die Diversität der Daten im ISP-Netzwerk zunimmt, was im Umkehrschluss die Arbeit lokalitätsfördernder Mechanismen begünstigt. Der Einsatz des adaptiven Bandbreitenmanagements wirft eine Reihe interessanter Fragestellungen auf, deren Beantwortung im Zuge der Arbeit erfolgt:

- Wie kann man sicherstellen, dass nur diejenigen Peers eine erhöhte Bandbreite zugewiesen bekommen, die sich vorteilhaft für den ISP verhalten?
- Ist es sinnvoll, nicht nur ausgewählte Peers zu befördern, sondern Peers, die sich unvorteilhaft verhalten, durch eine verminderte Bandbreite zu bestrafen?
- Profitiert der Mechanismus von dem Einsatz der adaptiven Server-Architektur?
- Was sind die infrastrukturellen Anforderungen für den ISP, um das adaptive Bandbreitenmanagement umzusetzen?
- Ist es notwendig, das Verhalten des eingesetzten P2P-Systems zu modifizieren, damit das Bandbreitenmanagement effektiv eingesetzt werden kann?

Der wissenschaftliche Beitrag der vorliegenden Arbeit umfasst mehrere Aspekte. Kernbestand ist die prototypische Entwicklung und Erprobung des adaptiven Bandbreitenmanagements in einem Live Testbed. Im Fokus der Untersuchungen steht die Reduktion des Inter-ISP-Datenverkehrs an primärer Stelle. Die Ziele, die ein Content Provider oder ein CDN verfolgt, werden aus der Betrachtung nicht ausgeschlossen, sind jedoch von sekundärer Natur. Zur Erprobung des adaptiven Bandbreitenmanagements in einer hybriden Systemarchitektur ist eine Server-Komponente erforderlich. Die Wahl diesbzgl. ist auf eine Strategie der adaptiven Server gefallen, deren erstmalige Beschreibung in [9, 10] erfolgt. Diese Arbeiten umfassen simulative Studien zur Performanz der konkreten Strategien. Hierzu zählt auch die Supporter-Strategie, die im Rahmen dieser Arbeit verwendet, prototypisch implementiert und ausführlichen Performanztests unterzogen wurde, um die Ergebnisse der simulativen Studien zu stützen. Überdies findet der implementierte Prototyp der Supporter-Strategie Anwendung innerhalb des EU-Forschungsprojekts *SmoothIT* und ist dort Gegenstand weiterer Untersuchungen.

Die Ergebnisse der Evaluation zeigen, dass der Einsatz einer hybriden Systemarchitektur unter Zuhilfenahme des adaptiven Bandbreitenmanagements die Interessen aller beteiligten Instanzen begünstigt:

- Content Provider: Durch die Verwendung einer P2P-basierten Lösung für den primären Distributionsweg ergeben sich keine bis wenige Kosten für den Content Provider.
- CDN: Der Einsatz einer P2P-basierten Lösung verringert die Server-Last zu allgemeinen Zeiten. Zu Stoßzeiten kann das CDN als sekundärer Distributionsweg eingesetzt werden, so dass im Falle eines Performanz-Verlusts des P2P-Systems der Endbenutzer keine Einbußen hinsichtlich der Dienstgüte erfährt.
- ISP: Durch die Verwendung des adaptiven Bandbreitenmanagements ist der ISP in der Lage, die Kosten für den Datenverkehr über Inter-ISP-Verbindungen weiter einzuschränken, ohne dabei die Performanz des P2P-Systems wesentlich zu beeinträchtigen.
- Endbenutzer: Die Dienst- und Erfahrungsgüte erfüllen die Anforderungen der Benutzer.

1.3 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich wie folgt. Möglichkeiten zur Verteilung von Daten im Internet, sowie Basiswissen zu P2P-Systemen im Hinblick auf die Verbreitung von Video-on-Demand Inhalten werden einführend vorgestellt (Kapitel 2). Bestehende Strategien zur Reduktion des Inter-ISP-Datenverkehrs sind seit geraumer Zeit Gegenstand

der Forschung. Kapitel 3 präsentiert eine Taxonomie über den bereits bestehenden Ansätzen und stellt deren Funktionsweise, Vor- und Nachteile gesondert heraus. Gleichzeitig rundet die Taxonomie den einführenden Teil der Arbeit ab. Im Anschluss daran erfolgt die Motivation und Einführung des adaptiven Bandbreitenmanagements in Theorie und Praxis (Kapitel 4). Der Mechanismus erfordert eine konkrete Strategie zur Umsetzung von Zugangsprofilen. Eine im Rahmen dieser Arbeit durchgeführte Studie soll aufzeigen, ob mit einer speziellen Softwarelösung – Linux Traffic Control – dies prinzipiell möglich ist (Appendix A). Für die Realisierung der vorgestellten Mechanismen bedarf es einiger Implementierungsarbeit: Die Emulation eines Video-on-Demand Clients, die Realisierung und Integration einer ausgewählten adaptiven Server-Strategie und die Implementierung des adaptiven Bandbreitenmanagements. Das Design der erforderlichen Komponenten wird ausführlich im Hinblick auf Funktionsweise und Integration in den NextShare Client beschrieben (Kapitel 5). Die anschließende Leistungsbewertung gliedert sich in drei separate Kapitel. Zunächst beschreibt Kapitel 6 die die Methodik der Evaluation. Die Performanz der eingesetzten adaptiven Server-Strategie ist Gegenstand von Kapitel 7. Die Evaluation des adaptiven Bandbreitenmanagements erfolgt in Kapitel 8. Den Abschluss dieser Arbeit bildet die kritische Diskussion der implementierten Mechanismen und der Ergebnisse der Leistungsbewertung, sowie ein Ausblick auf mögliche Erweiterungen und zukünftige Entwicklungen in diesem Segment (Kapitel 9).

2 Grundlagen

Gegenstand dieses Kapitels ist die Erarbeitung einer grundlegenden Wissensbasis, die essentiell für das Verständnis der vorliegenden Arbeit ist. Abschnitt 2.1 geht zunächst auf die Verteilstrategien im heutigen Internet ein und diskutiert die Vor- und Nachteile der entsprechenden Ansätze. Die Erörterung der hybriden Systemarchitektur, die in der vorliegenden Arbeit konkret Verwendung findet, ist ebenfalls Gegenstand dieses Abschnitts. Die Basis für die Implementierungsarbeiten ist der NextShare-Client, eine Tribler-Variante, die das BitTorrent und Give-to-Get Protokoll beherrscht. Aus diesem Grunde thematisiert Abschnitt 2.2 zunächst die prinzipielle Funktionsweise von BitTorrent. Peer-to-Peer gestütztes Video Streaming ist Gegenstand von Abschnitt 2.3. Hierin erfolgt eine Beschreibung der vorherrschenden Topologien und eine kurze Erläuterung des Give-to-Get Protokolls. Abschnitt 2.4 stellt die Supporter-Strategie als Server-Komponente vor. Die hybride Systemarchitektur unter Verwendung des Give-to-Get Protokolls und der Supporter-Strategie ist Gegenstand der Diskussion in Abschnitt 2.5.

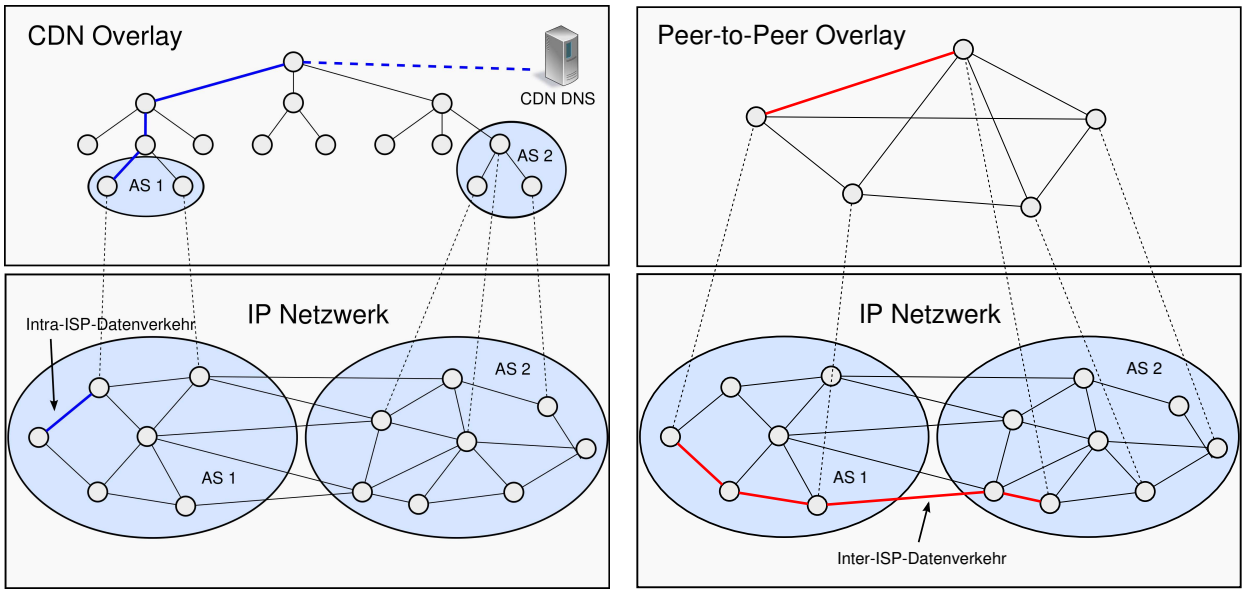
2.1 Verteilung von Daten im Internet

Die Verteilung von Daten im Internet findet klassischerweise über Client-/Server-Architekturen (bspw. CDNs) statt. Obwohl diese Architektur durchaus einige Vorzüge hat, so überwiegen doch die Nachteile gerade dann, wenn populäre Inhalte einer breiten Masse von Benutzern zugänglich sein sollen. Client/Server-Lösungen sind hier durch eine begrenzte Bandbreitenanbindung des Servers limitiert, der nur eine eingeschränkte Anzahl von Benutzern zeitgleich mit Inhalten versorgen kann, ohne dabei die Dienstgüte einzuschränken. Dienstgüte ist ein Maß für die Güte der Leistungen, die ein Kommunikationsservice erbringt. Der Begriff der Dienstgüte vereint dabei verschiedene Metriken, welche technische Aspekte des Systems beschreiben. Diese Metriken können vielfältig gewählt werden. Zu den grundlegenden Metriken zählen der Datendurchsatz, die Ende-zu-Ende-Verzögerung beim Übertragen von Datenpaketen, Jitter oder aber auch die potentielle Verfügbarkeit von Daten. Im Kontext eines Video-on-Demand Systems kann man Dienstgüte insbesondere bezüglich der Startup- und Stalling-Verzögerung beschreiben. Die Startup-Verzögerung ist die Zeitspanne zwischen dem Anfordern eines gewünschten Inhalts bis zu dessen Wiedergabe. Die Stalling-Verzögerung beschreibt dagegen die Gesamtzeit von Pausierungen der Wiedergabe. Stalling-Verzögerung tritt bspw. dann auf, wenn Videodaten mit einer unzureichenden Datenrate bezogen werden. Der Begriff der Erlebnisqualität beschreibt den subjektiven Eindruck, der beim Benutzer des Kommunikationssystems entsteht. Unverkennbar ist, dass Erlebnisqualität entscheidend von den technischen Kapazitäten – und damit den genannten Metriken zur Dienstgüte – beeinflusst wird.

Des Weiteren stellen zentralisierte Lösungen immer einen Single Point of Failure dar: Fällt die Server-Hardware aus, so ist gleichzeitig auch der Dienst für Benutzer bzw. potentiell zahlende Kunden nicht mehr verfügbar. Der Standort des Servers spielt ebenfalls eine gewichtige Rolle. Benutzer, die über weitreichende Verbindungen einen Dienst erreichen, der nur von einem Server bereitgestellt wird, müssen in der Regel hohe Latenzzeiten in Kauf nehmen, um die gewünschten Inhalte abrufen zu können. Dies mag für Anbieter von Download-Archiven akzeptabel sein. Spätestens wenn zeitkritische Dienste genutzt werden, wie bspw. bei Video-on-Demand Systemen, ist die Einhaltung einer entsprechenden Dienstgüte hingegen von hoher Priorität.

2.1.1 Content Delivery Netzwerke

Content Delivery Netzwerke weichen diese Probleme auf, indem sie ein hierarchisches Overlay von miteinander vernetzten Servern über der Topologie des Internets etablieren [11]. Abbildung 2.1(i) zeigt, dass ein hierarchisches CDN aus einem Ursprungsserver besteht, der die angebotenen Inhalte speichert. Der Server verteilt diese Inhalte über einen Distributionsmechanismus an Replika-Server. Die Abbildung suggeriert, dass sich ein Replika-Server innerhalb einer ISP-Domäne befindet. Dies ist eine vereinfachende Annahme und nicht die Regel in der Praxis. In der Praxis findet man häufig auch Server-Farmen vor, die sich außerhalb der Grenzen einer ISP-Domäne befinden. Sendet ein Client eine Anfrage an das CDN, so wird diese entsprechend der Lokalität des Clients an einen Replika-Server weitergeleitet, der in Bezug auf die physische Lokalität in der Nähe des Clients liegt. Die Weiterleitung erfolgt in der Regel über einen Redirektionsmechanismus, der von CDN-eigenen Domain Name Servern



(i) CDN-Overlay auf Netzschicht

(ii) P2P-Overlay auf Netzschicht

Abbildung 2.1: Qualitative Darstellung der Beziehungen zwischen CDN- und P2P-Overlays und der unterliegenden Netzschicht.

bereitgestellt wird [11]. Der Replika-Server beantwortet die ursprüngliche Anfrage des Clients und liefert die gewünschten Inhalte aus. Das System basiert nach wie vor auf einer Client-/Server-Architektur, minimiert aber durch Replikation der Daten auf weitere, verteilte Server nicht nur das Risiko eines zentralen Serverausfalls, sondern auch die Kommunikationspfade zwischen Konsument und Anbieter und damit auch die Latenz der Datenübertragung. Su et al. [12] bestätigen in ihrer Studie, dass durch den Einsatz eines CDN, wie bspw. Akamai, Transportwege eine signifikante Reduktion hinsichtlich ihrer Länge erfahren.

2.1.2 Peer-to-Peer Netzwerke

Robustheit und Skalierbarkeit sind Attribute, die teuer durch zusätzliche Server-Hardware und der Notwendigkeit für intelligente Überwachungsmechanismen, welche die Dienstgüste des Netzes jederzeit sicherstellen, erkauft werden müssen. Im Gegensatz dazu stellen Peer-to-Peer Netzwerke eine kostengünstige Alternative zur Verteilung von Inhalten dar, da ein Anbieter – insbesondere auch ein CDN – mit minimalem finanziellen Eigenanteil sein Angebot einer breiten Masse zugänglich machen kann. Während bei einem CDN die Transportwege durch einen Redirektionsmechanismus verkürzt werden, zieht die überwiegende Anzahl der Peer-to-Peer Anwendungen, die sich derzeit im Einsatz befinden, die Topologie der unterliegenden Netzschicht nicht in Betracht. Dadurch entstehen in der Regel lange Kommunikationspfade (vgl. Abbildung 2.1(ii)), die sich über die Grenzen der lokalen ISP-Domäne hinaus erstrecken. In Anbetracht der Popularität von gesuchten Inhalten ist es oftmals gar nicht nötig, weitreichende Verbindungen aufzubauen. Mit hoher Wahrscheinlichkeit werden diese Daten auch von Peers verteilt, die bzgl. ihrer Lokalität günstig liegen.

Der Umstand langer Kommunikationspfade wird vor allem von ISPs kritisch betrachtet, da für Verbindungen, die in andere ISP-Domänen reichen, Kosten entstehen, die ihnen in Rechnung gestellt werden. Ein ISP ist jedoch daran interessiert, die Aufwendungen für den operativen Geschäftsbetrieb minimal zu halten und bevorzugt natürlich Verbindungen, die innerhalb seiner Domäne zur Datenübertragung genutzt werden. Derartiger Datenverkehr verursacht keine hohen Kosten für den ISP und ist somit *ISP-freundlich*.

2.2 BitTorrent

BitTorrent ist ein Bespiel einer ursprünglich für reines File-Sharing ausgelegten Peer-to-Peer Applikation, deren Funktionsweise maßgeblich auf einem kooperativem Verhalten seiner Benutzer beruht [13]. Die Grundidee hinter BitTorrent ist relativ einfach: Während ein Benutzer im klassischen Client-Server-Modell eine gewünschte Datei lediglich von einem Server bezieht, stellen BitTorrent-Clients die Anteile des Downloads, die bereits bezogen worden sind, anderen Peers zur Verfügung. Dadurch wird die Last, die vorher am Server angefallen ist, auf einzelne Peers verteilt. BitTorrent erreicht hierdurch eine hohe Effizienz, die unter Ausnutzung von internen Mechanismen sogar soweit gesteigert werden kann, dass sich eine Pareto-Optimalität einstellt [13]. Pareto-Optimalität bedeutet nichts anderes, als dass ein Peer die Auslastung seiner Upload- und Download-Kapazitäten nicht weiter steigern könnte, ohne dass sich selbige bei einem anderen Peer um mindestens den gleichen Wert verschlechtern würde. Im Rahmen von BitTorrent spricht man von einem BitTorrent-Client als *Leecher* [13], wenn dieser gerade aktiv eine Datei bezieht (aber auch schon selbst Anteile dieser Datei zur Verfügung stellt), bzw. als *Seeder* [13], wenn die Datei bereits komplett heruntergeladen wurde und der Client somit nur noch Daten zur Verfügung stellt. Um zwischen Daten, die ein Peer bereits empfangen hat und solchen, an denen er weiterhin interessiert ist, möglichst einfach zu unterscheiden, teilt BitTorrent einen Download in mehrere Datenfragmente fester Größe, sogenannte *Chunks* ein. Diese Fragmente werden nochmals in Untereinheiten unterteilt. Der Datenaustausch zwischen Peers erfolgt ausschließlich in der Größenordnung dieser Untereinheiten, die wir im Zuge dieser Arbeit fortan als Datenblöcke benennen. Aus Gründen der Datenintegrität darf ein Peer einen Datenblock erst dann anbieten, wenn der korrespondierende Chunk vollständig heruntergeladen und gegen einen SHA1-Hashwert verglichen wurde [13].

Die ursprüngliche Variante von BitTorrent ist allerdings kein vollständig verteiltes System, da es eine zentrale Instanz benötigt, die Peers hilft, sich gegenseitig zu finden. Diese Instanz wird als *Tracker* bezeichnet und nimmt als einfacher Index-Server eine entscheidende Rolle in der Distribution von Inhalten über BitTorrent ein. BitTorrent-basierte Systeme, wie bspw. Azureus und Tribler, bieten allerdings auch die Möglichkeit an, Tracker-unabhängig über eine verteilte Hashtabelle zu arbeiten.

Wir werden im Folgenden die Funktionsweise von BitTorrent unter Einbezug der Rollen von Peers und des Trackers exemplarisch erläutern. Den Abschluss dieses Abschnitts bildet eine Diskussion diverser BitTorrent-interner Mechanismen. Diese Diskussion ist essentiell für ein umfassendes Verständnis der Funktionsweise und stellt ferner die Basis für weiterführende Optimierungen BitTorrent-ähnlicher Systeme dar.

2.2.1 Funktionsweise

Die Verteilung von Daten über ein BitTorrent-basiertes Peer-to-Peer Netzwerk beginnt typischerweise mit einem initialen Seeder. Dieser Seeder erstellt eine Metainformationsdatei, das sogenannte *Torrent*. Diese Metainformationsdatei enthält alle relevanten Angaben, die andere Peers benötigen, um die gewünschten Inhalte herunterzuladen. Zu diesen Metadaten[1] gehören bspw. die Adresse des zugehörigen Trackers, aber auch die Chunk-Größe, die Anzahl der Chunks sowie eine Liste von SHA1-Hashwerten, die den einzelnen Chunks zugehörig sind und zur Verifikation heruntergeladener Inhalte herangezogen werden können [14]. Der initiale Seeder publiziert die Torrent-Datei bspw. auf einem Webserver (vgl. Abbildung 2.2, Schritt 1) und macht sie somit vielen potentiellen Interessenten zugänglich.

Interessierte Benutzer laden Metainformationsdateien von einem Webserver herunter und starten den zugehörigen Download (vgl. Abbildung 2.2, Schritt 2). Mit dieser Aktion kontaktieren sie den entsprechenden Tracker und treten dadurch dem sogenannten *Schwarm* bei. Unter einem Schwarm versteht man die Menge aller Benutzer, die an den selben Daten interessiert sind.

Sobald der Peer dem Schwarm beigetreten ist, kann er eine Anfrage an den Tracker stellen, um eine Liste von anderen Peers zu erhalten, die sich im selben Schwarm befinden (vgl. Abbildung 2.2, Schritt 3). Der Tracker ist die

[1] An dieser Stelle muss vermerkt werden, dass der Begriff *Metadaten* in der fachbezogenen Literatur unterschiedliche semantische Ausprägungen kennt. Metadaten beschreiben bspw. in Bezug auf Datenbankmanagementsysteme jene Daten, die Aussagen über strukturelle Merkmale einer Datenbank liefern. Der Begriff *Metadatum* wird allerdings auch dann verwendet, wenn Informationen über ein Objekt oder einem Zusammenschluss von Objekten geliefert werden, die diese Entität genauer beschreiben. Im Kontext dieser Arbeit wird der Begriff mit der letztgenannten Semantik belegt.

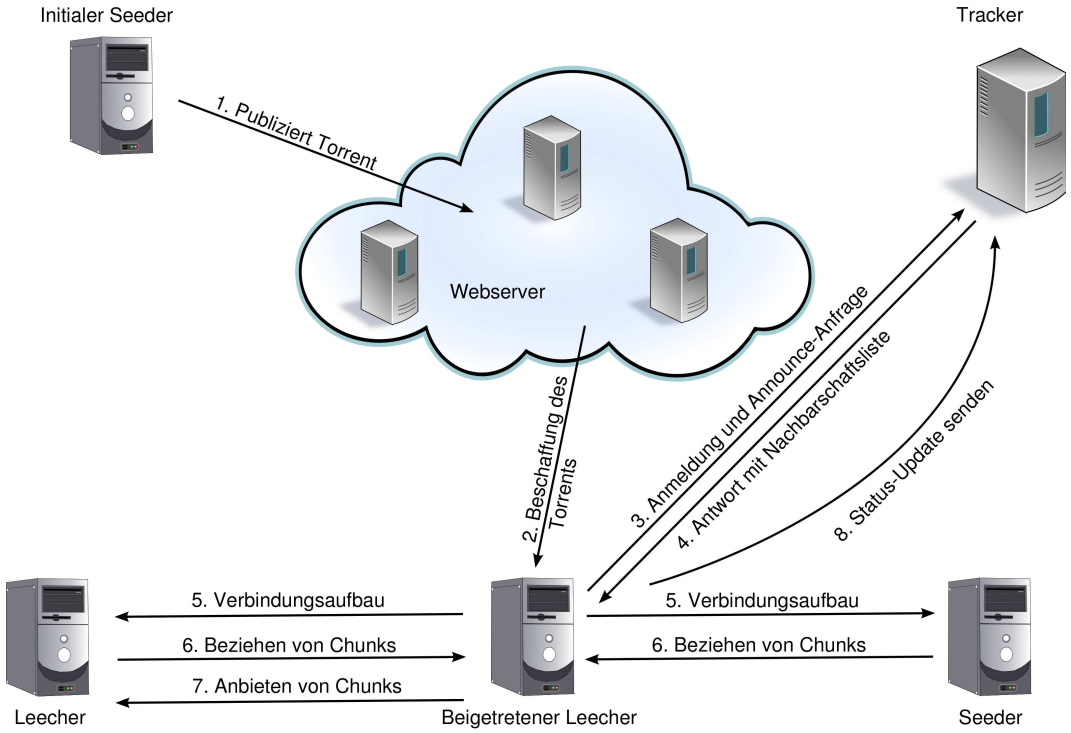


Abbildung 2.2: Temporale Abfolge der Aktionen in einem typischen BitTorrent-Szenario

einzige zentrale Komponente innerhalb des BitTorrent-Systems und kennt alle aktiven Peers, die sich in den von ihm verwalteten Schwärmen aufhalten [13]. Die Kommunikation zwischen Peer und Tracker erfolgt über ein simples Interface auf Basis des HTTP-Protokolls [14]. Aufgrund seiner Einfachheit und der Notwendigkeit der An- und Abmeldung einzelner Peers wird diese Komponente gerne für die Aggregation schwarm-bezogener Statistiken modifiziert. Auf eine Anfrage eines Peers antwortet der Tracker mit Schwarm-bezogenen Daten, insbesondere aber mit einer Liste (vgl. Abbildung 2.2, Schritt 4) von 3-Tupeln der Form (Peer-ID, IP-Adresse, Port-Nummer) [14]. Tupel dieser Liste repräsentieren potentielle Nachbarn des anfragenden Peers. Letzterer stellt im Folgenden Verbindungen zu den vorgeschlagenen Tauschpartnern her (vgl. Abbildung 2.2, Schritt 5). Wichtig ist, dass hierbei nicht zwangsweise an jeden Nachbarn Datenblöcke hochgeladen werden. Vielmehr ist eine Verbindung mit einem Zustand behaftet, der zwischen *blockiert* und *nicht blockiert* unterscheidet. Der Unterschied zwischen beiden Zuständen besteht darin, dass von blockierten Nachbarn lediglich Datenblöcke bezogen werden können, während zwischen dem Peer und nicht blockierten Nachbarn tatsächlich ein Austausch von Daten stattfindet. In der Standardvariante von BitTorrent unterhält ein Peer vier nicht blockierte Verbindungen, über die Datenblöcke gesendet werden können [13]. Der Peer prüft allerdings periodisch, ob sich an den Bandbreitenkapazitäten anderer Nachbarn etwas geändert hat und setzt den Zustand einzelner Verbindungen gemäß einer Ersetzungsstrategie gegebenfalls neu. Der zugrundliegende Mechanismus wird in Abschnitt 2.2.3 ausführlich diskutiert.

Über Verbindungen zu Nachbarn können Datenblöcke bezogen werden (vgl. Abbildung 2.2, Schritt 6). Die Auswahl der anzufragenden Datenblöcke spielt dabei eine zentrale Rolle. BitTorrent kann an dieser Stelle verschiedene Strategien benutzen. Die Rarest-First-Strategie[2], bei der ein Peer zuerst solche Chunks komplettiert, die von den wenigsten seiner Nachbarn bereits heruntergeladen worden sind, hat sich als besonders effektiv erwiesen [15]. Um die Download-Kapazität möglichst auszureizen, erfolgt das Herunterladen von Chunks parallel.

Nachdem alle Datenblöcke eines Chunks heruntergeladen worden sind, darf selbiger vom Peer zum Tausch angeboten werden (vgl. Abbildung 2.2, Schritt 7). Das Austauschen von Inhalten ist eine zentrale Komponente von BitTorrent und orientiert sich an theoretischen Überlegungen aus der Spieltheorie. In Anlehnung an die Tit-for-Tat-Strategie werden Daten zu denjenigen Peers gesendet, von denen Daten bezogen werden können. Während diese Strategie maßgeblich zur globalen Effizienz von BitTorrent beiträgt, muss man dennoch festhalten, dass sie in der Startphase Anlaufschwierigkeiten hat, da ein Peer zu diesem Zeitpunkt selbst noch keine Chunks anbieten kann.

[2] Abseits der Rarest-First-Strategie existieren weitere Regeln, bspw. Strict-Priority oder Random-First-Piece.

Sobald ein Peer alle Chunks des zugehörigen Downloads erhalten hat, meldet er dies dem Tracker (vgl. Abbildung 2.2, Schritt 8). Er kann nun weiter an der Distribution der Daten im Schwarm teilnehmen, oder aber den Schwarm verlassen. In diesem Fall gebietet es die Konvention, dass der Peer sich am Tracker explizit abmeldet [16].

2.2.2 Verwaltung der Nachbarschaftsliste

Bei Eintritt in den Schwarm kontaktiert ein Peer den zugehörigen Tracker, um eine Liste von potentiellen Tauschpartnern zu erhalten. Diese Anfrage lässt sich prinzipiell beliebig oft wiederholen, bspw. wenn der Peer an neuen Nachbarn interessiert ist. Auf Seiten des Trackers erfolgt die Selektion allerdings zufallsbasiert, ohne Lokalitätseigenschaften einzelner Peers zu berücksichtigen.

2.2.3 Arbeitsweise des Choke-Algorithmus

Um die vorhandene Upload-Kapazität eines Peers möglichst optimal auszunutzen, ist die Anzahl der Upload-Slots in BitTorrent beschränkt. Dies hat gute Gründe, denn der Datentransfer per BitTorrent erfolgt über TCP-Verbindungen und TCP skaliert erst ab einer gewissen Mindestdatenrate sinnvoll. Die Anzahl der interessierten Peers ist in der Regel größer als die Anzahl der verfügbaren Upload-Slots, so dass ein Mechanismus darüber entscheiden muss, zu welchen Peers Daten gesendet werden und welchen nicht. BitTorrent löst dieses Problem mit dem sogenannten Choke-Algorithmus.

Die konkrete Funktionsweise des Choke-Algorithmus hat sich über die Entwicklung der Produktlinie des ursprünglichen BitTorrent-Clients ebenfalls verändert. Der Choke-Algorithmus von BitTorrent vor Version 4.0.0 war bspw. mit dem Problem behaftet, keine faire Allokation von Upload-Slots zu gewährleisten, wenn der Peer nur noch Datenblöcke anbietet, selbst aber keine mehr bezieht [15]. Dies wurde in Version 4.0.0 korrigiert. Die jüngste Version von BitTorrent[3] fasst die Allokation von Upload-Slots erneut anders auf und soll aufgrund ihrer Aktualität Gegenstand der folgenden Betrachtung sein.

Die Re-Evaluierung der Zustände aller Nachbarn wird in festen Zeitintervallen δ durchgeführt und zusätzlich, wenn eines der folgenden Ereignisse eintritt:

- Ein Nachbar des Peers hat die Verbindung getrennt.
- Ein nicht blockierter Nachbar des Peers kündigt sein Interesse an einem Datenblock an.
- Ein nicht blockierter Nachbar des Peers kündigt an, dass er kein Interesse mehr an einem Datenblock hat.

Wir sprechen im Kontext dieser Betrachtung von einem Durchlauf, wenn die Re-Evaluierung der Choke-Zustände durchgeführt wird. In jedem Durchlauf wird der Kern des Choke-Algorithmus, die `rechoke`-Methode, abgearbeitet. Diese Methode arbeitet in mehreren Schritten. Zunächst erfolgt eine Gruppierung aller Nachbarn in zwei unterschiedliche Listen:

- Eine Downloader-Liste, die solche Verbindungen beinhaltet, von denen der Peer Datenblöcke beziehen möchte und die selbst Interesse an Datenblöcken haben, die der Peer anbietet. Nachbarn, die in der letzten Zeit selbst keine Daten zur Verfügung gestellt haben, werden ausgeschlossen.
- Eine Seeder-Liste, die Verbindungen zu interessierten Nachbarn beinhaltet, zu denen aber nur noch Datenblöcke gesendet werden.

Es erfolgt eine Priorisierung beider Listen. Die Downloader-Liste wird nach der Download-Transferrate zu den entsprechenden Nachbarn absteigend sortiert. Die Sortierung der Seeder-Liste erfolgt nach mehreren Kriterien. Peers, die ihre Nachrichten verschlüsselt austauschen, werden gegenüber solchen Peers, die ihre Daten unverschlüsselt versenden, bevorzugt. Das zweite Kriterium sieht eine Priorisierung nach der Upload-Transferrate vor. Haben zwei Peers, zu denen Daten gesendet werden, die selbe Transferrate, so wird der Nachbar gewählt, der eine niedrigere Position in der Seeder-Liste hat. Damit die Allokation von Upload-Slots im Seeder-Modus dennoch fair bleibt, wird nach einer bestimmten Anzahl von Durchläufen ein Round-Robin-Mechanismus angewandt, der die vollständige

[3] Zum Zeitpunkt der Anfertigung dieser Arbeit ist dies Version 5.3.0 des Linux-basierten Clients. Dieser und vorherige Releases können unter `http://www.bittorrent.com` bezogen werden.

Nachbarschaftsliste durchläuft und den ersten blockierten Nachbar, der an Datenblöcken des Peers interessiert ist, an den Anfang dieser Liste stellt. Dies beeinflusst die nachfolgende Einteilung in die oben genannten Listen, ebenso wie deren Sortierung.

Der nächste Schritt beinhaltet die konkrete Zuweisung von Nachbarn zu Upload-Slots. Die sortierten Peer-Listen werden anhand eines Verhältnisses in Abhängigkeit zu der maximal verfügbaren Anzahl von Upload-Slots aufgeteilt. Sei die maximale Anzahl dieser Slots m, dann gilt für beide Listen:

- Es werden $l \approx m \cdot 70\%$ Slots mit Nachbarn belegt, die ein Tauschverhältnis mit dem Peer eingehen (es werden sowohl Datenblöcke empfangen als auch gesendet.).
- Es werden $s \approx m \cdot 30\%$ Slots mit Nachbarn belegt, zu denen lediglich Daten gesendet werden.

Es muss natürlich für beide Listen geprüft werden, ob die Anzahl der dort geführten Nachbarn die entsprechende Anzahl von Slots belegen kann. Führt die Downloader-Liste bspw. weniger als $m \cdot 70\%$ Nachbarn, dann werden frei verfügbare Slots mit weiteren Nachbarn aus der Seeder-Liste belegt, die regulär nicht gewählt werden würden. Analog gilt dies für den Fall, in dem die Seeder-Liste zu wenige potentielle Kontakte enthält. Aus den jeweiligen Listen werden im Folgenden die besten l respektive s Nachbarn gezogen und nicht länger blockiert. Alle übrigen Nachbarn können allerdings noch einen sogenannten Optimistic Unchoke erhalten und somit ebenfalls Datenblöcke des lokalen Peers beziehen. Der Optimistic Unchoke ist ein Mechanismus, um neue und potentiell gute Nachbarn zu entdecken (unabhängig davon, was die gemessenen Download-Transferraten über die Güte aussagen, denn es kann sein, dass ein Tauschpartner seine Upload-Transferrate erst dann signifikant erhöht, wenn er ebenfalls Daten des Peers erhält). Die Anzahl der Nachbarn, die durch dieses Verfahren in den Unchoke-Zustand gelangen, wird anhand einer Formel errechnet und ist im Wesentlichen abhängig von der Zahl bereits selektierter Nachbarn und der maximalen Anzahl verfügbarer Upload-Slots. Der Choke-Algorithmus iteriert nun über die verbleibenden Nachbarn, wobei explizit der Blockiert-Zustand vergeben wird, wenn eine der folgenden Bedingungen erfüllt ist:

1. Der Nachbar bietet keine Datenblöcke an, die der lokale Peer benötigt.
2. Die Anzahl der Optimistic Unchokes wurde bereits ausgeschöpft.

Im Gegensatz dazu wechselt ein blockierter Nachbar in den unblockierten Zustand, wenn diese Bedingungen nicht erfüllt sind. Die Anzahl der durchgeführten Optimistic Unchokes wird jedoch nur erhöht, wenn dieser Nachbar an Datenblöcken des Peers interessiert ist. Insbesondere heißt dies, dass sich potentiell mehr Nachbarn im ungeblockten Zustand befinden, als Upload-Slots verfügbar sind. Der Choke-Algorithmus gewährleistet aber, dass zu jeder Zeit die Anzahl der Nachbarn, zu denen tatsächlich Datenblöcke gesendet werden, gleich der Anzahl der verfügbaren Upload-Slots ist.

Interessant ist, dass bei dieser Implementierung des Choke-Algorithmus der Wechsel in den Seeding-Zustand (es werden nur noch Datenblöcke angeboten), der in früheren Versionen explizit vom Leeching-Zustand (es werden selbst noch Datenblöcke benötigt) unterschieden wurde, fließend geschieht und schon während des Herunterladens von Datenblöcken Verbindungen aufrecht erhalten werden, die ein einseitiges Tauschverhältnis beschreiben. Die Implementierung ist allerdings, wie bereits in früheren Versionen, mit dem Problem behaftet, dass die priorisierten Listen einzig und allein durch Download- bzw. Upload-Transferraten bestimmt werden. Dies begünstigt natürlich Verbindungen zu solchen Nachbarn, die eine gute Netzwerkanbindung haben, auch wenn diese eine hohe Distanz bezüglich der unterliegenden Netzschicht zum lokalen Peer aufweisen. Dadurch können, wie bereits bei der Selektion potentieller Nachbarn durch den Tracker (vgl. Abschnitt 2.2.2), potentiell weitreichende Kommunikationspfade präferiert werden, die über die Grenzen des ISP hinausgehen.

2.3 Peer-to-Peer gestütztes Video Streaming

Mit der wachsenden Popularität von Video-Inhalten, die über das Internet verteilt werden, steigt auch die Last für Client-/Server-Architekturen, die in diesem Segment typischerweise eingesetzt werden. Peer-to-Peer Streaming Systeme versprechen hier eine Umverteilung der anfallenden Last auf einzelne Teilnehmer des Systems und stellen somit eine kostengünstige Alternative dar. Grundsätzlich lassen sich derartige Systeme in zwei Kategorien einteilen, die auf deren topologische Ausprägung zurückzuführen sind: Solche Peer-to-Peer Systeme, die eine baum-basierte Topologie aufweisen und solche, die eine mesh-basierte Topologie ausbilden. Ferner unterscheidet man die Art und Weise des Video Streams. Bei einem sogenannten Live-Stream ist der Sendezeitpunkt des Videos festgelegt, so

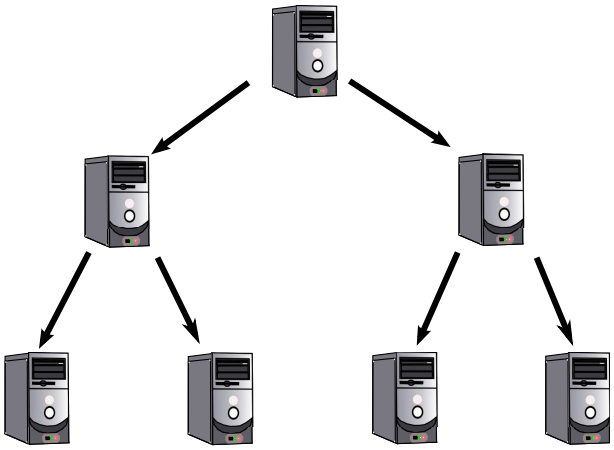


Abbildung 2.3: Qualitative Darstellung einer baum-basierten Topologie

dass sich Benutzer daran zeitlich orientieren müssen. Die Video-Daten werden in diesem Fall synchronisiert an alle Teilnehmer des Peer-to-Peer Streaming Systems verteilt. Davon unterscheiden sich sogenannte Video-on-Demand Aufzeichnungen, denn hier entscheidet der Benutzer selbst, wann er das gewünschte Video betrachten möchte. Das hat zur Folge, dass sich Abspielpositionen von Benutzern, die das gleiche Video betrachten, voneinander unterscheiden. Die Verteilung der Daten unterliegt in diesem Szenario komplexeren Anforderungen, da Peers nicht nur solche Datenblöcke beziehen müssen, die sie unmittelbar für das konstante Abspielen des Videos benötigen, sondern bestenfalls auch Videofragmente herunterladen, die rar im System sind und für sie erst zu einem späteren Zeitpunkt eminent werden.

Der Fokus der vorliegenden Arbeit liegt auf Video-on-Demand Systemen. Einige wichtige Attribute der baum- und mesh-basierten Topologie gleichen sich für Video-on-Demand und Live-Streaming, so dass in den nachfolgenden Abschnitten zunächst eine allgemeingültige Einführung in die oben angeführten Topologien erfolgt. Die Diskussion vertieft durch zusätzliche Verweise die besonderen Anforderungen, die für ein Peer-to-Peer basiertes Video-on-Demand System erfüllt werden müssen. Abschnitt 2.3.3 stellt einen konkreten Ansatz zur effizienten Verteilung von Video-on-Demand Daten in Peer-to-Peer Systemen vor, der unter dem Namen *Give-to-Get* Einzug in die Fachliteratur erhalten hat. An dieser Stelle müssen selbstverständlich die Unterschiede zwischen herkömmlichen Peer-to-Peer Lösungen, wie bspw. dem in Abschnitt 2.2 vorgestellten BitTorrent-System, und Peer-to-Peer Streaming Applikationen aufgezeigt werden. Dies erfolgt in Abschnitt 2.3.3 exemplarisch anhand einer Gegenüberstellung von BitTorrent und Give-to-Get.

2.3.1 Baum-basierte Topologie

Multicasting ist ein probates Mittel, um viele Teilnehmer, die an den gleichen Inhalten interessiert sind, von einer zentralisierten Stelle aus zu erreichen. Auf Ebene der Netzschicht, dem sogenannten IP-Multicasting, ist dies allerdings nicht skalierungsfähig, da intermediäre Router die Zugehörigkeit zu einer IP-Multicast-Gruppe verwalten müssen. Das Übertragungsprinzip bleibt jedoch interessant und wurde jüngst auf die Applikationsebene übertragen [17]. Eine Multicast-Gruppe auf Applikationsebene[4] wird dabei durch einen gerichteten Baum (vgl. Abbildung 2.4) beschrieben, wobei der Wurzelknoten den Content Provider bzw. Server-Hardware auszeichnet. Der Wurzelknoten steht, wie auch alle inneren Knoten der Arboreszenz, in einer Vater-Kind-Beziehung zu nachfolgenden Knoten und leitet Video-Inhalte über diese gerichteten Verbindungen weiter. Durch diese Architektur werden Daten sukzessive über den Multicast-Baum propagiert, bis sogenannte Blattknoten erreicht werden. Blattknoten repräsentieren Benutzer, die lediglich Daten empfangen, aber aktiv an der Weiterverteilung der Inhalte nicht teilnehmen. Der baum-basierten Topologie liegt eine inhärente Synchronität zugrunde, da Datenpakete von der Quelle, sprich, dem Wurzelknoten, aktiv in die Multicast-Gruppe geleitet und dort lediglich weiterverteilt werden. Sie ist daher insbesondere für Live Streaming geeignet, allerdings auch mit Nachteilen behaftet, die im Folgenden angesprochen werden sollen.

[4] Wenn im Folgenden von einem Multicast gesprochen wird, ist dabei immer eine Multicast-Gruppe auf Applikationsebene gemeint, es sei denn, der Text weist explizit auf eine andere Bedeutung hin.

Dadurch, dass die Übermittlung von Datenpaketen konsekutiv über die Ebenen des Multicast-Baumes erfolgt, erreicht ein Datenpaket A einen Benutzer der Ebene k früher als einen Benutzer, der auf Ebene $k+1$ des Baumes liegt. Diese Verzögerung wächst linear in der Anzahl der Ebenen des Multicast-Baumes und sollte weitestgehend minimiert werden. Eine Möglichkeit, dieses Ziel zu erreichen, ist den Baum so zu konstruieren, dass dieser möglichst in die Breite wächst, aber nicht zu sehr in die Tiefe [17]. Das bedeutet jedoch, dass die Arität einzelner Knoten möglichst hoch sein sollte, damit ein Peer Videodaten an viele Benutzer propagieren kann. Dieser Ansatz skaliert schlecht, da gerade bei asymmetrischen Netzwerkanbindungen die Upload-Kapazität eines Peers meist stark limitiert ist.

Problematisch ist ebenfalls die Verwaltung des Multicast-Baumes. Verlässt ein Teilnehmer das Peer-to-Peer Streaming System, so formen alle nachfolgenden Peers dieses Teilnehmers einen eigenständigen Baum, der vom Server und den übrigen Knoten isoliert ist. Die Wiederherstellung des Baumes erfordert einen Mechanismus, der Wissen über alle Teilnehmer haben und pflegen muss. Ansätze zur Lösung dieses Problem existieren und unterscheiden sich grundlegend in ihrer Realisierung. So wurde bspw. eine zentralisierte Lösung vorgeschlagen, bei der ein Server dazu eingesetzt wird, die Topologie zu überwachen und neuen Benutzern eine entsprechende Position innerhalb des Multicast-Baumes zuzuweisen. Der Server arbeitet mit expliziten An- und Abmeldungen und prüft die Aktivität der Knoten anhand eines Timeouts. Der Verlust von Knoten kann zwar festgestellt und korrigiert werden, jedoch geschieht dies zeitverzögert und ereignet sich im schlechtesten Fall erst nach Ablauf des kompletten Timeouts. Der zentralisierte Ansatz ist zudem mit dem Nachteil behaftet, dass er schlecht mit der Anzahl der Benutzer skaliert und einen sogenannten Single Point of Failure darstellt. Verteilte Ansätze, welche die zuletzt genannten Probleme umgehen, wurden vorgeschlagen. Nach [17] ist die Effizienz des zentralisierten wie auch verteilten Ansatzes nicht ausreichend, um eine hinreichend gute Dienstgüte zu erzielen.

Benutzer, die in den Blattknoten des Multicast-Baumes liegen, tragen nicht zur aktiven Verteilung von Video-Inhalten bei. Betrachtet man einen b-ären Multicast-Baum der Tiefe k, so lässt sich folgendes für das Verhältnis von Blattknoten zur Größe des Baumes festhalten:

- Die Anzahl der Blattknoten in diesem Baum ist b^k.
- Die Anzahl aller Knoten, inklusive der Wurzel, ist $\sum_{i=0}^{k} b^i$.
- Das Verhältnis $R(b,k)$ zwischen Blattknoten und der Gesamtzahl der Knoten ist dann (unter Verwendung der endlichen geometrischen Reihe):

$$R(b,k) = \frac{b^k}{\sum_{i=0}^{k} b^i} = \frac{b^k(1-b)}{1-b^{k+1}} = \frac{b^k - b^{k+1}}{1-b^{k+1}} = \frac{b^k}{1-b^{k+1}} - \frac{b^{k+1}}{1-b^{k+1}} = \frac{b^k}{1-b^{k+1}} + \frac{b^{k+1}}{b^{k+1}-1}$$

Für $R(2,5)$ heißt dies, dass knapp über 50% der Benutzer ihre Upload-Kapazität nicht zur Verfügung stellen. Für $R(3,5)$ liegt dieser Wert bereits bei $\approx 67\%$. Für wachsende b erhöht sich der Anteil von Blattknoten, was gerade in Bezug auf die ursprüngliche Optimierungsidee, die Arität einzelner Knoten zu erhöhen, damit der Baum nicht zu tief wird, ein interessantes Ergebnis ist. Um diesem Problem grundsätzlich entgegenzuwirken, wurden Vorschläge für Multi-Baum-basierte Topologien erarbeitet [18, 19]. Diesen Ansätzen ist gemein, dass der Server den Videostream in mehrere Substreams spaltet. Für jeden Substream existiert nun ein Multicast-Baum über den der Server korrespondierende Datenpakete verteilt. Jeder Teilnehmer ist Mitglied in jedem dieser Multicast-Gruppen, allerdings variiert seine individuelle Position innerhalb der Multicast-Bäume von Substream zu Substream. Wann immer ein Peer eine Position als innerer Knoten einnimmt, wird seine Upload-Kapazität zum Verteilen von Daten ausgenutzt. Derartige Ansätze sind besonders in heterogenen Szenarien interessant, da ein Peer je nach dem wie gut seine Netzwerkanbindung ist, häufiger als innerer Knoten (der Peer verteilt Daten) positioniert werden kann, als in der Rolle eines Blattknoten (der Peer empfängt nur Daten).

In Bezug auf Video-on-Demand ist insbesondere die inhärente Synchronität von Multicast-Bäumen ein Problem. Ein Vorschlag, um die notwendige Asynchronität über einer baum-basierten Topologie zu etablieren, liefert [20]. Die Arbeit basiert auf Patching-Mechanismen, die bereits auf Ebene von IP-Multicasts in Video Streaming Szenarien vorgeschlagen worden sind [21, 22]. Der Grundgedanke ist hierbei, dass Benutzer gemäß ihrer Ankunftszeit in sogenannte Sessions eingeteilt werden. Eine Session beinhaltet alle Benutzer, deren individuelle Ankunftszeiten innerhalb eines vorher festgelegten Schwellwertes liegen. Peers einer Session formen zusammen mit dem Server, der wieder die Wurzel repräsentiert, einen Multicast-Baum auf Applikationsebene. Videodaten werden nach wie vor als Stream vom Server an alle Teilnehmer baumabwärts gesendet. Tritt nun ein neuer Peer der Session bei,

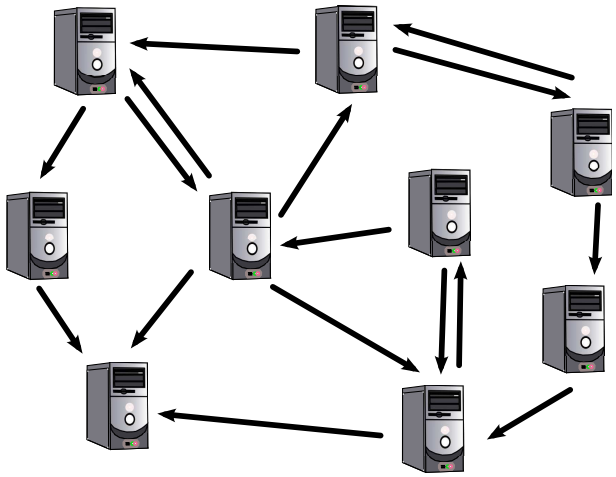


Abbildung 2.4: Qualitative Darstellung einer mesh-basierten Topologie

so wird er entsprechend in den Multicast-Baum eingegliedert und erhält die Videodaten, die kontinuierlich über den Server propagiert werden. Um die Teile des Videos, die bereits gesendet worden sind und dem neuen Peer fehlen, ebenfalls zu erhalten, muss ein sogenannter Patch angefordert werden, der diese Videodaten beinhaltet. Ein Patch kann sowohl von dem Server als auch von Peers bereitgestellt werden, die entsprechende Videodaten zuvor empfangen haben. Der Ansatz erfordert natürlich, dass Peers bereits gesehene Videofragmente zwischenspeichern müssen, was eine Speicherplatzbelastung in Abhängigkeit der Länge des Schwellwertes zur Folge hat. Durch den Patch-Mechanismus werden allerdings die Anforderungen eines Video-on-Demand Dienstes hinsichtlich der Asynchronität hinreichend erfüllt.

2.3.2 Mesh-basierte Topologie

Ein wesentlicher Unterschied zwischen einer mesh- und baum-basierten Topologie ist, dass ein Peer nicht mehr in einer Vater-Kind-Relation zu einem anderen Peer steht, sondern mehrere Verbindungen aufrecht halten kann, durch die Daten in *beide* Richtungen fließen können (vgl. Abbildung 2.4). Verbindungen zu Nachbarn ändern sich dynamisch und somit nicht wie in einer baum-basierten Topologie statisch vorgegeben. Ein Peer bleibt in der Regel auch dann noch mit anderen Peers verbunden, wenn einer seiner Nachbarn das System verlässt. Sinkt die Konnektivität eines Peers beträchtlich, so kann er explizit nach neuen potentiellen Nachbarn suchen und entsprechende Verbindungen aufbauen. Dieses Design macht die mesh-basierte Topologie vor allem sehr robust gegenüber Ausfällen bzw. Peer Churn. Unter Peer Churn versteht man das dynamische Verhalten von Teilnehmern eines Peer-to-Peer Systems im Hinblick auf deren Ankunfts- und Absprungzeit. Damit dieser Mechanismus funktioniert und Peers sich gegenseitig finden können, ist die Architektur eng an der Architektur von BitTorrent angelehnt. Es existiert ein zentraler Index-Server, der sogenannte Tracker, der eine ähnliche Funktionsweise hat, wie der Tracker in BitTorrent. Die Entscheidung, ob ein Peer A eine Verbindung mit vom Tracker vorgeschlagenen Nachbarn B_i eingeht, kann grundsätzlich von verschiedenen Kriterien abhängig gemacht werden.

Weitere Kennzeichen einer mesh-basierten Topologie lassen sich an der Art und Weise festmachen, wie Daten ausgetauscht werden. In Abschnitt 2.3.1 wurde beschrieben, dass in einem baum-basierten Overlay Videodaten sukzessive in das System gegeben und dort weiterverteilt werden. Diese implizite Synchronität findet man bei einer mesh-basierten Topologie nicht. Ferner ist die Einheit des Datentauschs analog zu BitTorrent ein Datenblock, der zu einem Videofragment gehört. Ein initialer Seeder, der bspw. durch den Content Provider bzw. durch dessen Server-Hardware gestellt wird, verteilt diese Datenblöcke an seine Nachbarn, wobei diese empfangene Daten ebenfalls an ihre Nachbarn weiterleiten können. Da Peers in einer mesh-basierten Video-on-Demand Applikation ihre Abspielzeitpunkte selbst bestimmen, ergibt sich durch die vorherrschende Asynchronität das Problem, dass Peers zum selben Zeitpunkt an der Komplettierung unterschiedlicher Videofragmente interessiert sind. Dieses Problem wird dadurch verschärft, dass ein Peer in der Regel Datenblöcke zufallsbasiert bezieht; er lädt diejenigen Datenblöcke herunter, die durch seine Nachbarn zur Verfügung gestellt werden. Eine strikte zeitliche Abfolge kann somit wesentlich schwieriger gewährleistet werden, als dies bspw. bei einer mesh-basierten Live-Streaming Anwendung

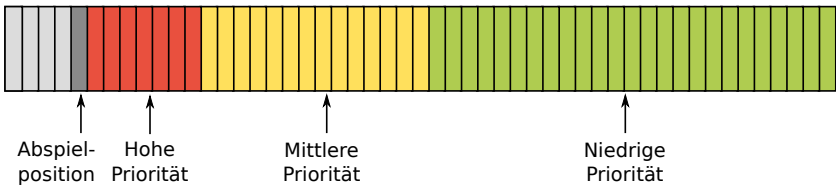


Abbildung 2.5: Kategorisierung ausstehender Chunks in Relation zur Abspielposition unter Verwendung des Give-to-Get Protokolls.

der Fall wäre. Stalling-Effekte können somit entstehen, was sich durch anhaltendes Pausieren des Videos negativ auf die Erlebnigsgüte eines Benutzers auswirkt.

Ein Peer sollte, auch im Sinne der maximalen Ausnutzung seiner Bandbreite, immer einen Partner finden, mit dem er Daten austauschen kann. Dies kann durch eine hohe Diversität von Datenblöcken im gesamten System erreicht werden. Peers, die das Abspielen des Videos beendet haben, besitzen allerdings keinen Anreiz, länger im System zu bleiben und Datenblöcke zu verteilen. Gerade dies wäre wichtig, denn solche Peers haben den kompletten Videoinhalt zwischengespeichert und können eine Vielzahl von potentiellen Nachbarn mit Datenblöcken versorgen. Neu hinzukommende Peers könnten insbesondere dadurch profitieren, da ein reiner Seeder als Bootstrap-Mechanismus fungieren könnte.

Entscheidende Designfragen sind daher, wie Ressourcen optimal unter Berücksichtigung der Asynchronität einzelner Peers im System allokiert werden können bzw. wie am günstigsten Nachbarschafsverhältnisse aufgebaut werden, so dass nicht nur der lokale Peer, sondern das System in seiner Gesamtheit davon profitiert. Ansätze für Lösungsstrategien existieren: Bspw. wurde die Verwendung von Server-Hardware vorgeschlagen, die dann von einem Peer genutzt und zum Bezug kritischer Datenblöcke herangezogen werden kann, wenn dieser es nicht über das Overlay geschafft hat, selbige rechtzeitig zu beziehen [23, 24]. Durch eine intelligente Blockselektion kann die Performanz des Systems ebenfalls gesteigert werden. Auch hier gibt es unterschiedliche Strategien. Bspw. wurde in [25, 24] eine Priorisierung von Datenblöcken durch eine Einteilung in Mengen mit niedriger und hoher Priorität vorgeschlagen. Diese Ansätze verfolgen die Optimierung der Blockselektionsstrategie allerdings aus einer lokalen, und damit egoistischen, Sichtweise. Jüngste Arbeiten zeigen, dass ein Kompromiss [26] zwischen lokaler und globaler Sichtweise durch einen Voting-Mechanismus erreicht werden kann. Durch diesen Ansatz ist es möglich, die Blockselektion nicht nur auf einen lokalen Peer zu beschränken, sondern auch das, was für eine Gruppe von Benutzern am besten ist, zu forcieren.

Dennoch sind in diesem Bereich viele Frage nach wie vor ungelöst und Gegenstand aktueller Forschung. Im Lichte dessen soll die Studie [27] nicht unerwähnt bleiben. Die Autoren haben in dieser Arbeit durch Experimente in Simulation gezeigt, dass eine mesh-basierte Topologie gemessen an Effizienz und Robustheit deutlich vor einem baum-basierten Overlay liegt.

2.3.3 Give-to-Get

Die Skalierfähigkeit zentralisierter Lösung für Video-on-Demand Systeme ist teuer zu erkaufen. Peer-to-Peer Systeme schaffen diesbzgl. Abhilfe, sind aber mit einem anderen Problem behaftet: Dem sogenannten Free-Riding. Darunter versteht man im Kontext von Peer-to-Peer Systemen einen Peer, der signifikant mehr Ressourcen konsumiert, als er selbst dem Schwarm bereitstellt. Die Güte eines P2P-Systems ist daher maßgeblich abhängig von altruistisch agierenden Peers, die andere Benutzer mit Datenblöcken versorgen. Überwiegt der Bedarf der Leecher[5] die Kapazität der Gesamtheit altruistischer Peers, so kann dies nicht mehr gewährleistet werden. Mol et al. [28] thematisieren dieses Problem und stellen eine Lösung in Form des Systems Give-to-Get (G2G) vor, in dem jeder Benutzer den Anreiz hat, möglichst viel von seiner Upload-Kapazität dem Schwarm zur Verfügung zu stellen. Der Ansatz ist relativ einfach: Peers verteilen die Datenblöcke, die sie von einem Peer *A* erhalten haben, selbst an andere Peers weiter. Erfolgt dies in hinreichendem Maße, erhalten diese Peers weiterhin Datenblöcke von Peer *A*. Das

[5] Da Give-to-Get viele Ähnlichkeiten zu BitTorrent besitzt, wird im Kontext dieses Abschnitts dieselbe Terminologie wie in Abschnitt 2.2 verwendet.

Problem des Free-Ridings wird zwar nicht ganzheitlich eliminiert, allerdings so stark limitiert, dass ein Free-Rider keine vernünftige Dienstgüte erwarten kann, wenn die Upload-Kapazität des Systems nahezu ausgeschöpft ist.

G2G selbst arbeitet unabhängig von einer bestimmten Video-Kodierung und ist damit relativ flexibel. Es unterteilt ein Video in Chunks konstanter Größe, was zur Folge hat, dass Frame-Grenzen nicht mit Chunk-Grenzen zusammenfallen müssen. Im Kontext der vorliegenden Arbeit ist die Effizienz des G2G-Systems von besonderer Bedeutung. G2G wird diesbzgl. den Anforderungen an ein P2P-gestütztes Video-on-Demand System gerecht. Zum Einen verwendet G2G eine sogenannte Prebuffering Strategie, die besagt, dass vor der Wiedergabe eines Videos Chunks in ausreichender Anzahl vorliegen müssen. Zum Anderen priorisiert G2G noch nicht heruntergeladene Chunks durch Einteilung in drei unterschiedliche Gruppen (siehe Abbildung 2.5). Chunks, deren Position in Relation zur Abspielposition näher liegen, weisen eine höhere Priorität als andere Chunks auf und werden bevorzugt heruntergeladen. G2G geht durch diesen Mechanismus explizit auf die zeitlichen Anforderungen bei der Wiedergabe von Videoinhalten ein. Insgesamt sorgen beide Strategien für eine hohe Effizienz und tragen dazu bei, dass die Wiedergabe eines Videos möglichst unterbrechungsfrei bleibt. Die folgenden Abschnitte gehen auf einzelne, wesentliche Aspekte des G2G-Protokolls näher ein.

Verwaltung der Nachbarschaftsliste

Ein zentraler Aspekt bei G2G ist, wie bei jedem Peer-to-Peer gestützten System, das Entdecken neuer Nachbarn, zu denen Daten gesendet oder von denen Daten empfangen werden. [28] geht nicht auf die konkrete Strategie zum Aufspüren neuer Nachbarn ein, gibt allerdings Anhaltspunkte dafür, wie eine solche prinzipiell aussehen könnte. Unabhängig von der eingesetzten Strategie, fordert ein Peer nach dem Beitritt in das P2P-System eine Liste von 10 Nachbarn an. Die Nachbarschaftssuche kann analog zu BitTorrent wiederholt durchgeführt werden, bspw. wenn bei erstmaligem Anfragen keine ausreichende Anzahl von Nachbarn zurückgeliefert wurde, oder wenn der Peer an neuen, potentiell besseren Nachbarn interessiert ist. Zu all diesen Nachbarn unterhält der Peer Verbindungen, über die er Datenblöcke beziehen bzw. senden kann. Sobald ein Peer seinen Video-Download abgeschlossen hat, also selbst nicht länger an neuen Datenblöcken interessiert ist, trennt er alle Verbindungen zu anderen Seedern. Dadurch werden Verbindungen eingespart, über die ansonsten Daten gesendet werden würden, die lokal nicht mehr verarbeitet werden könnten.

Verteilung von Chunks

Ein Peer bezieht einen Chunk, wenn er diesen explizit von einem seiner Nachbarn anfordert. Die Grundlage hierfür ist, dass alle Peers die Chunks, die sie bereits heruntergeladen haben, bei ihren Nachbarn annoncieren. In Analogie zu BitTorrent entscheidet ein Peer selbst darüber, welche Peers seiner Nachbarschaftsliste Anfragen zur Übertragung spezifischer Chunks stellen dürfen. Für jede Verbindung zu einem Nachbar wird eine Sende-Warteschlange verwaltet, die alle Chunks beinhaltet, die vom korrespondierenden Nachbar angefragt worden sind. Chunks werden zu gegebener Zeit der Warteschlange entnommen und an den entsprechenden Nachbar hochgeladen. Die Einheit des Datenaustausch erfolgt, ebenfalls wie in BitTorrent, in der Größenordnung von Datenblöcken, die eine Untereinheit der Video Chunks repräsentieren.

Arbeitsweise des Choke-Algorithmus

Der Algorithmus, der darüber entscheidet, welche Nachbarn anfragen stellen dürfen und welche nicht, arbeitet in ähnlicher Weise wie der unter 2.2.3 vorgestellte Choke-Mechanismus von BitTorrent. Der Choke-Algorithmus von G2G arbeitet ebenfalls in Runden, die nach einer Zeitspanne von δ Sekunden angestoßen werden. Während einer Runde erfolgt die Re-Evaluierung der Zustände aller Nachbarn. Dabei wird die Nachbarschaftsliste eines Peers A jedoch nicht aufgrund von Download-Raten oder dergleichen priorisiert, sondern anhand des Upload-Verhaltens der Nachbarn B_i von A. Dies geschieht über einen Vergleich der Anzahl von Chunks, die Nachbar B_k von A empfangen und an seine Nachbarn weiterverteilt hat. Ist dieser Wert für zwei Nachbarn B_k und B_l mit $k \neq l$ identisch, so entscheidet die Gesamtzahl aller hochgeladener Chunks dieser Peers über den besseren Listenplatz. Beide Sortierkriterien sind nötig, um einen gerechten Ablauf des Choke-Algorithmus zu gewährleisten [28]. Die

hierfür notwendige Information erhält Peer *A*, indem er sogenannte Feedback-Verbindungen zu den Nachbarn seiner Nachbarn aufbaut und diese nach der Anzahl empfangener Chunks fragt. Standardmäßig erhalten die drei Peers mit dem besten Upload-Verhalten den Wechsel in den Unchoke-Zustand. Um die Upload-Kapazität des Peers *A* möglichst auszureizen, wechseln weitere Peers solange in den Unchoke-Zustand, bis $\approx$ 90% von *A*'s Upstream-Bandbreite verwendet wird, oder die maximale Anzahl von Upload-Verbindungen erreicht worden ist. Analog zu dem Choke-Verfahren von BitTorrent (vgl. Abschnitt 2.2.3) wird bei G2G ebenfalls der Optimistic Unchoke durchgeführt. Dieser erfolgt nur alle 2δ Runden und arbeitet die verbleibenden Nachbarn nach einem Round-Robin-Schema ab. Ferner heißt dies, dass ein Peer, der den optimistischen Unchoke erhalten hat, 2δ Sekunden lang Zeit hat, sich als Peer mit gutem Weiterleitungs-Verhalten zu etablieren. Für jeden Peer existiert der Anreiz, möglichst viele Datenblöcke zu seinen Nachbarn hochzuladen, um damit unmittelbar selbst einen guten Upload-Slot in der Nachbarschaft eines anderen Peers zu bekommen. Die Ausreizung der Upload-Kapazität eines Peers erfolgt anhand der priorisierten Nachbarschaftsliste, so dass Free-Rider nur dann nicht blockiert werden, wenn hinreichend viel freie Kapazität im gesamten System übrig ist [28].

Auswahl einzelner Chunks

Um einen spezifischen Chunk zu erhalten, muss ein Peer eine konkrete Anfrage an einen seiner Nachbarn stellen. Zu jedem Chunk wird aus Sicht des lokalen Peers eine Deadline mitgeführt. Ist dieser Zeitpunkt überschritten, so verliert der Chunk an Wert, da er sowieso nicht mehr abgespielt werden würde. Die Reihenfolge, mit der fehlende Chunks angefragt werden, obliegt jedem Peer selbst, wobei G2G standardmäßig die nachfolgend beschriebene Strategie verfolgt. Ein Peer wird immer dann eine Anfrage durchführen, wenn er von einem seiner Nachbarn nicht blockiert wird und die folgenden Bedingungen zutreffen:

1. Der Peer hat Interesse an einem Chunk *m*, den der Nachbar besitzt.
2. Der Peer besitzt *m* nicht und hat bislang noch kein Interesse an *m* geäußert. Er hat allerdings freie Kapazitäten und kann *m* beziehen.
3. Chunk *m* erreicht den Peer potentiell bevor dessen Deadline abgelaufen ist.

Ferner sollte eine gute Auswahlstrategie die folgenden Anforderungen beherzigen [28]:

1. Chunks müssen in einer gewissen Reihenfolge eintreffen, damit die Wiedergabe des Videos fortgeführt werden kann.
2. Im Sinne einer maximalen Ausreizung der Upload-Kapazität soll der Peer Chunks beziehen, die von anderen Peers unmittelbar oder künftig gesucht sind.

Die Anforderungen 1 und 2 stehen orthogonal zueinander, so dass sich ein Peer mit einem Kompromiss konfrontiert sieht: Lädt er zunächst Chunks in Reihe herunter, oder bevorzugt er Chunks, die selten im System sind und die mit hoher Wahrscheinlichkeit auch von anderen Peers gesucht sind? G2G löst dieses Problem, indem verbleibende Chunks, also solche, deren Zeitpunkt hinter der aktuellen Abspielzeit des Videos liegt, in drei unterschiedliche Mengen einordnet. Diese Mengen bestehen jeweils aus Chunks der gleichen Priorität, wobei die Priorität eines einzelnen Chunks anhand der zeitlichen Differenz zwischen seiner Abspielzeit und der aktuellen Zeit des Videos festgelegt wird. Während der Wiedergabe lädt ein Peer bevorzugt Chunks mit hoher Priorität herunter. Chunks verändern ihre Priorität mit fortschreitender Wiedergabe. Ein geringer Anteil der Download-Kapazität verwendet ein Peer, um rare Chunks niedriger Priorität zu beziehen. Dies hat zum Einen den Vorteil, dass der Peer zu gegebenem Zeitpunkt als Anbieter von Chunks für seine Nachbarn interessant wird. Zum Anderen ermöglicht es eine rasche Komplettierung des Videos, da der Peer rare Chunks bereits frühzeitig besitzt.

Abgrenzung von Give-to-Get zu BitTorrent

Die Hauptunterschiede zwischen G2G und BitTorrent lassen sich anhand der internen Mechanismen beschreiben, die das jeweilige System charakterisieren.

- **Blockselektionsstrategie**:
  G2G unterscheidet bei der Auswahl von anzufragenden Chunks drei Prioritätsklassen (vgl. Abbildung 2.5): Chunks, die nahe an der derzeitigen Abspielposition liegen und somit eine hohe Priorität haben, sowie

Chunks mittlerer und niedriger Priorität. Durch diese Priorisierung werden Chunks entsprechend ihrer Wiedergabeposition angefordert. In BitTorrent ist eine solche Strategie nicht notwendig. Hier ist es wichtiger, den Download möglichst schnell zu komplettieren und insbesondere seltene Chunks bevorzugt zu beziehen. Des Weiteren nutzt G2G einen kleinen Anteil der verfügbaren Bandbreite, um Chunks mit niedriger Priorität zu beziehen (Prefetching). Dies erhöht die Verteilung von Chunks im System und sorgt für eine bessere Auslastung der Bandbreiten.

- **Choke-Algorithmus**:
  Obwohl das grundlegende Prinzip für beide Systeme ähnlich ist, unterscheiden sie sich in einem Aspekt sehr stark. BitTorrent blockiert Nachbarn nicht, die sich als schnelle Up- und Downloader erwiesen haben, sprich, möglichst viel Bandbreite ausnutzen und durch Upload-Slots ihre Ressourcen dem Schwarm zur Verfügung stellen. G2G distanziert sich von diesem an die Tit-for-Tat-Strategie angelehnten Prinzip, da dieses nur in einem Szenario mit bilateralen Interessen zufriedenstellend funktionieren kann. Durch die Relevanz des Abspielzeitpunktes eines Chunks sind die Interessen eines Peers in einem Video-on-Demand System meist *einseitig*. Aus diesem Grund wird die Nachbarschaftsliste im Hinblick auf das Verteilungsverhalten eines Peers priorisiert.
- **Vorzeitiges Puffern**:
  Durch vorzeitiges Puffern soll ein möglichst unterbrechungsfreies Wiedergeben des Videos ermöglicht werden. Bevor die Wiedergabe starten kann, muss sich eine gewisse Anzahl von Chunks bereits im Ausgabepuffer des lokalen Peers befinden. Die Berechnung der Anzahl notwendiger Chunks ist abhängig von der aktuellen Download-Transferrate. Aus diesem Maß kann die Zeitspanne bis zum Abschluss des Downloads extrapoliert werden. Aus der Zeitspanne wiederum erhält man die Anzahl der Chunks, die notwendig sind, um eine Wiedergabe über die verbleibende Download-Zeit aufrecht zu halten. BitTorrent hingegen ist eine reine Filesharing-Applikation, so dass das vorzeitige Puffern entfällt.

2.4 Adaptive Server-Allokationsstrategien

Peer-unterstützte Video-on-Demand Systeme zeichnen sich durch eine hohe Dynamik bezüglich der Peers aus, die gleichzeitig Interesse an einem Video haben. Gepaart mit einer hohen Diversität in den Bandbreitenkapazitäten einzelner Peers ergibt sich daraus das Problem, dass eine hinreichende Dienstgüte auf Ebene des einzelnen Benutzers nicht zu jedem Zeitpunkt durch den Schwarm gewährleistet werden kann. Um diesem Problem entgegenzuwirken, definierten Pussep et al. [10] sogenannte adaptive Server-Allokationsstrategien. Eine solche Strategie zeichnet sich dadurch aus, dass sie Peers in Form von dedizierten Servern, die sich ebenfalls im Overlay befinden, Unterstützung bei Unterversorgung anbietet. Server-Allokationsstrategien bedienen sich hierbei einer einfachen Architekur: Ein Überwachungsmechanismus M beobachtet die Performanz einzelner Peers und entscheidet individuell für jeden Peer anhand einer Entscheidungsmetrik Δ, ob Unterstützung in Form eines Servers erfolgen muss. Sofern sich unterstützungswürdige Peers im Overlay befinden, aktiviert der Überwachungsmechanismus entsprechende Server-Prozesse und allokiert einen Peer zu einem Server. Entscheidend dabei ist das Verbindungsmanagement C, welches nach [9] durch den Server-Prozess realisiert wird. Der Überwachungsmechanismus ist dafür verantwortlich, eine vernünftige Zuteilung von Peers zu Servern durchzuführen. Der Server ist in der Regel ein Peer mit hoher Bandbreite, der regulär im Schwarm teilnimmt und das Protokoll des eingesetzten Peer-to-Peer Systems spricht. Eine aktive Datenübertragung seitens des Servers an Peers findet jedoch nur dann statt, wenn der Überwachungsmechanismus dies zuvor signalisiert hat.

2.4.1 Supporter-Strategie

Die Supporter-Strategie ist eine solche adaptive Server-Allokationsstrategie und wurde in [10, 9] konzeptuell beschrieben und in Simulation erprobt. Sie stellt den einzelnen Benutzer in den Fokus und prüft, ob dieser Unterstützung durch einen dedizierten Server, dem sogenannten Supporter, benötigt oder nicht. Durch diesen Ansatz soll verhindert werden, dass einzelne Benutzer eine schlechte Dienstgüte erfahren, obwohl die Performanz des gesamten Schwarms gut erscheint. Die Supporter-Strategie sorgt ferner für einen Kompromiss zwischen Server-Auslastung und Einhaltung einer gewissen Dienstgüte. Durch das dynamische Aktivieren und Deaktivieren einzelner Supporter passt sich die Strategie der aktuellen Defizit-Situation des Schwarms an. In Zeiten niedriger Last kann so eine Reduktion der akkumulierten Server-Last erfolgen, ohne dabei die Dienstgüte zu vernachlässigen. In

Zeiten hoher Last hingegen können weitere Server aktiviert werden, um den wachsenden Peer-Bedarf hinreichend abzudecken.

2.4.2 Funktionsweise der Supporter-Strategie

Peers registrieren sich zunächst an der Überwachungskomponente und können Unterstützung durch subsequente Unterversorgt-Nachrichten anfordern, wenn ihr Wiedergabepuffer nicht vollständig gefüllt ist, sprich, sämtliche Chunks mit hoher Priorität (vgl. 2.3.3) nicht rechtzeitig geladen wurden. Zu diesem Zeitpunkt erfolgt eine Beobachtung des Zustands des unterversorgten Peers durch den Überwachungsmechanismus. Ein Peer meldet explizit durch das Senden einer Versorgt-Nachricht, dass er keine weitere Unterstützung mehr benötigt.

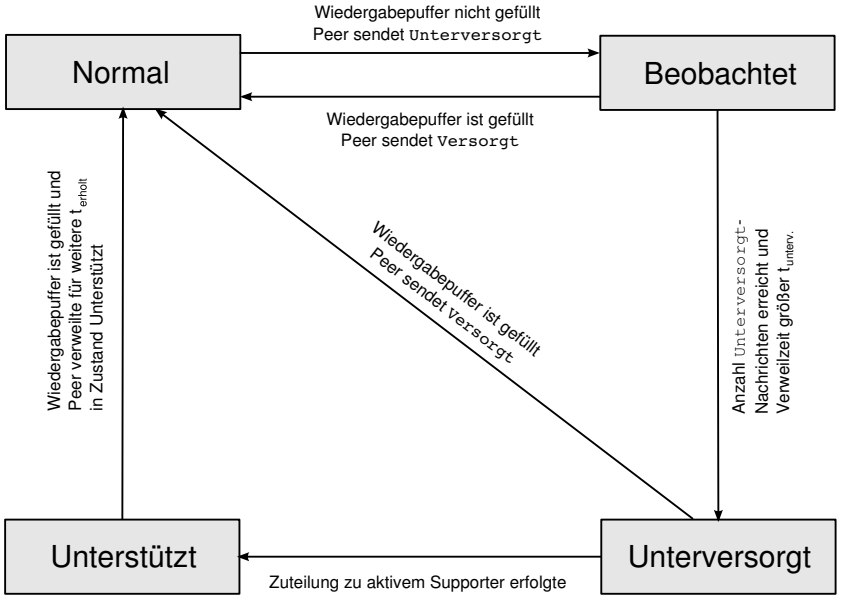


Abbildung 2.6: Zustandsdiagramm der Peer-Zustände auf Seiten des Überwachungsmechanismus

Während der Beobachtung von Peers durch die Überwachungskomponente durchlaufen selbige verschiedene Zustände. Abbildung 2.6 stellt das korrespondierende Zustandsdiagramm dar. Nachdem ein Peer p sich an der Überwachungskomponente registriert hat, weist diese ihm zunächst den Zustand *Normal* zu. Durch das Senden einer Unterversorgt-Nachricht wechselt der Peer in den Zustand *Beobachtet*. Zu diesem Zeitpunkt erfolgt keine Zuteilung zu einem Supporter, da der Überwachungsmechanismus zunächst sicherstellen muss, dass sich der Zustand des unterversorgten Peers p von alleine nicht bessert. Dieser Schritt ist notwendig, da anderenfalls zu viele Peers einem Supporter zugewiesen werden würden, obwohl ihre Unterversorgtheit nur eine temporäre Erscheinung gewesen ist. Dies würde ferner dazu führen, dass Supporter fälschlicherweise beansprucht würden, während Peers, die tatsächlich Unterstützung benötigen, nicht hinreichend versorgt werden können. Die folgenden Ereignisse erzwingen einen Zustandswechsel:

1. Der Zustand des Peers p hat sich nicht verbessert, was durch das Empfangen subsequenter Unterversorgt-Nachrichten der Überwachungskomponente signalisiert wird. Sofern der Peer p bereits eine gewisse Zeit $t_{\text{unterv.}}$ im *Beobachtet*-Zustand verweilt und eine Mindestanzahl m an Unterversorgt-Nachrichten gesendet hat, wechselt er in den Zustand *Unterversorgt*.
2. Peer p konnte seinen Wiedergabepuffer füllen. Peer p interpretiert dies als Zustandsverbesserung und meldet der Überwachungskomponente, dass eine Unterstützung nicht mehr notwendig ist. In diesem Fall wechselt der Zustand zurück nach *Normal*.

Sobald Peer p durch den Überwachungsmechanismus als unterversorgter Peer erkannt wurde, kann er einem aktiven Supporter zugeteilt werden. Dies geschieht durch die Überwachungskomponente selbst. Über die konkrete Zuteilung werden die entsprechenden Supporter informiert. Ein Supporter ist passiv, wenn er keine Peers versorgt. Man bezeichnet einen Supporter als aktiv, wenn er eine gewisse Mindestanzahl l von Peers versorgt. An dieser Stelle gilt es auf Seiten des Überwachungsmechanismus die nachfolgenden Fälle zu unterscheiden:

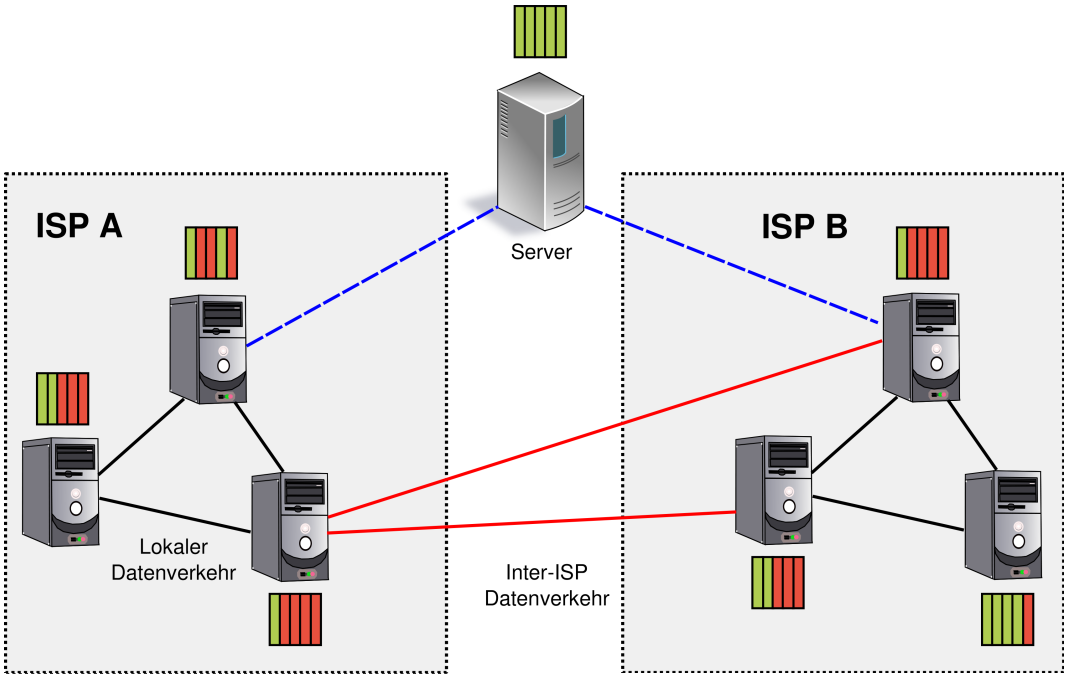


Abbildung 2.7: Darstellung der hybriden Systemarchitektur eines Peer-to-Peer gestützten Video-on-Demand Systems

1. Es befindet sich kein aktiver Supporter im Overlay. Die Überwachungskomponente versucht nun, einen passiven Supporter zu finden, dem mindestens l unterversorgte Peers zugewiesen werden können. Kann ein solcher Server nicht gefunden werden, verweilt der Peer weiter im Zustand *Unterversorgt*, bis sich l unterversorgte Peers im System befinden und der Supporter aktiviert werden kann.
2. Es befindet sich ein aktiver Supporter im Overlay. Dieser Server besitzt noch verfügbare Kapazitäten. Die Zuteilung von Peer p zu diesem Server erfolgt, wodurch der Peer in den Zustand *Unterstützt* übergeht.
3. Es befinden sich aktive Supporter im Overlay. Diese Supporter haben keine verfügbaren Kapazitäten. Peer p verweilt aus diesem Grund weiterhin im Zustand *Unterversorgt*.

Der zugewiesene Supporter erhält eine Liste aller Peers, die er mit Datenblöcken versorgen soll von der Überwachungskomponente. Alle anderen Peers, mit denen der Supporter in Kontakt steht, erhalten keine Datenblöcke.

Sobald sich Peer p erholt hat, sendet er eine Versorgt-Nachricht an den Überwachungsmechanismus. Der Mechanismus registriert dies, gewährt Peer p allerdings noch eine Schonzeit t_{erholt}, bevor dieser zurück in den Zustand *Normal* wechselt und somit keine Unterstützung mehr bekommt. Ist der Zustandswechsel vollzogen, so wird der zuvor zugewiesene Supporter darüber informiert, dass er Peer p nicht länger unterstützen soll.

2.5 Kombination von Give-to-Get und adaptiver Server-Allokation

Abbildung 2.7 zeigt ein vereinfachtes Modell einer hybriden Peer-to-Peer Video-on-Demand Systemarchitektur in Bezug auf die Verteilung *eines* Inhalts. Die farbig markierten Blockstrukturen über den Teilnehmern symbolisieren die dort verfügbaren Chunks. Grün kodiert die Verfügbarkeit, rot die Abwesenheit eines spezifischen Chunks. Dieses Modell bildet die Grundlage für die Systemarchitektur des in dieser Arbeit vorgestellten Mechanismus zum adaptiven Bandbreitenmanagement. Es vereint die Vorzüge einer Client-/Server-basierten Lösung mit denen einer Peer-to-Peer basierten Verteilstrategie und versucht, deren beider Nachteile zu kompensieren. Für die vorliegende Arbeit ist insbesondere die Kombination eines G2G-fähigen P2P-Systems (vgl. Abschnitt 2.3.3) mit der Supporter-Strategie (vgl. Abschnitt 2.4) von Bedeutung.

In diesem Szenario besitzt lediglich der Server den Inhalt komplett. Das Versenden von Chunks erfolgt über die Verbindungen zwischen den Teilnehmern und dem Server. Den primären Distributionsweg stellt ein P2P-System, das sich über den Teilnehmern beider ISPs bildet. Durch die Beschaffenheit des P2P-Systems entstehen kostenintensive Inter-ISP Verbindungen zwischen ISP A und ISP B. Der Einsatz lokalitätsfördernder Mechanismen [7, 8] kann die

Beschränkung auf lokale Verbindungen fördern und somit Kosten für den ISP einsparen. Sofern der Schwarm den Peer-Bedarf selbst nicht hinreichend bedienen kann und ein Performanz-Verlust droht, kann die Server-Komponente einspringen, um Peers mit den entsprechenden Inhalten zu versorgen (sekundärer Distributionsweg). Dies ist bspw. dann der Fall, wenn ein Peer den Chunk, der zur unmittelbaren Wiedergabe des Video-Inhalts benötigt wird, nicht lokal vorliegen hat und nicht über den Schwarm beziehen kann.

3 Verwandte Arbeiten

Dieses Kapitel beherbergt eine Taxonomie der zentralen Arbeiten, die sich mit der Reduktion von Inter-ISP-Datenverkehr beschäftigen. Die Arbeiten lassen sich grob in drei Haupttaxa einteilen: Traffic Shaping, Bereitstellung dedizierter Hardware und Locality Aware Peer Selection. Die Unterteilung in diese Haupttaxa folgt der Notwendigkeit, eine möglichst gute Abgrenzung über den diskutierten Arbeiten zu erzielen. Sie erhebt nicht den Anspruch auf Vollständigkeit.

3.1 Traffic Shaping

Unter Traffic Shaping versteht man die Manipulation von ein- und ausgehendem Datenverkehr, so dass dieser einem gewünschten Modell entspricht. Die Ziele des Traffic Shaping können dabei vielfältig sein und reichen von einer Begünstigung bestimmter Verkehrsklassen bis hin zur simplen Ratenlimitierung oder Verbindungsblockierung. Letztgenannte Mechanismen sind insbesondere im Kontext dieser Arbeit interessant, da sie von einigen Netzanbietern [29, 30] genutzt werden, um unerwünschten Datenverkehr zu limitieren bzw. zu eliminieren. Eine Aufstellung von Netzanbietern, die solche Methoden benutzen, findet der interessierte Leser unter [31]. Die technologischen Mittel hierfür bringt die Netzwerkhardware diverser Anbieter bereits mit [32, 33, 34]. Die konkrete Technik, um Verbindungen zu blockieren, ist sehr simpel. Im Falle des Netzanbieters Comcast hat man bspw. herausgefunden, dass durch in das lokale ISP-Netz injizierte TCP Reset Pakete sich dafür verantwortlich zeigten, Verbindungen zwischen bestimmten Applikationsprotokollen zu terminieren [35]. Für den Endbenutzer ist dieses Vorgehen nur schwer ersichtlich. Software-Werkzeuge, wie bspw. BTTest [36], schaffen hier Abhilfe. Durch sie ist ein Endbenutzer in der Lage, seinen Netzanbieter auf komfortable Weise bezüglich der Blockierung von BitTorrent-Verbindungen zu überprüfen.

Derartige Praktiken verfolgen nicht das eigentliche Ziel der Reduktion von Inter-ISP-Datenverkehr, sondern sorgen lediglich dafür, dass der Endbenutzer höhere Downloadzeiten in Kauf nehmen muss, oder aber die Peer-to-Peer Software gar nicht benutzen kann. Sie verletzen somit das Prinzip der *Netzneutralität*, welches ursprünglich von Tim Berners-Lee, dem Erfinder des World Wide Web, folgendermaßen definiert wurde [4]:

> *If I pay to connect to the Net with a certain quality of service, and you pay to connect with that or greater quality of service, then we can communicate at that level.*

Das Prinzip der Netzneutralität besagt also nicht anderes, als dass die Netzanbieter die Gleichbehandlung der Daten ihrer Benutzer gewährleisten sollen. Techniken, wie bspw. Ratenlimitierung oder Verbindungsblockierung verletzen dieses Prinzip, da der Kunde aktiv an der Partizipation im Internet gehindert wird – wohlgemerkt für einen Dienst, den er bezahlt. Dass man sich bei der Einhaltung des Prinzips nicht auf die Kräfte des Marktes verlassen kann, zeigen Netzanbieter wie Comcast eindrucksvoll.

Dass man sich auch intelligenten Mitteln bedienen kann, um den ein- und ausgehenden Datenverkehr zu optimieren, zeigt die Arbeit von Wang et al. [37]. Das dort verwendete MESH-System ist an die Architektur eines BitTorrent-Systems (vgl. Abschnitt 2.2) angelehnt, zeigt aber hybriden Charakter, da es als sekundären Distributionspfad Server einsetzt. Damit Netzanbieter nicht auf die oben genannten Methoden zurückgreifen, ist es laut Wang et al. wichtig, eine Kooperation zwischen Netzanbieter und Inhaltsanbieter zu etablieren. Eine solche Kooperation fußt auf wirtschaftlichen als auch technologischen Aspekten. Die Idee hinter der Arbeit von Wang et al. ist, einen feingranularen Scheduling-Algorithmus zu entwickeln, der sich durch eine explizite Allokation von Übertragungsraten zu jedem Zeitpunkt den Gegebenheiten des Overlays anpassen kann, ohne dabei bestimmte Optimierungskriterien zu vernachlässigen. Diese Optimierungsziele sind:

- Reduktion der Server-Last (primär)
- Reduktion von Inter-ISP-Datenverkehr (sekundär)
- Maximierung von Prefetching (tertiär)

Als Basisvergleich betrachten Wang et al. die gleichmäßige Aufteilung von Übertragungsraten bei multiplen TCP-Verbindungen durch TCP Congestion Control. Die Autoren zeigen in der Arbeit, dass der von ihnen entwickelte Scheduling-Algorithmus auf Paket-Ebene günstiger arbeitet als TCP Congestion Control. Insbesondere in einer Topologie, die durch Locality Aware Peer Selection (vgl. Abschnitt 3.3) geformt wird, erreicht der Scheduling-Algorithmus eine weitere Reduktion von Inter-ISP-Datenverkehr um bis zu weitere 40%, ohne dabei die Server-Last zu erhöhen.

Streng genommen verletzt ein derartiger Mechanismus ebenfalls das Prinzip der Netzneutralität. Allerdings muss man die Zielsetzung im Vergleich zu den naiven Verfahren, wie bspw. der Ratenlimitierung, diesbzgl. unterscheiden. So hat die Arbeit von Wang et al. [37] gezeigt, dass der Datenverkehr durch intelligente Mechanismen so geformt werden kann, dass die Allgemeinheit der Peers innerhalb eines Peer-to-Peer Systems davon profitieren kann.

3.2 Bereitstellung dedizierter Hardware

Der Netzanbieter kann durch die Bereitstellung zusätzlicher Hardware dafür sorgen, Inter-ISP-Datenverkehr einzusparen. Grundsätzlich können die in der Literatur bekannten Verfahren entweder Caches oder Gateway Peers zugeordnet werden. Dieser Abschnitt diskutiert die Funktionsweise, Möglichkeiten und Probleme ausgewählter Arbeiten aus diesem Forschungsfeld.

3.2.1 Cache-gestützte Lösungen

Cache-gestützte Lösungen finden Anwendung in vielen Bereichen des Internets. Eine Untersuchung auf die Anwendbarkeit in der Peer-to-Peer basierten Verteilung von Daten ist daher naheliegend und Gegenstand zahlreicher Arbeiten in diesem Feld. Caches können sich hinsichtlich ihrer konkreten Funktionsweise und Rolle im Netzwerk unterscheiden. In [38] identifizieren Lehrieder et al. unterschiedliche Cache-Typen und untersuchen diese hinsichtlich ihres globalen Einflusses in einem BitTorrent-ähnlichen Peer-to-Peer System (vgl. Abschnitt 3.3). Diese Cache-Typen sind:

- **Transparente Caches**: Die Anfragen, die ein ISP-lokaler Peer an einen Peer aus einer externen Domäne richtet, werden an der Grenze des lokalen ISP-Netzes abgefangen und mit Hilfe von Deep Packet Inspection untersucht. Mit Hilfe dieser Technik kann ein transparenter Cache ermitteln, ob er die gewünschten Inhalte selbst verwaltet. Ist dies der Fall, so beantwortet er die Anfrage des Peers selbst und leitet die gewünschten Daten an selbigen. Dies reduziert die Notwendigkeit für Inter-ISP-Verbindungen und damit auch unerwünschten, *eingehenden* Inter-ISP-Datenverkehr.
- **ISP-verwaltete Ultrapeers**: Diese Caches sprechen das Protokoll des eingesetzten Peer-to-Peer Systems und agieren als Peers innerhalb des Systems. Der Unterschied zu regulären Peers besteht darin, dass ihre Upload-Kapazität deutlich erhöht ist. Durch die Teilnahme innerhalb des Peer-to-Peer Systems ist die Notwendigkeit zur Deep Packet Inspection nicht mehr gegeben. Für reguläre Peers arbeiten diese Caches transparent. Das bedeutet, dass ein regulärer Peer nicht weiß, ob ein Tauschpartner ein Cache ist oder nicht. Konsequenterweise heißt dies natürlich auch, dass die Effizienz der Caches von der Wahl der Nachbarn abhängig ist. Ein regulärer Peer, der keinen ISP-verwalteten Ultrapeer in seiner Nachbarschaftsliste vorfindet, wird nicht in die Vorzüge einer Cache-basierten Verteilung gelangen.
- **ISP-verwaltete Caches**: Diese Caches sprechen ebenfalls das Protokoll des eingesetzten Peer-to-Peer Systems. Der Unterschied zu den ISP-verwalteten Ultrapeers besteht in erster Linie darin, dass sie für reguläre Peers nicht transparent arbeiten, sondern ihre Rolle explizit innerhalb des Peer-to-Peer Systems bekannt ist. Dies gibt Peers die Möglichkeit, selbst zu entscheiden, ob sie den gewünschten Inhalt von einem Cache oder über andere, reguläre Peers beziehen möchten. Bei Einsatz dieses Verfahrens besteht jedoch die Notwendigkeit eines Entdeckungsmechanismus, so dass Peers gezielt Adressen von ISP-verwalteten Caches in ihrer Nachbarschaft ermitteln können. Die Application Layer Traffic Optimization (ALTO) Arbeitsgruppe arbeitet aktuell an einer Standardisierung für einen derartigen Mechanismus.

Unabhängig von der eingesetzten Cache-Variante müssen Caches dafür sorgen, Inhalte mit bestimmten Merkmalen vorrätig zu halten, da ansonsten ihre Effektivität sinkt. Typischerweise beschränken sich Caches auf populäre Inhalte, jedoch hat [39] gezeigt, dass eine koordinierte Cache-Allokation für weniger populäre Inhalte durchaus

die globale Effizienz einer Peer-to-Peer basierten Verteilstrategie für Video-Inhalte steigern kann. Inwiefern eine solche Cache-Allokation eine Topologie begünstigen kann, die durch Locality Aware Peer Selection geformt wurde, ist nach Wissen des Autors derzeit noch unklar. Die Literatur bietet allerdings auch kooperative Caching-Strategien, die explizit vor dem Hintergrund der Reduktion von Inter-ISP-Datenverkehr entwickelt wurden [40].

Es existieren einige grundsätzliche Probleme mit Cache-gestützten Verfahren, die den Einsatz in realen Szenarien deutlich erschweren. Caches arbeiten protokollabhängig. Dies erfordert die Realisierung einer entsprechenden Caching-Lösung für ein spezifisches Peer-to-Peer System. Des Weiteren besitzen Caches keine semantischen Informationen zu den Inhalten, die sie speichern. Dies ist vor Allem ein Problem bezüglich der Distribution von Daten, die einem Copyright unterliegen oder nur in bestimmten geographischen Zonen legal vertrieben werden dürfen. Lehrieder et al. behaupten in ihrer Arbeit [38], dass der Einsatz eines transparenten Caches, der die Transferrate des entfernten Peers, an den eine Anfrage gerichtet ist, emuliert, Legalitätsprobleme beseitigt. Die Autoren rechtfertigen diese Aussage damit, dass der Cache keine zusätzlichen Ressourcen für die Verbreitung illegaler Inhalte anbietet. Dies ist eine sehr naive Sichtweise auf die Rechtslage. Transparente Caches sind – auch wenn dies nicht ersichtlich ist – diejenige Entität, die aktiv entsprechende Inhalte speichert und an interessierte Peers im lokalen ISP-Netz ausliefert. Die Probleme mit illegalen Inhalten bestehen daher weiterhin. Unabhängig davon muss der Netzanbieter dafür sorgen, dass die notwendige Hardware bereitgestellt wird. Dies verursacht nicht nur Anschaffungskosten, sondern auch operative Kosten für die fortwährende Bereitstellung und Instandhaltung der Caches.

Alternativ zu klassischen Cache-Typen zeigt [41] einen Ansatz auf, der am ehesten als konkrete Realisierung eines ISP-verwalteten Caches zu betrachten ist. Die Terminologie unterscheidet sich leicht von der in [38], so dass nicht von ISP-verwalteten Caches die Rede ist, sondern von *ISP-owned Peers* (kurz IoP). Die Studie zeigt durch Simulation, dass der Einsatz von IoPs durchaus in der Lage ist, für eine Reduktion von Inter-ISP-Datenverkehr zu sorgen. Die Kombination des IoP mit einem Mechanismus zur Lokalitätsförderung (vgl. Abschnitt 3.3) birgt weiteres Potential für derartige Einsparungen. IoPs sind zwar der zentrale Mechanismus in [41], allerdings untersucht die Arbeit ebenfalls die potentiellen Vorzüge von sogenannten *hoch aktiven Peers* (kurz HAP). Ein HAP erhält von seinem Netzanbieter ein erhöhtes Zugangsprofil, in der Annahme, die gesteigerte Kapazität anderen Peers zugute kommen zu lassen. Grundsätzlich unterscheidet sich die Arbeitsweise eines HAP nicht von der des IoP. Das Problem der Weiterverbreitung illegaler Inhalte durch Hardware des Netzanbieters entfällt allerdings, da der HAP nicht dem Verantwortungsbereich des Netzanbieters untersteht. Mögliche Inferenzmechanismen zur Beförderung regulärer Peers zu HAPs sind nicht Gegenstand von [41]. Antwortmöglichkeiten hierzu liefert jedoch [42] durch eine Reihe von Identifikationsmetriken, die bis dato nur simulativ erprobt wurden.

3.2.2 Gateway Peers

Gateway Peers sind ebenfalls vom Netzanbieter speziell bereitgestellte Knoten, die an den Grenzen eines ISP-Netzes lokalisiert sind und den aus- und eingehenden Datenverkehr regulieren. Reguläre Peers, die sich in einer Domäne befinden, die von einem Gateway Peer unterstützt wird, dürfen sich nur untereinander – also im selben ISP-Netz – oder aber mit dem Gateway Peer verbinden. Peering-Beziehungen in externe Domänen sind nicht zulässig. Der Gateway Peer selbst darf Verbindungen in externe Domänen unterhalten, so dass ISP-lokale Peers die Möglichkeit besitzen, über den Gateway Peer mit externen Peers in Kontakt zu treten. Der Gateway Peer kann in diesem Szenario als Regulator dienen, da er sehr genau verfolgen kann, ob Datenverkehr über externe Verbindungen *redundant* in das lokale ISP-Netz eingeführt wird.

Bindal et al. diskutieren in [7] die fundamentalen Schwächen dieses Ansatzes im Kontext eines BitTorrent-ähnlichen Systems. Dimensioniert man den Gateway Peer zu klein, so dass sich seine Netzwerkanbindung nicht von der regulärer Peers unterscheidet, so erhöht sich die Download-Zeit signifikant. Dimensioniert man den Gateway Peer dagegen angemessen, so erscheint er Peers aus externen Domänen, die nicht von einem Gateway Peer unterstützt werden, attraktiver. Für den ISP-lokalen Gateway Peer erscheinen externe Peers ebenfalls attraktiver, da ISP-lokale Peers in der Regel keine Inhalte an den Gateway Peer liefern, so dass das Verbindungsmanagement zugunsten anderer Netzanbieter manipuliert wird. Die grundsätzlichen Probleme zum Thema Illegalität von Downloads sind von den Caches auf Gateway Peers übertragbar.

3.3 Locality Aware Peer Selection

Das Verbindungsmanagement in Peer-to-Peer Systemen arbeitet auf Applikationsebene und bezieht üblicherweise das Wissen aus der unterliegenden Netzwerkschicht nicht bei der Suche nach neuen Nachbarn mit ein. Durch dieses Verhalten können potentiell weitreichende Verbindungen auf Netzebene entstehen, deren Unterhaltung für den Netzanbieter kostenintensiv ist (vgl. Abschnitt 2.1). Auf die Probleme einer mangelnden Kooperation zwischen Protokollen auf Applikationsebene und Protokollen der Netzwerkschicht weisen Keralapura et al. exemplarisch in [43] hin. Die Synthese dieser Arbeit ist auf Peer-to-Peer Systeme übertragbar. Hat ein Peer Wissen über die Lokalität anderer Peers, so kann er *bevorzugt* lokale Ressourcen benutzen.

In [6] zeigen Karagiannis et al. erstmals, dass sich existierende Peer-to-Peer Lösungen[1] negativ für den ISP verhalten können. Die Arbeit hebt insbesondere BitTorrent hervor, welches durch die Tit-for-Tat-Strategie ein großes Volumen an Inter-ISP-Datenverkehr erzeugt. Dass dies nicht mal nötig ist, zeigt die Studie ebenfalls. Peers laden 70% bis 90% der Inhalte, die andere Peers im selben ISP-Netz anbieten, über Inter-ISP-Verbindungen herunter. Karagiannis et al. zeigen mit ihrer Studie, dass lokalitätsfördernde Mechanismen ein enormes Potential haben. Die Autoren skizzieren die Funktionsweise eines entsprechenden Ansatz grob, untersuchen diese Lösung in ihrer Arbeit allerdings nicht im Detail.

Die Bedeutung der Lokalisierung von P2P-Datenverkehr ist der IETF (Internet Engineering Task Force) ebenfalls bekannt. Die ALTO (Application Level Traffic Optimization) Gruppe des IETF untersucht die Möglichkeiten einer Standardisierung der im Folgenden diskutierten Ansätze durch ein gemeinsames Rahmenwerk [44].

Einen konkreten Vorschlag zur Umsetzung der Lokalisierung von P2P-Datenverkehr liefern Bindal et al. in [7] anhand von BitTorrent. Die Autoren haben erkannt, dass das fundamentale Problem von BitTorrent an der zufälligen Auswahl von Peers durch den Tracker liegt. Peers eines lokalen ISP laden ein und denselben Chunk öfter über Inter-ISP-Verbindungen herunter, als dies nötig wäre. Bindal et al. schlagen als Lösungsansatz Biased Neighbour Selection (BNS) vor. Die Idee dahinter ist, die Tracker-seitige Peer-Auswahl so zu manipulieren, dass die Tracker-Antwort anteilmäßig mehr Peers aus der ISP-Domäne des anfragenden Peers enthält. Der anfragende Peer erhält somit eine größere Auswahl ISP-lokaler Tauschpartner, kann aber nach wie vor über einige wenige ISP-übergreifende Verbindungen Chunks beziehen, die Peers aus der gleichen ISP-Domäne nicht anbieten können. Das Ergebnis der Studie ist, dass sich BitTorrent durch den Einsatz von BNS weiterhin nahezu optimal verhält, insbesondere dann, wenn der initiale Seeder eine sehr hohe Bandbreite hat und BitTorrent auf die Blockselektionsstrategie Rarest First Replication zurückgreift. Die Arbeit schlägt ferner Kombinationsmöglichkeiten mit einem Mechanismus zur Ratenlimitierung vor, speziell in einem Szenario, in dem Peers aus entfernten ISP-Domänen grundsätzlich eine sehr hohe Bandbreite im Vergleich zu den Peers aus der gleichen ISP-Domäne haben. Die Implementierung von BNS kann laut Bindal et al. auf zwei unterschiedliche Arten erfolgen: Zum Einen über die Modifikation des Trackers und der P2P-Client-Software. Zum Anderen über P2P Traffic Shaping Endgeräte, die Tracker-Antworten abfangen und entsprechend manipulieren. Letzterer Ansatz ist insbesondere für ISPs interessant, da eine Modifikation des P2P-Systems nicht erforderlich ist.

Oechsner et al. haben mit Biased Unchoking (BU) die Idee hinter BNS auf den Mechanismus des Optimistic Unchoke übertragen [8]. BU erfordert die Modifikation des Choke-Algorithmus einer auf BitTorrent basierenden P2P-Lösung. Die reguläre Implementierung des Choke-Algorithmus sieht Optimistic Unchokes (vgl. Abschnitt 2.2.3) über allen n interessierten und derzeit blockierten Peers mit einer Wahrscheinlichkeit von $1/n$ vor. BU modifiziert dieses Verhalten. Die Wahrscheinlichkeit, dass ein Peer den Optimistic Unchoke erhält, hängt von dem Lokalitätswert $L(x, y)$ der potentiellen Verbindung zwischen Peer x und Peer y ab. Anhand der konkreten Werte der Funktion $L(x, y)$ teilt BU Nachbarn bewerteter Peer-Kombinationen in präferierte und nicht präferierte Verbindungspartner ein. Mit einer Wahrscheinlichkeit von q erhält ein Peer aus der Menge der präferierten Verbindungspartner den Optimistic Unchoke. Enthält eine der beiden Mengen keine Peers, so wählt BU Verbindungspartner aus der jeweils komplementären Menge. Das Verfahren führt dazu, dass Peers mit einem hohen Lokalitätswert häufiger den Optimistic Unchoke erhalten, als andere. Die Arbeit definiert Lokalität durch die Anzahl an AS-Hops zwischen zwei Peers x und y. Die Ergebnisse in [8] zeigen, dass BU am besten in Szenarien arbeitet, in denen hoher Peer-Bedarf herrscht. Der alleinige Einsatz von BU erreicht bereits eine Reduktion des Inter-ISP-Datenverkehrs. Die Kombination von BNS mit BU liefert jedoch die besten Ergebnisse. Das ist nicht weiter verwunderlich, da BNS die Peers vorschlägt,

[1] Die Arbeit ist im Jahre 2005 erschienen. Die Aussage bezieht sich auf die Systeme, die zu diesem Zeitpunkt geläufig gewesen sind.

die von BU bevorzugt den Optimistic Unchoke erhalten. Die Ergebnisse zeigen auch, dass bereits ein Gewinn durch den Einsatz von BNS und BU zu erzielen ist, wenn nur ein Teil der Peers eines Schwarms diese Verfahren nutzt.

3.3.1 Informationsserver

Sowohl BNS als auch BU gehen davon aus, dass die notwendigen Lokalitätsinformationen vorliegen. In [7] liefern Bindal et al. eine Reihe konkreter Möglichkeiten, um entsprechende Informationen zu beziehen, untersuchen diese jedoch nicht näher. Grundsätzlich kann die Informationsquelle verschiedenartig sein. Die Umsetzung von BU erfolgt bspw. client-seitig, so dass sich für die Informationsbeschaffung auch GeoIP-Datenbanken [45] eignen würden. Eine solche Lösung ist natürlich auf die Feingranularität dieser Datenbanken beschränkt und und kann bzgl. der Exaktheit deutlich hinter komplexeren Messtechniken zurückliegen. Dedizierte Informationsserver [46, 47, 48], die einen Bewertungsdienst zur Verfügung stellen, können von Peers oder im Falle von BNS auch durch den Tracker kontaktiert werden, um entsprechende Informationen zu erhalten. Der Vorteil ist, dass ein Informationsserver komplexe Mechanismen umsetzen kann, auf die ein Peer zurückgreifen kann, ohne konkretes Wissen darüber haben zu müssen. Die Integration in das Gesamtsystem ist relativ einfach. Kontaktiert ein Peer den Informationsserver, so übergibt er diesem eine Reihe von potentiellen Peers, mit denen er in Verbindung treten möchte. Der Informationsserver weist diesen Peers einen Gütewert zu, der anhand unterschiedlicher Metriken [46] berechnet werden kann und sendet die Liste bewerteter Peers zurück an den anfragenden Peer. Der Peer kann die dadurch gewonnen Information dazu nutzen, um sein lokales Verbindungsmanagement zu beeinflussen, indem er Verbindungen nur zu denjenigen Peers zulässt, die einen hohen Gütewert aufweisen.

Aggarwal et al. schlagen in [46] einen konkreten Mechanismus vor. Es handelt sich um einen ISP-betriebenen Orakel-Dienst, den P2P-Clients kontaktieren können, um Gütewerte zu erhalten. Prinzipiell greift die Arbeit die Idee hinter BNS auf, generalisiert sie aber soweit, dass verschiedene Kriterien Einfluss auf die Bewertung der Peers nehmen können. So ist es bspw. möglich, die Anzahl der AS-Hops zwischen einem Peer x und einem Peer y heranzuziehen (in Analogie zu BNS [7]), oder aber konkret die Leistung der Peers zu berücksichtigen. Die Bewertung kann entlang mehrerer Kriterien erfolgen. Denkbar ist eine initiale Sortierung von Peers nach deren Lokalität, wobei Peers mit hohem Lokalitätswert nachfolgend nach Performanz-Aspekten zusätzlich einem Ranking unterzogen werden. Die Ergebnisse der Arbeit zeigen eine Verbesserung hinsichtlich der Lokalität von P2P-Datenverkehr, ohne dabei bestimmte Eigenschaften der Overlay Topologie, wie bspw. mittlere Pfadlänge oder Verbindungsgrad eines Knotens, negativ zu beeinflussen. Insbesondere für ISPs ist dieser Mechanismus interessant, da sie eigene Ziele zur Handhabung von P2P-Datenverkehr durch geschickte Bewertungsstrategien umsetzen können. Der Erfolg des Verfahrens ist jedoch entscheidend davon abhängig, ob eine Vertrauensbasis zwischen ISP und Content Provider besteht, da sich ansonsten die angestrebte kooperative Beziehung zwischen ISP und P2P-System vermutlich nicht einstellt.

Einen ähnlich motivierten Ansatz verfolgt das Projekt Provider Portal for Applications (P4P) [47]. P4P basiert ebenfalls auf der Annahme einer Kooperation von ISP und P2P-System. Die Systemarchitektur von P4P sieht im Wesentlichen zwei tragende Entitäten, iTracker und p-distance, vor. Der ISP verwaltet einen sog. iTracker, der als Informationsserver dient und verschiedene Schnittstellen implementieren kann. Peers können über diese Schnittstellen Informationen über den Zustand des Netzwerks beziehen. Der Vorteil des iTrackers ist seine Flexibilität. Die Implementierung ist nicht auf die Kommunikation mit einer bestimmten P2P-Anwendung beschränkt, sondern liefert Netzwerkinformationen unabhängig eines bestimmten P2P-Protokolls aus. In Tracker-basierten P2P-Systemen kann der Tracker selbst mit dem iTracker kommunizieren, um bspw. das nötige Wissen zur Umsetzung einer BNS zu erhalten. In Tracker-losen Systemen haben Peers die Möglichkeit, den iTracker selbst zu kontaktieren und die Ergebnisse entsprechend auszuwerten. Der Begriff p-distance stellt die zweite wesentliche Entität von P4P dar. p-distance beschreibt im Wesentlichen die Distanz zwischen zwei Peers. Die Berechnung der p-distance erfolgt durch den ISP selbst. Je nach Strategie des ISPs kann sich die Berechnung unterschiedlicher Metriken bedienen. Da der ISP konkretes Wissen über den Zustand des Netzwerks hat, können bspw. Routing-Informationen nach OSPF-Gewichtungen (Open Shortest Path First) oder BGP-Präferenzen (Border Gateway Protocol), aber auch Überlastsituationen auf gewissen Links oder Inter-ISP-Linkkosten in die Distanzberechnung miteinfließen. Die Evaluation des Ansatzes umfasst Simulationen und eine Erprobung innerhalb eines PlanetLab-Testbeds. Die Ergebnisse zeigen, dass P4P durchaus in der Lage ist, die Performanz des Systems zu erhöhen und dabei gleichzeitig Inter-ISP-Verbindungen zu reduzieren. Diese positiven Ergebnisse hat eine Feldstudie des ISP Comcast bestätigt [49].

Eine Systemarchitektur, die ähnlich der bereits erwähnten Informationsserver arbeitet, ist Gegenstand von Forschungs- und Entwicklungsarbeit im Rahmen des SmoothIT Projekts [50]. Die zentrale Komponente des Systems stellt der SmoothIT Information Service (SIS) dar [48], der ebenfalls unter der Verwaltung des ISP steht. Auf konzeptioneller Ebene arbeitet der SIS ähnlich wie die bereits diskutierten Ansätze [46, 47]. Nach Soursos et al. [48] kann der potentielle Gewinn eines lokalitätsfördernden Mechanismus auf Grund der hohen Dynamik von P2P-Systemen über die Zeit variieren. Dies ist einer der wesentlichen Unterschiede zu [47], denn der SIS forciert den Bewertungsmechanismus nicht, sondern bietet selbigen als Zusatzoption an. Der Benutzer kann demzufolge selbst entscheiden, ob er zu einem gegebenen Zeitpunkt den Mechanismus nutzen möchte oder nicht. Als konkrete Bewertungsstrategie inkorporiert der SIS die BGP-basierte nach Racz et al. [51]. Die Bewertungen des SIS können auf Seiten der P2P-Applikation genutzt werden, um Mechanismen wie BNS und BU mit den nötigen Informationen zu versorgen. Dass sich BGP-basierte Gütewerte zur Förderung der Lokalität bzgl. P2P-Verbindungen eignen, zeigen Simulationsergebnisse in [51].

3.3.2 Client-basierte Ansätze

Die Umsetzung eines lokalitätsfördernden Mechanismus erfordert nicht notwendigerweise eine zentrale Komponente. So ist bspw. die Ono-Software [52] eine Lösung, die Informationsbeschaffung und Peer-Auswahl auf Seiten des P2P-Clients durchführt. Ono basiert auf der Beobachtung, dass ein CDN einen anfragenden Client mittels DNS Redirektion an einen CDN Replika Server weiterleitet. DNS Redirektionen erfolgen primär anhand der Latenz [12], so dass Choffnes et al. die Schlussfolgerung ziehen, dass sich Peers mit ähnlichem Redirektionsverhalten mit hoher Wahrscheinlichkeit *in Nähe* zueinander befinden. Ein Ono-unterstützter Peer sendet mehrfach Anfragen an populäre CDNs. Anhand der Antworten berechnet er einen Vektor, welche die Häufigkeit der Redirektion zu den CDN Replika Servern inkorporiert. Peers tauschen diese Vektoren untereinander aus. Über das Kosinusähnlichkeitsmaß kann ein Peer herausfinden, ob sein eigener Vektor dem eines anderen Peers ähnelt und somit die Peer-Selektion explizit steuern. Die Ergebnisse der Arbeit zeigen, dass eine signifikante Reduktion hinsichtlich der AS-Pfadlängen durch die Ono-Software erreicht wird. Jedoch fallen AS-Grenzen nicht notwendigerweise mit ISP-Grenzen zusammen. Umspannt die ISP-Domäne mehrere AS, so arbeitet die Peer-Auswahl von Ono möglicherweise zu grobgranular und verweigert Verbindungen zu Peers, die vielleicht eine erhöhte Latenz zeigen, aber immernoch innerhalb der ISP-Domäne liegen würden. Der Ansatz ist daher als *Heuristik* zu verstehen, da er keine Garantie dafür liefert, Verbindungen zwischen Peers innerhalb der gleichen ISP-Domäne zu fördern.

3.4 Diskussion

Viele der vorgestellten Ansätze erreichen zwar eine Reduktion des Inter-ISP-Datenverkehrs, skalieren aber mitunter nicht gut (vgl. Abschnitt 3.2.1) oder lassen die Interessen von Benutzern und Content Providern außer Acht. Piatek et al. fassen in ihrer Studie [53] einige Problemfelder zusammen. Problematisch sind insbesondere artifiziell konstruierte Evaluationsszenarien, die idealisierte Voraussetzungen haben und nicht auf einen realen Ansatz des entsprechenden Mechanismus generalisierbar sind. Tatsächlich zeigen Piatek et al. auf, dass das Ono-System (vgl. Abschnitt 3.3.2) im realen Einsatz den Inter-ISP-Datenverkehr um weniger als 1% bei Zugang über einen großen ISP reduziert. Die Studie zeigt ferner, dass der generelle Einfluss lokalitätsfördernder Mechanismen hinter den Erwartungen zurückbleibt, da ein Peer in den meisten Fällen zuwenige Tauschpartner findet, die sich in derselben ISP-Domäne befinden. Piatek et al. unterstützen diese Behauptungen durch umfassende Simulationen auf aggregierten, realen Peer-Daten. Hohlfeld et al. gelangen in [54] zu einem ähnlichen Schluss und zeigen, dass die Anzahl von AS signifikant mit der Größe eines Schwarms steigt. Dies hat zur Folge, dass sich in mehr als 90% der untersuchten AS weniger als 10 Peers gemeinsam in einem zufällig gewählten Schwarm befinden.

Ein weiteres Problem ist sicherlich die Abnahme der Robustheit eines P2P-Systems, wenn das Overlay aufgrund zu viel Lokalität zu einem lokalen Cluster degradiert. In [55] zeigen Blond et al., ab wann sich zuviel Lokalität negativ auf die strukturellen Eigenschaften des Overlays auswirken.

Generell ist es von Bedeutung, dass entsprechende Ansätze nicht einseitige Interessen bedienen, sondern versuchen, die Problemstellung in ihrer Gesamtheit zu erfassen. Insbesondere die Arbeiten zu P4P [47] und dem SmoothIT-Projekt [50] zeigen Ansätze auf, von der alle beteiligten Interessengruppen profitieren können.

4 Adaptives Bandbreitenmanagement

Um den Bedürfnissen von ISPs, Content Providern und den Benutzern eines P2P-basierten Video-on-Demand Systems gerecht zu werden, muss eine Optimierung entlang divergierender Ziele erfolgen. Im Interesse des ISPs ist es, kostenintensiven Inter-ISP-Datenverkehr auf ein notwendiges Minimum zu reduzieren. Content Provider sind im Wesentlichen daran interessiert, die Last von ihren Servern zunehmen, so dass diese lediglich als Ausweichlösung dienen, wenn das P2P-System den Bedarf seiner Nutzer nicht alleine decken kann. Der Anspruch der Benutzer liegt schließlich im Erfüllen einer gewissen Dienstgüte, so dass eine unterbrechungsfreie Wiedergabe gewünschter Video-Inhalte möglich ist. In Kapitel 3 hat man bereits gesehen, dass existierende Lösungen diese Ansprüche unzureichend erfüllen, da sie selten alle Ziele im Design entsprechender Mechanismen berücksichtigen. Eine Kompromisslösung zwischen der Optimierung der Ziele beteiligter Interessengruppen ist gesucht. Dies erfordert jedoch eine hinreichende Abstimmung zwischen einzelnen Mechanismen. Reduziert man bspw. den Inter-ISP-Datenverkehr, so vermindert man auch potentiell die Verfügbarkeit von Datenblöcken, wodurch im schlechtesten Fall die Mindestanforderungen an die Dienstgüte nicht mehr einzuhalten sind. Für eine konstante Performanz des Overlays ist es daher von entscheidender Bedeutung, im Zweifelsfalle auf Peers zurückgreifen zu können, die sich *nicht* im lokalen ISP-Netz befinden, um so rare Datenblöcke in das Netz einzuführen.

Ein ISP kann adaptives Bandbreitenmanagement als Methode des *Traffic Shaping* einsetzen, um alle beteiligten Parteien bis zu einem gewissen Grad zu begünstigen. Die grundlegende Idee hinter dem Mechanismus ist, ausgewählte Peers mit einer erhöhten Bandbreite auszustatten. Dies fördert verschiedene Aspekte:

- Durch die erhöhte Bandbreite sind ausgewählte Peers in der Lage, als ISP-lokale Server[1] zu arbeiten. Sie können ihre zusätzliche Kapazität dazu nutzen, um andere Peers aus derselben ISP-Domäne mit Daten zu versorgen. Konsequenterweise reduziert sich dadurch die Notwendigkeit, Daten über Inter-ISP-Verbindungen zu beziehen. Im besten Fall ist dies nur noch nötig, wenn rare Datenblöcke in unzureichender Anzahl bei Peers der lokalen ISP-Domäne vorliegen.
- Der Content Provider profitiert ebenfalls davon, denn durch eine erhöhte Upstream-Kapazität ISP-lokaler Peers können dessen Server entlastet werden. Dieser Effekt stellt sich voraussichtlich erst dann ein, wenn viele ISPs den vorgeschlagenen Mechanismus nutzen.
- Der Mechanismus vernachlässigt die Qualitätsansprüche der Benutzer nicht. Durch die erhöhte Upstream-Kapazität einzelner Peers steigt die gesamte Leistungsfähigkeit des Systems. Dies kommt indirekt den Benutzer zu Gute, die nun Downloads schneller beenden bzw. mit einer höheren Transferrate gewünschte Daten beziehen können.

Der Mechanismus kann potentiell die Ziele aller beteiligter Interessengruppen erfüllen. Die negativen Effekte einer einseitigen Lösung kann der Ansatz ebenfalls kompensieren. So kann bspw. eine alleinige Reduktion des Inter-ISP-Datenverkehrs dazu führen, dass das Overlay zu einem lokalen Cluster degeneriert. Das schränkt die Verfügbarkeit von Daten ein. Durch eine erhöhte Bandbreite erscheinen ausgewählte Peers jedoch auch für Peers aus anderen ISP-Domänen attraktiv. Diese Tatsache kann man ausnutzen, um eine kleine Zahl von Inter-ISP-Verbindungen zuzulassen, über die man rare Datenblöcke in die lokale ISP-Domäne einführen kann. An dieser Stelle ist insbesondere die Kombination des Mechanismus mit BNS oder BU (vgl. Abschnitt 3.3) zu berücksichtigen, um zu gewährleisten, dass die erhöhte Upload-Kapazität größtenteils Peers aus der lokalen ISP-Domäne zu Gute kommt.

Abschnitt 4.1 beschreibt die grundlegende Funktion des adaptiven Bandbreitenmanagements und stellt die grundlegenden Herausforderungen, die mit der Realisierung dieses Mechanismus einhergehen, einführend dar. Gegenstand der Abschnitte 4.2, 4.3 und 4.4 ist die detaillierte Diskussion der Funktionsweise. In Abschnitt 4.5 erhält der Leser Lösungsvorschläge zur Umsetzung des adaptiven Bandbreitenmanagements auf Seiten eines ISP. Den Abschluss bildet Abschnitt 4.6 mit der Diskussion von Einschränkungen des Ansatzes bzgl. der Evaluation in einem Live-Testbed.

[1] In manchen Arbeiten findet man auch den Begriff des Super Peers. Ein solcher Peer besitzt i. A. ein wesentlich höheres Zugangsprofil als andere Peers im System.

4.1 Ansatz

Das adaptive Bandbreitenmanagement besteht aus drei wesentlichen Komponenten, die untereinander interagieren.

- **Datenerhebungskomponente**: Für die korrekte Funktionsweise des Mechanismus muss eine Beobachtung aller Peers der lokalen ISP-Domäne erfolgen. Die Datenerhebungskomponente zeichnet sich dafür verantwortlich, entsprechende Statistiken zu aggregieren und zu Verfügung zu stellen.
- **Bewertungskomponente**: Diese Komponente nutzt die aggregierten Statistiken der Datenerhebungskomponente, um Peers einer Bewertung zu unterziehen. Die Bewertung ist ein Maß für die Güte eines Peers.
- **Bandbreitenzuteilung**: Die gemäß ihrer Bewertung besten k Peers erhalten ein erhöhtes Zugangsprofil. Die Aufgabe der Bandbreitenzuteilung ist es, diese Zugangsprofile umzusetzen.

4.1.1 Interaktion der Komponenten

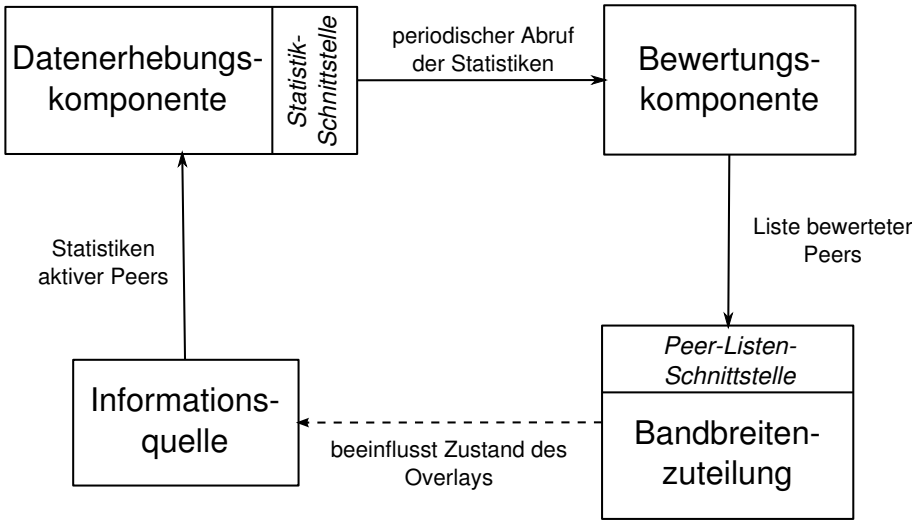


Abbildung 4.1: Die Grafik zeigt das Zusammenspiel der Komponenten untereinander. Die Pfeile geben stets die Richtung des Datenflusses an. Der *gestrichelte* Pfeil zeigt einen indirekten Einfluss auf die Informationsquelle an.

Abbildung 4.1 stellt das Zusammenspiel der Komponenten grafisch dar. Die Datenerhebungskomponente sammelt *permanent* Statistiken über Peers und stellt diese über eine wohldefinierte Schnittstelle zur Verfügung. In regelmäßigen Zeitabständen nutzt die Bewertungskomponente diese Schnittstelle, um die bis zu diesem Zeitpunkt aggregierten Statistiken zu den aktuell sich noch im Schwarm befindlichen Peers zu laden. Die Bewertungskomponente berechnet für jeden aktiven Peer einen Gütewert. Anhand dieses Gütewerts lässt sich die Liste aktiver Peers absteigend sortieren. Die Bandbreitenzuteilungskomponente stellt wiederum eine Schnittstelle zur Verfügung, über die sie die sortierte Peer-Liste erhalten kann. Die Bandbreitenzuteilungskomponente selektiert Peers mit einem hohen Gütewert nach einer bestimmten Strategie und stößt die Umsetzung neu gewählter Zugangsprofile an.

Insbesondere sei darauf hingewiesen, dass das adaptive Bandbreitenmanagement ein *zyklischer Prozess* ist. Die Leistung eines P2P-Systems ist maßgeblich von den individuellen Kapazitäten ihrer Benutzer abhängig. In der Praxis sind Zugangsprofile der Benutzer sehr heterogen und variieren in einem breiten Spektrum. Zusätzlich sorgt das dynamische Verhalten der Benutzer dafür, dass sich der Systemzustand des P2P-Systems zeitabhängig ändert. Die Zuteilung eines erhöhten Zugangsprofils durch das adaptive Bandbreitenmanagement beeinflusst den Zustand des P2P-System ebenfalls. Damit der Mechanismus effizient arbeiten kann, berücksichtigt er diesen Umstand und führt eine Re-Evaluierung der aktiven Peers in festen Zeitabständen durch.

Eine ausführliche Beschreibung der Funktionsweise dieser Komponenten folgt in den nächsten Abschnitten.

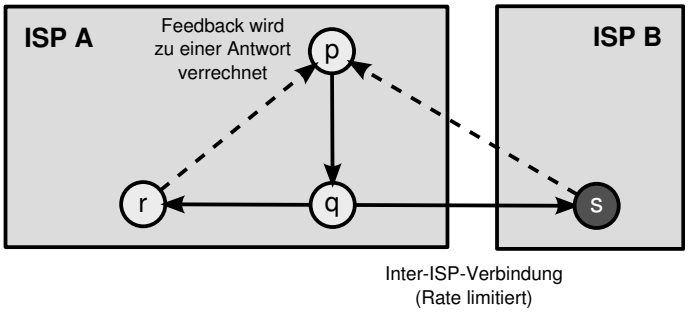


Abbildung 4.2: G2G-Peers holen Feedback von den Nachbarn ihrer Nachbarn (gestrichelte Linie) um zu entscheiden, ob ein Nachbar blockiert werden soll.

4.1.2 Beeinflussung des Zugangsprofils und einhergehende Herausforderungen

Eine Erhöhung der Upload-Kapazität ausgewählter Peers scheint zunächst nicht sehr intuitiv. Unterhält ein Peer eine Vielzahl von Inter-ISP-Verbindungen, so könnte eine Erhöhung seines Zugangsprofils dem ISP eher schaden, als nützen, da er durch die gesteigerte Upload-Kapazität noch mehr Daten über ungünstige Verbindungen senden kann. Für den ISP ist dies natürlich nicht wünschenswert. Die *Identifikation von Peers*, die sich für ein erhöhtes Zugangsprofil eignen, ist daher eine zentrale Fragestellung des Mechanismus. Des Weiteren ist es wichtig, ein Maß für die *Anzahl zu befördernder Peers* zu kennen. Befördert man zuviele Peers, so werden Ressourcen verschwenderisch genutzt. Befördert man hingegen zu wenige Peers, so stellen sich die erhofften Effekte nicht ein. Von zentraler Bedeutung ist ebenfalls die Frage, mit welcher Häufigkeit eine Re-Evaluation – und damit die Promotion ausgewählter Peers – erfolgt. Ist der *zeitliche Abstand* zwischen konsekutiven Re-Evaluationen zu kurz, so könnte sich ein sprunghaftes Verhalten einstellen und die Effizienz des Mechanismus darunter leiden. Ist der zeitliche Abstand dagegen zu groß, so könnte die erwünschte Auswirkung durch die erhöhte Upload-Kapazität verschwindend gering sein. Die nachfolgenden Abschnitte widmen sich diesen und weiteren Fragen.

Intuitiv möchte man sagen, dass eine Limitierung der Kapazitäten bzgl. Inter-ISP-Datenverkehr für *nicht* ausgewählte Peers ebenfalls sinnvoll erscheint. Tatsächlich wäre dies aber kontraproduktiv und könnte sowohl die Leistungsfähigkeit des Systems reduzieren, als auch die Ansprüche der Benutzer an eine gewisse Dienstgüte nicht erfüllen:

- Angenommen, der Mechanismus würde die Upload-Kapazität über Inter-ISP-Verbindungen limitieren (siehe Abbildung 4.2). Seien p, q und r Peers aus derselben ISP-Domäne und s ein Peer aus einer anderen ISP-Domäne. Ferner ist q an Inhalten von p interessiert und s an Inhalten von q. Der Choke-Algorithmus von G2G entscheidet über Feedback der Nachbarn von q darüber, ob die Verbindung zwischen $p \rightarrow q$ blockiert wird. Die Ratenlimitierung auf der Inter-ISP-Verbindung $q \rightarrow s$ sorgt dafür, dass s deutlich weniger Daten erhält, als über ISP-lokale Verbindungen. Das Feedback, dass s an p bzgl. q schickt, fällt daher in Relation zu anderen Verbindungen schlechter aus und könnte die Entscheidung beeinflussen, dass die Verbindung $p \rightarrow q$ blockiert wird. Dies ist kontraproduktiv, da $p \rightarrow q$ eine Verbindung innerhalb der gleichen ISP-Domäne beschreibt und der Mechanismus bestrebt, gerade solche Verbindungen zu fördern. Die Arbeitsweise des G2G-Protokolls verbietet daher eine Limitierung der Upload-Kapazität über Inter-ISP-Verbindungen.
- Angenommen, der Mechanismus würde die Download-Kapazität über Inter-ISP-Verbindungen limitieren. Sei b die Bitrate des gewünschten Video-Inhalts und f der Prefetching-Faktor des G2G-Protokolls. Dann muss ein Peer p eine Mindestdatenrate von $d_{\min} = b \cdot f$ akkumuliert über alle Verbindungen haben, damit eine konstante Wiedergabe des Videos möglich ist. Unter Umständen können Peers aus derselben ISP-Domäne diesen Bedarf nicht stillen. In diesem Fall muss es dem Peer erlaubt sein, die nötigen Daten mit hinreichender Download-Rate über Inter-ISP-Verbindungen zu beziehen, so dass Verluste bzgl. der Dienstgüte nicht auftreten. Limitiert man die Download-Rate über Inter-ISP-Verbindungen, so kann dies unter Umständen nicht erfolgen.

Eine Kombination des adaptiven Bandbreitenmanagements mit lokalitätsfördernden Mechanismen wie BNS und BU (vgl. Abschnitt 3.3) erscheint an dieser Stelle sinnvoll. Durch BNS und BU erfolgt eine Präferierung loka-

ler Verbindungen, sofern sich ISP-lokale Peers anbieten. Bieten sich nicht genügend ISP-lokale Peers an, so kann der Peer Verbindungen mit Peers aus anderen ISP-Domänen eingehen und dadurch einem potentiellen Verlust an Dienstgüte entgegenwirken. Des Weiteren erlaubt die entsprechende Konfiguration dieser Mechanismen es, dass ein Peer einen gewissen Anteil an Inter-ISP-Verbindungen aufrecht erhalten darf, selbst dann, wenn sich genügend ISP-lokale Peers anbieten. Aus diesem Grund ist zu erwarten, dass das adaptive Bandbreitenmanagement in einem auf G2G-basierenden P2P-System seine Effizienz durch Kombination mit BNS und BU steigert.

4.2 Datenerhebung

Das adaptive Bandbreitenmanagement ist ein kontinuierlicher Prozess, da sich der Zustand des P2P-Systems in Abhängigkeit der Zeit ändert. Damit der Mechanismus korrekt funktioniert, ist demnach eine fortwährende Datenerhebung erforderlich. Ein ISP, der adaptives Bandbreitenmanagement nutzt, muss wissen, welche seiner Benutzer aktiv innerhalb eines P2P-Systems teilnehmen und wie sie sich innerhalb dieses Systems verhalten. Die notwendige Datenerhebung erfolgt unabhängig von einem bestimmten P2P-Protokoll und betrachtet den P2P-Datenverkehr, den ein Benutzer verursacht, als aggregierten Datenstrom. Dies hat zum Einen den Vorteil, dass der Mechanismus flexibel ist und nicht für bestimmte P2P-Systeme angepasst werden muss. Zum Anderen ist es nicht nötig, dem aggregierten Datenstrom inhaltsspezifische Attribute zu entnehmen. Der Mechanismus respektiert somit die Privatheit der Benutzer. Die Vorhaltedauer der Daten ist abhängig von der Länge des Zeitintervalls zwischen konsekutiven Re-Evaluierungen des Systemzustands. Um P2P-Datenverkehr vom restlichen Datenverkehr zu trennen, kann bspw. eine Filterung durch Analyse des Port-Feldes von TCP-Paketen erfolgen[2]. Selbst wenn ein Peer nicht allgemein bekannte Standard-Ports nutzt, so hat der ISP mittels Deep Packet Inspection die Möglichkeit, Rückschlüsse auf die Zugehörigkeit eines Pakets zu P2P-Datenverkehr zu ziehen.

4.2.1 Informationsquelle

Die Datenerhebung kann sich im Wesentlichen zwei unterschiedlichen Datenquellen bedienen:

- **ISP-seitige Wissensbasis**: Für das Erstellen von Abrechnungen, die Verbindlichkeiten des Kunden gegenüber dem ISP nach Transfer-Volumen aufstellen, benötigt der ISP konkretes Wissen um den akkumulierten Datenverkehr. Der ISP führt mit, wieviele Daten jeder seiner Kunden empfangen und gesendet hat. Diese Statistik kann er weiter aufschlüsseln, indem er die Lokalität der Verbindungspartner berücksichtigt. Eine Kategorisierung von ein- und ausgehendem Datenverkehr in Intra- und Inter-ISP-Datenverkehr ist somit auf Basis der Daten eines Benutzers möglich. Durch dieses Wissen stellt der ISP die Datenerhebungskomponente, die als Teil des adaptiven Bandbreitenmanagements notwendig für die Funktionsweise des Mechanismus ist, bis auf die Kommunikationsschnittstelle (vgl. Abbildung 4.1) selbst.
- **Peer-unterstützte Wissensbasis**: Ein Peer kennt die Endpunkte der Peers, zu denen er eine Verbindung aufgebaut hat und kann die Anzahl empfangener/gesendeter Bytes pro Verbindung mitführen. Über eine Reporting-Schnittstelle kann der Peer kurze Berichte in regelmäßigen Intervallen an die Datenerhebungskomponente schicken. Diese Komponente wertet die erhaltenen Informationen pro Benutzer hinsichtlich Intra- und Inter-ISP-Datenvolumen bspw. unter Verwendung einer GeoIP-Datenbank [45] aus. Unter Nutzung der Peer-unterstützten Wissenbasis kann der Mechanismus weitere Informationen berücksichtigen, bspw. die Bitrate eines Videos. Die Beschreibung des Mechanismus verzichtet jedoch aus Gründen der Einfachheit und Äquivalenz der Wissensbasen auf die Diskussion der dadurch entstehenden Vorzüge. Die Nutzung der Peer-unterstützten Wissensbasis ist mit einem höheren Aufwand verbunden, da die ISP-seitige Datenaggregation entwickelt und konkret an ein P2P-System angepasst werden muss, so dass der Mechanimus seine ursprüngliche Flexibilität einbüßt. Des Weiteren sind bewusst falsche Informationen oder fehlerhafte Daten innerhalb der Peer-Berichte ein Problem. Die Peer-unterstützte Wissensbasis eignet sich daher nur in einem System, in dem eine Vertrauensbeziehung zwischen Benutzer und ISP besteht.

[2] BitTorrent nutzt standardmäßig Ports zwischen 6881 und 6889 [14].

4.2.2 Statistiken über den P2P-Datenstrom eines Benutzers

Dem aggregierten P2P-Datenstrom für einen Benutzer lassen sich die folgenden Informationen entnehmen und im Rahmen des Mechanismus weiterverwenden:

- Das lokale Upload-Volumen eines Peers p, beschrieben durch $U_{\text{lokal}}(t)$. Das lokale Upload-Volumen berechnet sich anhand der Datenmenge, die an Peers innerhalb der selben ISP-Domäne während des Zeitintervalls t *gesendet* wurde.
- Das lokale Download-Volumen eines Peers p, beschrieben durch $D_{\text{lokal}}(t)$. Das lokale Download-Volumen berechnet sich anhand der Datenmenge, die von Peers innerhalb der selben ISP-Domäne während des Zeitintervalls t *empfangen* wurde.
- Das gesamte Upload-Volumen eines Peers p über das Zeitintervall t, beschrieben durch $U_{\text{gesamt}}(t)$. Das gesamte Upload-Volumen schließt gesendete Daten an Peers *innerhalb* und *außerhalb* der ISP-Domäne ein, in der sich Peer p befindet.
- Das gesamte Download-Volumen eines Peers p über das Zeitintervall t, beschrieben durch $D_{\text{gesamt}}(t)$. Das gesamte Download-Volumen schließt empfangene Daten von Peers *innerhalb* und *außerhalb* der ISP-Domäne ein, in der sich Peer p befindet.

Die Datenerhebungskomponente assoziiert diese Statistiken mit einem Benutzer bzw. aus technischer Sicht einem Peer und offeriert die Daten in Form einer unsortierten Liste über die Statistik-Schnittstelle (vgl. Abbildung 4.1). Die Ausgestaltung dieser Schnittstelle und das Format der übergebenen Daten ist Gegenstand der Implementierung und nicht Bestandteil der konzeptionellen Diskussion des Mechanismus. Sie kann von Implementierung zu Implementierung variieren.

4.3 Inferenz

Die Bewertungskomponente hat die Aufgabe, eine Evaluierung aller aktiven Peers in regelmäßigen Zeitabständen durchzuführen. Er erhält die notwendigen Daten durch die Statistik-Schnittstelle der Datenerhebungskomponente. Abschnitt 4.1.2 deutete bereits an, dass eine Erhöhung der Upstream-Kapazität beliebiger Peers zu einem nicht wünschenswerten Ergebnis führen kann. Das adaptive Bandbreitenmanagement kann nur dann funktionieren, wenn Peers, die sich in den Augen eines ISPs günstig verhalten, eine höhere Upload-Bandbreite erhalten. Zu diesem Zweck bedient sich die Bewertungskomponente einer Reihe von Identifikationsmetriken, die einem Peer p einen Gütewert zuordnen. Dieser Gütewert zeigt an, wie sehr sich Peer p für eine Erhöhung der Bandbreite eignet. Die Komponente führt die Berechnung für alle aktiven Peers durch, erstellt aus dem Resultat eine sortierte Liste und gibt diese über die Peer-Listen-Schnittstelle (vgl. Abbildung 4.1) an die Bandbreitenzuteilungskomponente weiter.

4.3.1 Identifikationsmetriken

Die in diesem Abschnitt vorgestellten Metriken zeigen eine starke Kopplung zu den Statistiken, welche von der Datenerhebungskomponente aggregiert werden. Die Liste der Identifikationsmetriken ist sicherlich nicht vollständig. Insbesondere durch Nutzung einer Peer-unterstützten Wissensbasis ist eine applikationsspezifische Ausrichtung von Identifikationsmetriken möglich. Dies unterbindet jedoch die flexible Einsatzmöglichkeit des Ansatzes. Die im Folgenden vorgestellten Identifikationsmetriken haben den Vorteil, dass sie sich alleine aus dem Wissen, das der ISP bereits besitzt, ermitteln lassen. Die Motivation der Metriken ist im Wesentlichen aus Pussep et al. [42] entnommen. Dennoch gibt es subtile Unterschiede zwischen den Metriken in [42] und den hier vorgestellten. Die nachfolgende Einführung der Metriken berücksichtigt dies.

Auslastung der Upload-Kapazität $U(t)$

Damit eine Erhöhung der Upload-Bandbreite eines Peers p nicht ungenutzt bleibt, ist der ISP vornehmlich an der Beförderung solcher Peers interessiert, die möglichst viel von ihren Ressourcen dem Schwarm zur Verfügung stellen.

Dazu genügt es, die Upload-Auslastung über ein bestimmtes Zeitintervall zu betrachten. Die Auslastung ist nichts anderes als das Verhältnis zwischen dem tatsächlich gesendeten Daten-Volumen und der maximalen theoretischen Upload-Kapazität des Peers über das Zeitintervall. Sei d die Länge dieses Zeitintervalls und $B(t)$ eine Funktion, die in Abhängigkeit der Zeit t die Upload-Kapazität des Peers liefert. Dann ist die Auslastung der Upload-Kapazität:

$$U(t) = \frac{U_{\text{gesamt}}(t)}{d \cdot B(t)}$$

Diese Metrik hat die wünschenswerte Eigenschaft, dass sie unabhängig von der Heterogenität der Peers ist. Das bedeutet, dass die Metrik für einen Peer mit einer niedrigen Upload-Kapazität im Vergleich zu einem Peer mit hoher Upload-Kapazität denselben Wertebereich $W = [0, 1]$ umfasst. Die Metrik bewertet einen Peer ausschließlich auf die Nutzung der ihm zur Verfügung stehenden Ressourcen. Ein hoher Gütewert nach dieser Metrik ist demnach ein starker Indikator dafür, ob die Bandbreite eines Peers erhöht werden soll, oder nicht.

Verhältnis zwischen Upload und Download $S(t)$

Die Idee hinter dieser Metrik ist, altruistische Peers zu identifizieren. Ein altruistischer Peer beteiligt sich an der Verteilung der Daten im Schwarm über seine Download-Zeit hinaus. Dahingegen stellt ein egoistischer Peer den Upload an andere Peers des Schwarms ein, sobald er seinen Download beendet hat. Stattet man einen altruistischen Peer mit einer erhöhten Upload-Kapazität aus, so ist die Annahme, dass er diese Upload-Kapazität anderen Peers über einen längeren Zeitraum zur Verfügung stellt als ein egoistischer Peer. Das steigert die Leistungsfähigkeit des Overlays, da andere Peers ihren Download schneller abschließen und eine größere Diversität an Datenblöcken anbieten können. Die Aufgabe der hier vorgestellten Metrik ist daher, Peers gemäß ihres Verhaltens voneinander abzugrenzen, wobei altruistische Peers einen hohen Gütewert erhalten sollen. Eine Funktion, die dies leistet ist:

$$S(t) = \frac{U_{\text{gesamt}}(t)}{U_{\text{gesamt}}(t) + D_{\text{gesamt}}(t)}$$

Diese Metrik besitzt ebenfalls die Eigenschaft, dass sie unabhängig von der Heterogenität der Peer arbeitet. Der Wertebereich von $S(t)$ umfasst ebenfalls $W = [0; 1]$. Gilt $S(t) = 0{,}5$, dann hat der korrespondierende Peer soviel hochgeladen, wie er heruntergeladen hat. In Anbetracht asymmetrischer Zugangsprofile befindet sich ein solcher Peer bereits länger als notwendig im System. Gütewerte über 0,5 sind daher bereits ein starker Indikator dafür, dass sich ein Peer altruistisch verhält. Je weiter sich der Gütewert der 1,0 annähert, desto länger befindet sich der korrespondierende Peer im System. Eine Korrelation mit der Online-Zeit des bewerteten Peers ist allerdings nicht in allen Fällen möglich, da ein neu angekommener Peer bereits einen hohen Gütewert aufweisen kann, wenn er schnell Tauschpartner findet, an die er Daten senden kann, selbst aber zu Beginn mit einer niedrigeren Transferrate Daten herunterlädt. Dieser Fall ist jedoch die Ausnahme, da in der Regel Peers bewertet werden, die sich bereits eine gewisse Zeit im Overlay befinden.

Lokalitätsmetrik $L(t)$

Das adaptive Bandbreitenmanagement soll Peers mit zusätzlichen Ressourcen ausstatten, die sie größtenteils zur Versorgung von Peers aus derselben ISP-Domäne nutzen. Dies liegt sicherlich im Interesse des ISPs, denn ein Peer, der größtenteils Verbindungen zu Peers aus anderen ISP-Domänen unterhält, würde zusätzliche Ressourcen mit hoher Wahrscheinlichkeit für Transfers über Inter-ISP-Verbindungen aufwenden. Damit dieser Fall nicht eintritt, ist eine Bewertung der Peers hinsichtlich der Lokalität ihrer Ressourcennutzung wichtig. Dabei sollen Peers, die größtenteils Peers aus derselben ISP-Domäne versorgen, einen höheren Gütewert aufweisen, als Peers, die ein hohes Daten-Volumen über Inter-ISP-Verbindungen erzeugen. Diverse Arbeiten [55, 53] zeigen jedoch, dass ein zu hohes Maß an lokalem Datenverkehr für ISP und P2P-System ebenfalls ungünstig sein kann. Die Lokalitätsmetrik muss diesen Umstand berücksichtigen und Peers, die *zu viel* lokalen Datenverkehr erzeugen, abwerten. Diese Abwertung

darf allerdings nur geringfügig erfolgen. Zum Einen, weil ein hoher Lokalitätswert grundsätzlich darauf hindeutet, dass der korrespondierende Peer lokale Verbindungen bevorzugt. Zum Anderen soll durch die Abwertung eine möglichst sinnvolle Abgrenzung zweier Peers vorgenommen werden, die sich bereits durch hohe Lokalitätswerte ausgezeichnet haben. Letzten Endes soll der Peer den besseren Gütewert erhalten, der ein lokalitätsbewusstes Verhalten verspricht, ohne dabei ISP-interne Kommunikationspfade zu überlasten.

Eine simple, aber naive Metrik, welche die angesprochenen Probleme nicht berücksichtigt, berechnet das Verhältnis zwischen Intra- und Inter-ISP-Datenvolumen:

$$L(t) = \frac{U_{\text{lokal}}(t) + D_{\text{lokal}}(t)}{U_{\text{gesamt}}(t) + D_{\text{gesamt}}(t)}$$

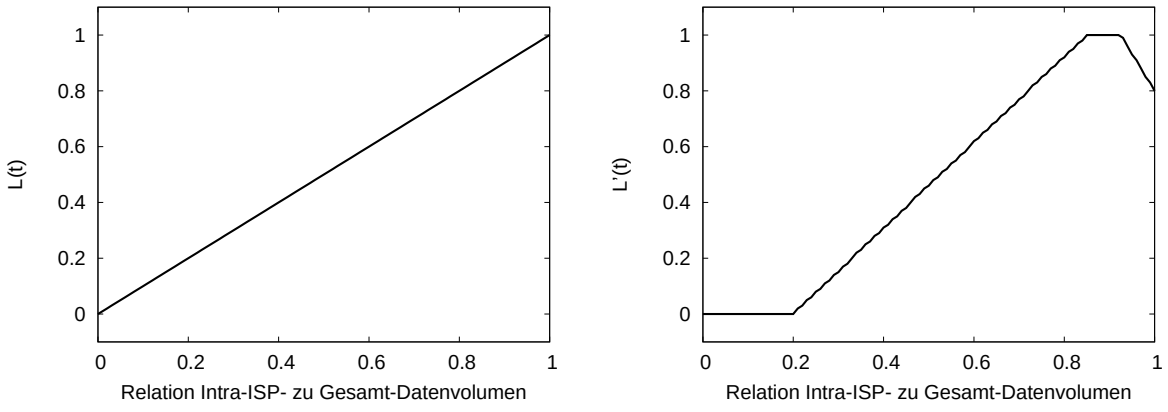
Eine gute Separation der Peers ist bereits unter Verwendung dieser Funktion als Lokalitätsmetrik zu erwarten. Die Berücksichtigung der angesprochenen Probleme ist durch eine einfache Transformation möglich. Die Transformation der Ausgabewerte von $L(t)$ realisiert eine Trapezoid-Funktion $T(x)$:

$$T(x) = \begin{cases} 0{,}0, & \text{falls } x \leq k_0 \\ \frac{x}{k_1 - k_0} - \frac{k_0}{k1 - k0}, & \text{falls } k_0 < x < k_1 \\ 1{,}0, & \text{falls } k_1 \leq x \leq k_2 \\ \frac{-x}{k_3 - k_2} + \frac{k_3}{k_3 - k_2}, & \text{falls } k_2 < x \leq k_3 \end{cases}$$

Die Transformationsfunktion $T(x)$ basiert auf vier Parametern, die den An- und Abstieg der Funktion charakterisieren. Die Vorgabe hierfür ist $x_0 = 0{,}2, x_1 = 0{,}85, x_2 = 0{,}925, x_3 = 1{,}3$. Der Anstieg der Kurve beginnt erst ab einem $x = 0{,}2$, so dass ungünstige Peers schärfer ausgegrenzt werden. Die Kurve fällt im Intervall $[0{,}925, 1{,}3]$, wobei Werte jenseits 1,0 keine Rolle spielen, da der Wertebereich von $L(t)$ auf $W = [0; 1]$ normiert ist. Die konkreten Parameterwerte kann der ISP gemäß seinen Vorstellungen (infrastruktureller Auslegung, Bestrafung von zu wenig Lokalität usw.) auslegen.

Die im Rahmen dieser Arbeit verwendete Lokalitätsmetrik $L'(t)$ fasst die Transformation von $L(t)$ zusammen:

$$L'(t) = T(L(t))$$



(i) Kurvenverlauf für $L(t)$

(ii) Kurvenverlauf für $L'(t)$

Abbildung 4.3: Qualitativer Kurvenverlauf der Lokalitätsmetrik nach $L(t)$ und $L'(t)$. $L'(t)$ berücksichtigt negative Effekte, die sich durch zuviel Lokalität ergeben können.

Abbildung 4.3 zeigt den Kurvenverlauf für beide Lokalitätsmetriken. Über die Selektionsgüte beider Metriken kann an dieser Stelle keine Aussage getroffen werden. Es ist Gegenstand der Evaluation in Kapitel 8, die Funktionen hinsichtlich ihrer Selektionsgüte zu untersuchen.

4.3.2 Berechnung des Gütewerts

Die Berechnung eines Gütewerts zu einem Peer kann anhand einer oder mehreren der vorgestellten Metriken erfolgen. Die Resultate der Metriken können individuell gewichtet werden, um bspw. den Einfluss der Lokalitätsmetrik stärker in die Berechnung einfließen zu lassen als die Resultate des Upload-Download-Verhältnisses. Sei $\vec{w}$ ein Gewichtungsvektor und $\vec{x}^T$ ein Vektor über den Identifikationsmetriken, dann ist die Bewertungsfunktion $R(t)$:

$$R(t) = \vec{w}^T \cdot \vec{x} = \begin{pmatrix} w_u & w_s & w_l \end{pmatrix}^T \begin{pmatrix} U(t) \\ S(t) \\ L'(t) \end{pmatrix}$$

Eine Gewichtung von $w_i = 0$ sorgt dafür, dass die korrespondierende Identifikationsmetrik keinen Einfluss auf den Gütewert ausübt. Der Gütewert eines Peers kann in Anbetracht kurzfristiger, negativer Erscheinungen schwanken. Um für eine Stabilisierung der Berechnung zu sorgen, kann der bisherige Verlauf der Gütewerte eines Peers unter Verwendung des exponentiell gleitenden Durchschnitts einbezogen werden. Die Berechnungsvorschrift für $R(t)$ ist dann:

$$R(t) = \alpha \cdot \vec{w}^T \cdot \vec{x} + (1 - \alpha) \cdot R(t - d)$$

Wobei α ein Gewichtungsfaktor ist, der angibt, zu welchem Anteil die aktuelle Situation Einfluss in die Güteberechnung erhalten soll und d die Dauer zwischen zwei konsekutiven Bewertungen beschreibt. Im Rahmen der vorliegenden Arbeit gilt $\alpha = 0{,}8$.

4.3.3 Selektion der besten Peers

Der Mechanismus selektiert aus der Menge der bewerteten Peers die Besten. Dies kann auf unterschiedliche Arten erfolgen:

- **Ein-Pass-Selektion**: Die Berechnung des Gütewerts erfolgt einmal. Der Mechanismus wählt aufgrund der Gütewerte die besten k Peers aus.
- **Zwei-Pass-Selektion**: Der Mechanismus bewertet die Peers zunächst nach einer spezifischen Identifikationsmetrik und selektiert die k_1 besten Peers unter ihnen. Nun wiederholt der Mechanismus diesen Selektionsprozess, indem er für jeden Peer der reduzierten Peer-Menge erneut Gütewerte berechnet, diesmal aber anhand einer anderen Identifikationsmetrik. Die besten k_2 Peers werden selektiert. Man erhält durch diese Selektionsvariante eine konsekutiv reduzierte Peer-Menge, welche die besten k_2 Peers anhand einer primären und einer sekundären Identifikationsmetrik enthält.

4.4 Bandbreitenzuteilung

Die Bandbreitenzuteilung erhält eine nach Gütewerten absteigend sortierte Liste aller aktiven Peers. Die Aufgabe dieser Komponente ist es nun, einen gewissen Anteil λ von diesen Peers zu selektieren und diese mit einer erhöhten Upstream-Kapazität auszustatten. Die erhöhte Upstream-Kapazität unterscheidet sich um einen Faktor μ von der Grund-Kapazität eines Peers. Nach der Auswahl geeigneter Peers erfolgt die Umsetzung der neu zugeteilten Bandbreiten. Der Mechanismus kann hierzu eine bestehende Lösung, wie bspw. Linux Traffic Control (vgl. Kapitel A), integrieren. Peers, deren Bandbreite bereits erhöht worden ist, erhalten keine weitere Erhöhung.

Da die Ressourcen eines ISPs begrenzt sind, ist es von Vorteil, wenn dieser von einem fixen Kontingent an zusätzlichen Ressourcen ausgehen kann, die er für das adaptive Bandbreitenmanagement zur Verfügung stellen möchte. Auf Basis dieser Vorgabe kann der ISP bspw. einen Kompromiss zwischen dem Anteil zu befördernder Peers λ und dem Multiplikationsfaktor μ ausarbeiten. Ein einfaches mathematisches Modell soll dies verdeutlichen.

Sei C_{Zusatz} die zusätzliche Kapazität, die der ISP für das adaptive Bandbreitenmanagement aufwenden möchte. Sei ferner P die Menge aller Peers und $u_p(t)$ die derzeitige Upstream-Kapazität eines Peers p. Dann gilt:

$$\begin{aligned} & \lambda \sum_{p \in P} (\mu \cdot u_p(t) - u_p(t)) && \leq C_{\text{Zusatz}} \\ \Leftrightarrow \quad & (\mu - 1) \cdot \lambda \sum_{p \in P} u_p(t) && \leq C_{\text{Zusatz}} \end{aligned}$$

Stellt man diese Gleichung nach λ resp. μ um, so kann eine konkrete Wertbelegung für diese Parameter errechnet werden. λ und μ verbindet eine wechselwirkende Beziehung miteinander. Da die Vorgabe der zusätzlichen Ressourcen fix ist, sorgt eine Erhöhung des Upstream-Multiplikationsfaktors μ dafür, dass ein geringerer Anteil λ an Peers mit einer erhöhten Bandbreite ausgestattet werden kann. Umgekehrt sorgt die Promotion mehrerer Peers dafür, dass sich der Multiplikationsfaktor niedriger einstellt.

Nachfolgende Auflistung zeigt den prinzipiellen Ablauf der Bandbreitenzuteilung:

1. Der Algorithmus errechnet zuerst die Anzahl k der Peers, die eine erhöhte Upstream-Kapazität erhalten sollen.
2. Der Algorithmus prüft für einen Peer p aus den k besten Peers, ob dessen Upstream-Kapazität bereits erhöht wurde.
   a) Wurde die Kapazität noch nicht verändert, so erhöht der Algorithmus die Upstream-Kapazität des Peers p um den Faktor μ.
   b) Wurde die Kapazität in einer vorhergehenden Iteration bereits erhöht, so verändert der Algorithmus das Zugangsprofil von Peer p nicht.
3. Der Algorithmus betrachtet nun für jeden Peer p aus den nicht zu befördernden Peers, ob dessen Bandbreite zuvor erhöht wurde.
   a) Wurde die Kapazität in einer vorhergehenden Iteration erhöht, so wird sie jetzt auf die Upstream-Kapazität zurückgesetzt.
   b) Wurde die Kapazität in einer vorhergehenden Iteration nicht erhöht, so bleibt die Upstream-Kapazität unverändert.

4.5 Realisierung auf Seiten des ISP

Ein ISP hat mehrere Möglichkeiten, das adaptive Bandbreitenmanagement innerhalb seiner Netzstruktur zu platzieren. Die entsprechend modifizierten Zugangsprofile stellt das adaptive Bandbreitenmanagement bereit. Der Mechanismus kann auf einem dedizierten Host im ISP-Netzwerk arbeiten. Zur Umsetzung konkreter Zugangsprofile eignet sich bspw. NGN-kompatible Hardware. Diese Hardware ist in der Lage, automatisierte Aktualisierungen von Zugangsprofilen einzelner Kunden durchzuführen [56, 57].

Abbildung 4.4 zeigt eine weitere Möglichkeit, die auf konventionelle Hardware aufsetzt. Die Idee ist, den Zugang des Kunden *nicht* durch den DSLAM (Digital Subscriber Line Access Multiplexer) zu limitieren, sondern auf der Verbindung zwischen dem DSLAM und einem nachgeschalteten Peering Router. Der gesamte Datenverkehr zwischen DSLAM und dem Peering Router passiert zunächst einen speziell für diesen Zweck eingerichteten Linux-Server, der bspw. Linux Traffic Control nutzt, um ein- und ausgehenden Datenverkehr zu manipulieren [57]. Dieser Traffic Shaper setzt individuell zugewiesene Zugangsprofile um und ist in der Lage, diese Profile automatisch zu aktualisieren. Die notwendigen Daten liefert wieder das adaptive Bandbreitenmanagement.

4.6 Einschränkungen im Rahmen der Arbeit

In der Praxis ist man daran interessiert, das Benutzerverhalten über eine gewisse Zeit zu beobachten, bevor man ein erhöhtes Zugangsprofil vergibt. Der zeitliche Rahmen dürfte schätzungsweise bei 24 Stunden liegen. In dieser Arbeit ist eine derartige Konfiguration des Mechanismus nicht möglich, da die Evaluation in einem Live Testbed

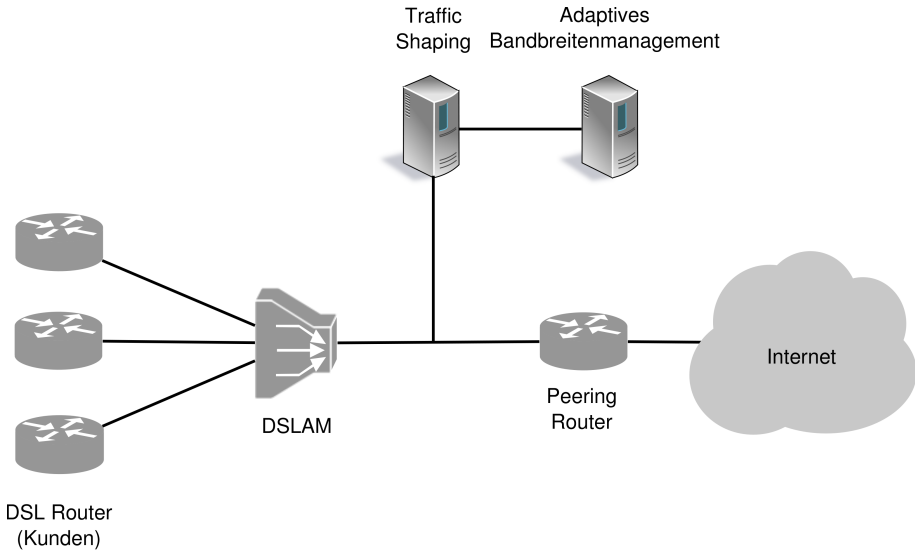


Abbildung 4.4: Vereinfachte Darstellung einer beispielhaften ISP-Plattform mit Zugangsbereich

erfolgt. Die zeitlichen Abstände zwischen konsekutiven Re-Evaluierungen sind daher deutlich kürzer (vgl. Kapitel 8), um den Mechanismus sinnvoll in einer Reihe von Testszenarien erproben zu können.

Des Weiteren ist die Integration eines Tools zur Bandbreitenumsetzung nicht möglich, da die Evaluationsplattform selbst durch ein derartiges Tool verwaltet wird. Im Rahmen der vorliegenden Arbeit nutzen wir daher eine Ratenlimitierung auf Applikationsebene, die bereits im NextShare Client implementiert ist.

5 Implementierung

Die Erprobung des adaptiven Bandbreitenmanagements in einer hybriden Systemarchitektur erfordert entsprechende Software-Lösungen. Dieses Kapitel gibt Aufschluss über die notwendigen Implementierungen im Rahmen der vorliegenden Arbeit.

Die Basis für Implementierungsarbeiten stellt der NextShare Client in Version M24 dar. Die Supporter-Strategie (Abschnitt 5.2) sowie der Video-on-Demand Client-Emulator (Abschnitt 5.3) sind als unabhängige Pakete für NextShare entwickelt worden. Die Realisierung des adaptiven Bandbreitenmanagements erfolgt separat. Die Beschreibung der Implementierungsarbeiten zur Supporter-Strategie ist etwas detaillierter, da sie insbesondere die Unterschiede zur konzeptionellen Arbeit [9, 10] herausstellt.

5.1 Implementierung des adaptiven Bandbreitenmanagements

Die Implementierung des adaptiven Bandbreitenmanagements fasst aus Gründen der Einfachheit die notwendigen Komponenten in einer Software-Lösung zusammen. Der Datenaustausch zwischen den Komponenten erfolgt über wohldefinierte Schnittstellen. Die nachfolgenden Abschnitte widmen sich der Struktur der Implementierung und dem konkreten Ablauf einer Bewertungsrunde mitsamt Zuteilung neuer Zugangsprofile.

5.1.1 Modulübersicht

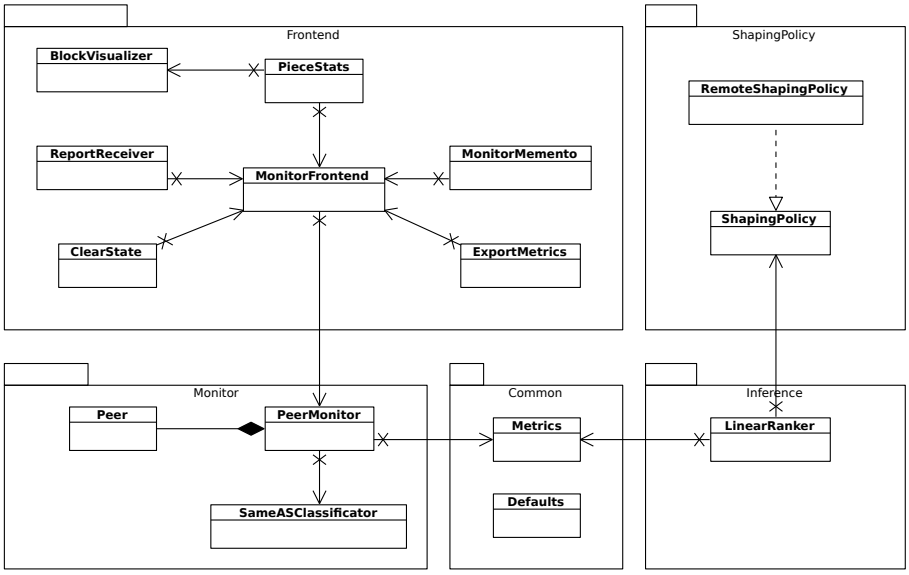


Abbildung 5.1: Klassendiagramm des adaptiven Bandbreitenmanagements (Grobansicht)

Die Software ist in mehrere Module unterteilt. Abbildung 5.1 zeigt die Zugehörigkeit der Klassen zu den Modulen und gibt Aufschluss über die Beziehungen dieser Entitäten untereinander. Das Paket `Common` beinhaltet Klassen, die von allen anderen Paketen des Projekts verwendet werden.

`Monitor` realisiert die Datenerhebungskomponente und speichert den aktuellen (und vergangene) Peer-Zustände. Die Schnittstelle des Datenempfangs realisiert das Modul `Frontend`. Es nutzt das CherryPy-Framework als Webserver und realisiert URL-Handler, die Anfragen auf bestimmte HTTP-Ressourcen (bspw. `/export_metrics` durch Klasse `ExportMetrics`) beantworten. Der in Abschnitt 5.3 beschriebene Client-Emulator kann diese web-basierte Schnittstelle nutzen, um Peer-Statistiken an die Datenerhebungskomponente zu senden. Zu Debugging-Zwecken kann der aktuelle Zustand aller Peers zur Laufzeit aufgerufen werden.

Das Paket `Inference` realisiert die Bewertungskomponente. Es besteht im Wesentlichen aus einem linearen Ranking-Algorithmus, der alle aktiven Peers von der Datenerhebungskomponente anfordert und unter Zuhilfenahme der Identifikationsmetriken bewertet.

Das Paket `ShapingPolicy` realisiert die Bandbreitenzuteilung. Im Rahmen dieser Arbeit erfolgt die Zuteilung der Bandbreiten über eine client-seitige Ratenlimitierung. Das Paket beinhaltet realisiert eine XML-RPC-Anbindung an einen NextShare Client, um dessen Ratenlimitierung zu beeinflussen.

5.1.2 Ablauf einer Bewertungsrunde

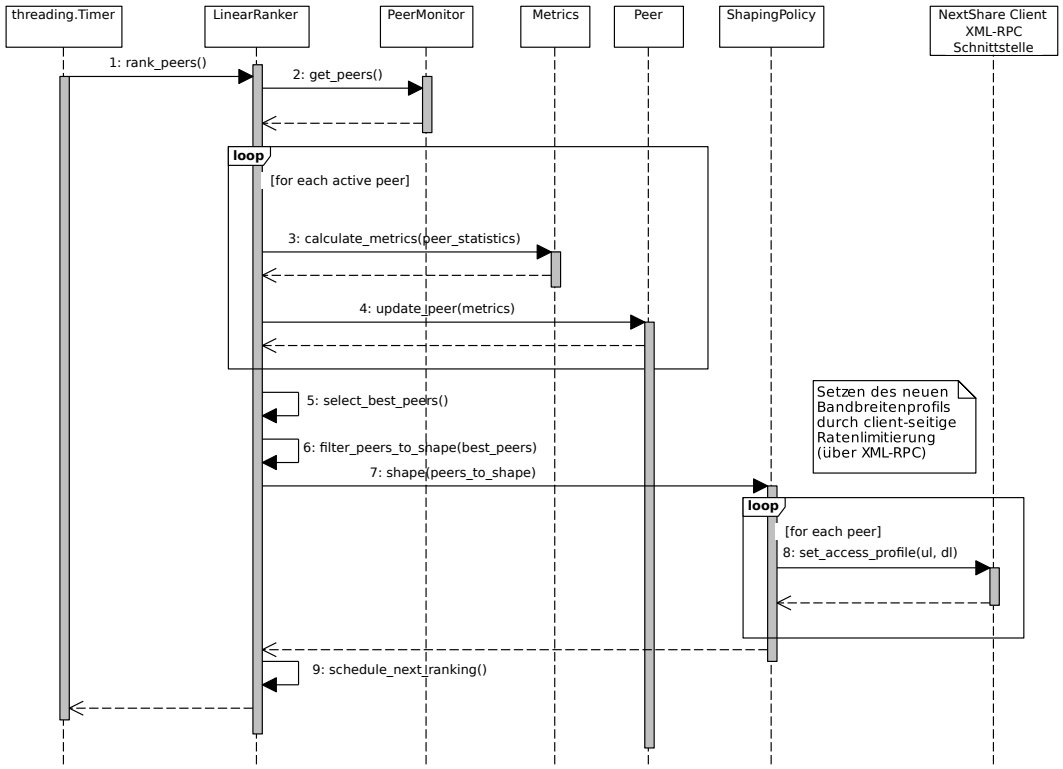


Abbildung 5.2: Ablauf einer Bewertungsrunde mit anschließender Zuteilung der neuen Zugangsprofile

Abbildung 5.2 zeigt den Ablauf einer Bewertungsrunde. Die Bewertung von Peers erfolgt in periodischen Abständen nach einem vorher konfigurierten Zeitmaß. Der lineare Ranking-Algorithmus fordert von der Datenerhebung alle aktiven Peers an. Ein Peer gilt als inaktiv, wenn er über einen Zeitraum von mehr als 10 Sekunden keine Statistiken eingereicht hat. Der lineare Ranking-Algorithmus berechnet für jeden aktiven Peer den Gütewert durch die Identifikationsmetriken (`calculate_metrics`) aufgrund der Statistiken, die die Datenerhebung seit der letzten Berechnungsrunde aggregiert hat. Die Methode `select_peers` wählt aus den bewerteten Peers eine Untermenge der besten Peers aus. In der Implementierung erfolgt diese Filterung zuerst nach Metrik $U(t)$ (vgl. Abschnitt 4.3.1), dann ein zweites mal nach $S(t)$ (vgl. Abschnitt 4.3.1).

Die Bandbreite eines Peers muss angepasst werden, wenn einer der folgenden Fälle eintritt:

1. Der Peer gehört zu der Menge der besten Peers und seine Bandbreite wurde in einer vorhergehenden Berechnungsrunde noch nicht modifiziert. Eine Erhöhung des Zugangsprofils dieses Peers erfolgt.
2. Der Peer gehört *nicht* zu der Menge der besten Peers, aber seine Bandbreite wurde in einer vorhergehenden Runde erhöht. Der Peer nimmt seine ursprüngliche Bandbreite wieder an.

Die Methode `filter_peers_to_shape` filtert Peers entsprechend dieser Fälle. Der lineare Ranking-Algorithmus übergibt die Liste aller Peers, deren Zugangsprofil geändert werden muss, an eine konkrete Realisierung der Schnittstelle `ShapingPolicy`. Die Aufgabe der `ShapingPolicy` ist die Umsetzung der neuen Zugangsprofile. Die

Implementierung nutzt hierfür eine XML-RPC-basierte Anbindung an den betreffenden NextShare Client, der seinerseits einen XML-RPC-Server zur Verfügung stellt, über den er neue Zugangsprofile erhalten kann. Eine clientseitige Ratenlimitierung sorgt anschließend dafür, dass dieses Zugangsprofil umgesetzt wird.

5.2 Implementierung der Supporter Strategie

Die Supporter-Strategie realisiert die Server-Komponente der in dieser Arbeit verwendeten hybriden Systemarchitektur. Sie wurde im Rahmen der Arbeit prototypisch implementiert. Die Implementierung folgt der Beschreibung aus Abschnitt 2.4.2.

5.2.1 Kommunikation

Für die Umsetzung der Supporter-Strategie ist die Kommunikation aller Komponenten der Systemarchitektur von entscheidender Bedeutung. Zum Einen müssen unterversorgte Peers in der Lage sein, ihren Bedarf an dem Überwachungsmechanismus anzumelden. Zum Anderen muss der Überwachungsmechanismus die aktuelle Server-Allokation an entsprechende Supporter kommunizieren können. Dieser Abschnitt beschäftigt sich mit der Definition bzw. Modifikation der entsprechenden Protokolle und der Integration mit dem NextShare-Client. Eine wichtige Eigenschaft der Integration ist es, Änderungen an bestehenden Protokollen (und in zweiter Instanz dem Quelltext) minimal zu halten. Das ursprüngliche Protokoll darf durch die Erweiterung nicht verletzt werden, so dass Peers, die den Supporter Mechanismus nicht kennen, sich dennoch in gewohnter Weise im Overlay bewegen können, ohne einen Nachteil zu erfahren.

Kommunikation von Video-on-Demand Client und Supporter mit dem Überwachungsmechanismus

Als Integrationspunkt für den Überwachungsmechanismus bietet sich der zentrale Index-Server an, der in NextShare durch einen internen Tracker gebildet wird. Dieser Tracker ist konform zur BitTorrent-Spezifikation [14] und bietet demzufolge eine Reihe von HTTP-Ressourcen für die Interaktion mit einzelnen Peers an. Diese Ressourcen erlauben es einzelnen Peers, ihren Zustand am Tracker zu aktualisieren[1] und Informationen über spezifische Schwärme zu erhalten. Es ist durchaus sinnvoll, die Signalisierung der Supporter-Strategie als Erweiterung dieses HTTP-basierten Protokolls zu implementieren. Peers, die den Supporter-Mechanismus nicht kennen, können weiterhin mit dem Tracker in gewohnter Weise kommunizieren und erfahren die reguläre Behandlung. Peers, welche die Protokoll-Erweiterung kennen, können die neu definierten HTTP-Ressourcen benutzen, um die Supporter-Strategie in Anspruch zu nehmen. Ein Supporter muss sich lediglich einmal registrieren. Sein Zustand wird über den Überwachungsmechanismus selbst bestimmt, so dass Zustandsaktualisierungen durch den Supporter nicht notwendig sind. Wir unterscheiden die nachfolgenden Anwendungsfälle:

1. Ein Supporter tritt dem Overlay bei und möchte dies dem zentralen Index-Server signalisieren (Registrierung).
2. Ein Peer tritt dem Overlay bei und möchte dem zentralen Index-Server signalisieren, dass er die Protokollerweiterung spricht und gegebenfalls benutzen möchte (Registrierung). Die Registrierung erfolgt pro Schwarm.
3. Ein bereits registrierter Peer stellt fest, dass sein Wiedergabepuffer nicht vollständig gefüllt ist und möchte Unterstützung anfordern (Zustandsaktualisierung).
4. Ein Peer der bereits Unterstützung erhält, stellt fest, dass sein Wiedergabepuffer wieder vollständig gefüllt ist und möchte dem zentralen Index-Server signalisieren, dass die Unterstützung nicht länger notwendig ist (Zustandsaktualisierung).

Aus diesen Anforderungen lassen sich direkt die erforderlichen HTTP-Ressourcen ableiten. Tabelle 5.1 zeigt die neu definierten HTTP-Ressourcen nebst Bedeutung und erforderlichen Parametern.

Anzumerken ist, dass der Parameter Port, der von Peers übermittelt wird, für die korrekte Umsetzung der Supporter-Strategie nicht von Bedeutung ist. Er dient in der prototypischen Implementierung lediglich dazu, weitere Infor-

[1] Der Zustand eines Peers variiert zwischen *beigetreten*, *Download aktiv* und *Download abgeschlossen*.

HTTP-Ressource	Bedeutung	Parameter
`/register_supporter`	Supporter registriert sich	ID, Port, min./max. Peers
`/register_peer`	Peer registriert sich	Peer-ID (schwarm-bezogen), Port
`/request_support`	Peer fordert Unterstützung an	Peer-ID (schwarm-bezogen)
`/abort_support`	Peer fordert keine Unterstützung mehr an	Peer-ID (schwarm-bezogen)

Tabelle 5.1: Übersicht über die HTTP-Ressourcen der Tracker-Protokollerweiterung

mationen über partizipierende Peers zu sammeln. Eine praxistaugliche Implementierung der Protokollerweiterung kann diesen Parameter vernachlässigen und somit zusätzlichen Signalisierungs-Overhead einsparen. Für die Kommunikation mit einem Supporter Server ist die Port-Information jedoch von zentraler Bedeutung.

Kommunikation von Überwachungsmechanismus zu Supporter

Da der Überwachungsmechanismus die Allokation von Peers zu Supporter vornimmt, bietet es sich an, ein push-orientiertes Protokoll zur Kommunikation der Zustandsänderung zu realisieren. Da sich Supporter Server nicht explizit von der Überwachungskomponente abmelden, wenn sie das Overlay verlassen, ist es ferner nötig, über einen pull-orientierten Ansatz periodisch zu prüfen, ob sich registrierte Supporter noch im Overlay befinden. Diese beiden Anforderungen lassen sich sehr einfach über eine XML-RPC-Schnittstelle realisieren, die ein Supporter als Server bereitstellt und von der Überwachungskomponente als Client genutzt werden kann. Es folgt die Spezifikation der notwendigen Methoden:

Methode `receive_peer_list(list_of_peer_ids)`:

Über diese Methode kann dem entsprechenden Supporter eine Liste zugewiesener Peers übergeben werden. Diese Liste besteht aus schwarm-bezogenen Peer-IDs. Durch das Supporter-seitige Verbindungsmanagement kann diese Liste weiterverarbeitet werden. Der Supporter retourniert entsprechend der erfolgreichen Weiterverarbeitung einen Wahrheitswert.

Methode `is_alive()`:

Über diese Methode kann die Überwachungskomponente erfragen, ob der Supporter noch aktiv ist. Der Supporter retourniert einen Wahrheitswert.

5.2.2 Erweiterung des NextShare Client um die Supporter-Strategie

Der NextShare Client wird im Wesentlichen um zwei Module erweitert, die in den bestehenden Code integriert wurden:

- Modul `SupporterMonitor.py`: Dieses Modul beinhaltet sämtliche Klassen, die für die Implementierung des Überwachungsmechanismus notwendig sind.
- Modul `Supporter.py`: Dieses Modul implementiert den Supporter Server als modifizierten NextShare Client. Der Supporter Server arbeitet im Konsolenmodus.

Modul `SupporterMonitor`

Die nachfolgenden Paragraphen beschreiben die konkrete Umsetzung einzelner Aspekte des Überwachungsmechanismus. Zur visuellen Stütze zeigt Abbildung 5.3 alle beteiligten Klassen sowie deren Beziehungen untereinander. Die Abbildung verzichtet dabei auf die Anzeige öffentlicher Methoden und privater Variablen. Diese Details lassen sich Abbildung C.2 entnehmen.

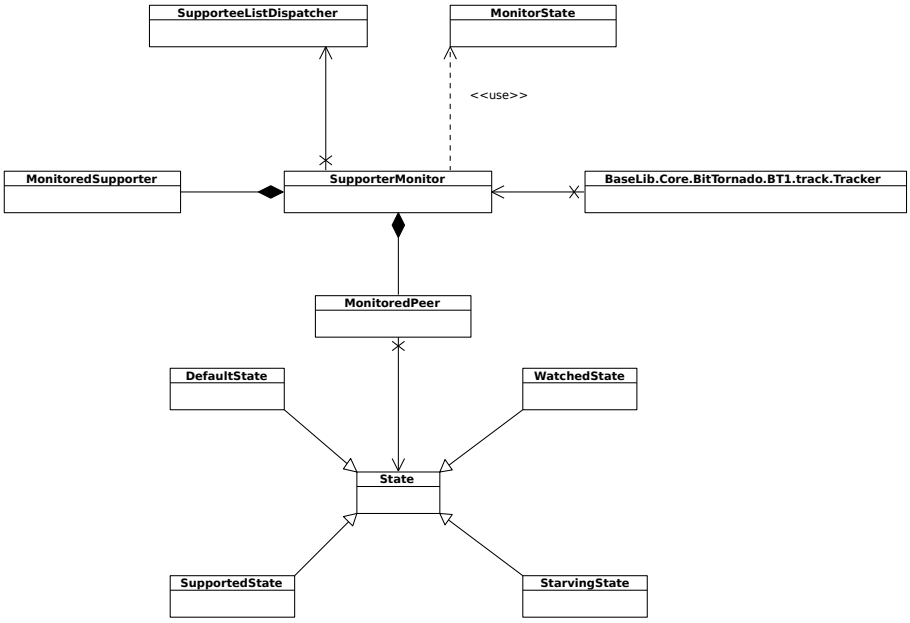


Abbildung 5.3: Klassendiagramm des Moduls `SupporterMonitor` (Grobansicht)

Das Transitionssystem

Das Transitionssystem wird durch die Klassen `DefaultState` (Zustand *Normal*), `WatchedState` (Zustand *Beobachtet*), `StarvingState` (Zustand *Unterversorgt*) und `SupportedState` (Zustand *Unterstützt*) gebildet. Die Zustandsklassen implementieren die gemeinsame Schnittstelle `State`. Jeder Zustand besitzt eine Referenz auf die interne Repräsentation des zu beobachtenden Peers (vgl. `MonitoredPeer`) und implementiert die Zustandstransition in der Methode `State.transition`. Die Umsetzung der Zustandswechsel erfolgt gemäß des in Abbildung 2.6 vorgestellten Transitionssystems, mit der Ausnahme einer zusätzlichen Bedingung für jeden Zustand. Erhält ein Peer Unterstützung, so sendet er über diese Phase weiterhin Unterversorgt-Nachrichten an die Überwachungskomponente. Bleiben diese Nachrichten aus, so wechselt der Peer nach einer gewissen Zeit automatisch zurück in den Zustand *Normal*. Dort verweilt er bis er wieder Nachrichten schickt oder wird nach einer Zeitüberschreitung als nicht mehr im System eingestuft und nicht mehr länger von der Überwachungskomponente verwaltet. Diese Überprüfung erfordert keine Kommunikation zwischen dem Überwachungsmechanismus und dem Peer, da allein anhand der zeitlichen Differenz zu der letzten empfangenen Nachricht des Peers entschieden wird, ob dieser als im System befindlich eingestuft wird.

Der Überwachungsmechanismus

Der Überwachungsmechanismus der Supporter-Strategie ist in der Klasse `SupporterMonitor` implementiert. `SupporterMonitor` verwaltet zu beobachtende Peers (vgl. `MonitoredPeer`) und Supporter Server (vgl. `MonitoredSupporter`), die sich im System befinden. Eingehende HTTP-Anfragen werden an die entsprechenden Klassenobjekte zugehöriger Peers oder Supporter weitergereicht und dort verarbeitet. Die Zustände aller im System befindlicher Peers und Supporter werden *synchron* alle t Zeiteinheiten (konfigurierbar) aktualisiert. Die synchrone Zustandsaktualisierung erfolgt daher unabhängig davon, ob eine Nachricht empfangen wurde. Dies ist notwendig, da sich Peers und Supporter aus dem Overlay entfernen können, ohne dass der Überwachungsmechanismus eine explizite Nachricht über den Austritt erhält. Damit die Zustände der Peers stets aktuell sind, muss eine sofortige Verarbeitung empfangener Nachrichten gewährleistet werden. Aus diesem Grund erfolgt eine *asynchrone* Zustandsaktualisierung individuell für den Peer, von dem eine Nachricht empfangen wurde. Durch die Notwendigkeit für synchrone und asynchrone Zustandsaktualisierungen muss die Verarbeitung innerhalb der Klasse nebenläufig erfolgen. Die Implementierung ist durch `threading.Lock` an den entsprechenden Stellen gegenüber nebenläufigen Verletzungen des Objektzustands abgesichert. Während der synchronen Zustandsaktualisierung werden unterversorgte Peers an Supporter zugewiesen. Die chronologische Reihenfolge der Abarbeitung zeigt nachfolgende Auflistung:

1. Peers, die einer Zeitüberschreitung unterliegen, werden aus der Überwachungskomponente entfernt.
2. Supporter, die einer Zeitüberschreitung unterliegen, werden aus der Überwachungskomponente entfernt.
3. Eine Aktualisierung des Zustands aller im System befindlichen Peers p wird erzwungen.
4. Eine Aktualisierung des Zustands aller im System befindlichen Supporter s wird erzwungen.
5. Unterversorgte Peers werden bereits aktiven Supporter Servern zugewiesen.
6. Ist die Anzahl verbleibender unterversorgter Peers > 0, so versucht der Überwachungsmechanismus weitere Supporter zu aktivieren und restliche Peers zuzuweisen.
7. Sofern sich eine Änderung am Systemzustand ergeben hat, werden Peer-Listen an aktivierte Supporter verschickt, so dass diese ihr Verbindungsmanagement aktualisieren können.

Insbesondere sei darauf hingewiesen, dass zwischen aktivem und passivem Supporter einzig und allein auf Seiten der Überwachungskomponente unterschieden wird. Der Supporter muss darüber kein Wissen haben, da er standardmäßig alle Verbindungen blockiert, die nicht explizit von der Überwachungskomponente freigegeben worden sind. Des Weiteren ist die Aktivierung passiver Supporter in Abhängigkeit von der Anzahl verbleibender unterversorgter Peers ein Optimierungsproblem, welches in der Implementierung mit einer sehr einfachen Heuristik gelöst wurde. Die Liste aller passiver Supporter wird aufsteigend nach deren minimal notwendiger Kapazität sortiert. Aus dieser Liste werden beginnend mit dem ersten Eintrag Supporter aktiviert. Dieser Algorithmus garantiert keine optimale Allokation (es können unterversorgte Peers übrig bleiben, obwohl eine andere Allokation dies eventuell hätte verhindern können), sorgt aber dafür, dass zumindest eine Teilmenge der verbleibenden unterversorgten Peers möglichst schnell Unterstützung erhalten.

Die interne Repräsentation von Peers und Supporter

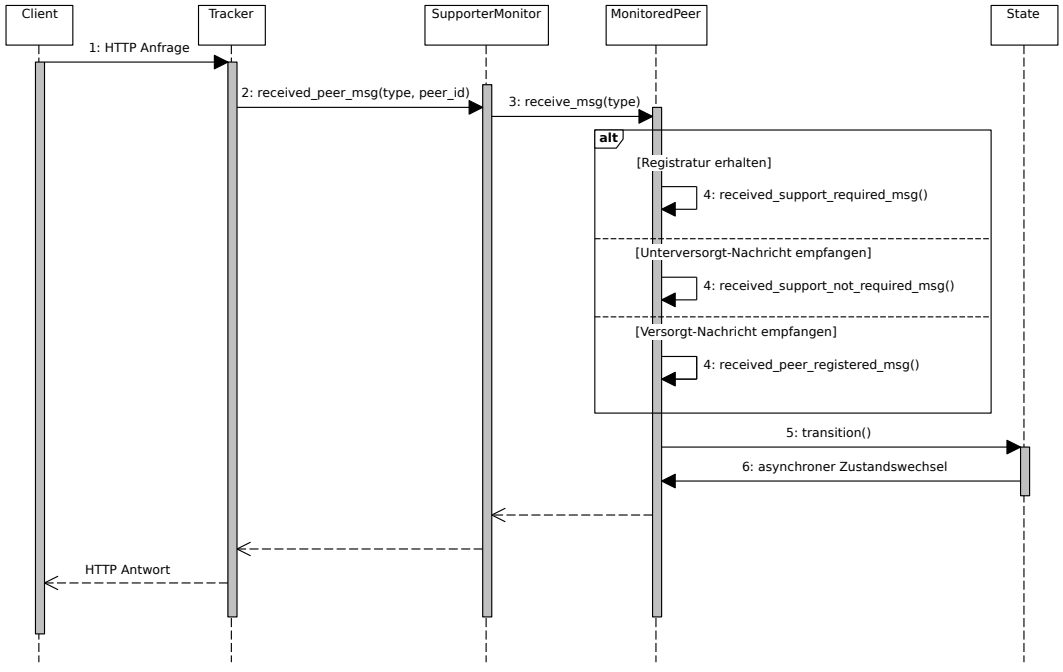


Abbildung 5.4: Visualisierung der internen Abläufe bei der Verarbeitung client-seitiger Nachrichten

Der Überwachungsmechanismus muss zu jedem Zeitpunkt Wissen über die sich im Overlay befindlichen Peers und Supporter Server besitzen. Dieses Wissen wird in Objekten der Klassen `MonitoredPeer` und `MonitoredSupporter` vorgehalten. Die Funktionalität beider Klassen geht jedoch über die reine Datenaggregation hinaus. `MonitoredPeer` ist bspw. Empfänger von Nachrichten, die über den korrespondierenden Peer im Overlay an die Überwachungskomponente gesendet werden.Bei Ankunft einer Nachricht wird der interne Objektzustand aktualisiert, was bspw. eine Zustandsveränderung zur Folge haben könnte. Die konkrete Zustandsveränderung ereignet sich zwar in den Zustandsklassen selbst, jedoch benötigen diese Wissen über den assoziierten Peer, wie bspw. die Anzahl der Unterversorgt-Nachrichten oder Informationen darüber, ob spezifische Zeitintervalle überschritten worden sind und dergleichen. Abbildung 5.4 zeigt den sequentiellen Ablauf bei Nachrichtenempfang.

Dies gilt in ähnlicher Weise für `MonitoredSupporter`. Diese Klasse verwaltet den Zustand eines Supporters, insbesondere die Allokation von Peers zu dem korrespondierenden Supporter Server.

Ferner stellen beide Klassen über öffentliche Methoden die Möglichkeit bereit, den Objektzustand aus Sicht des `SupporterMonitor` in einfacher Weise abzufragen.

Die Anbindung eines XML-RPC-Client

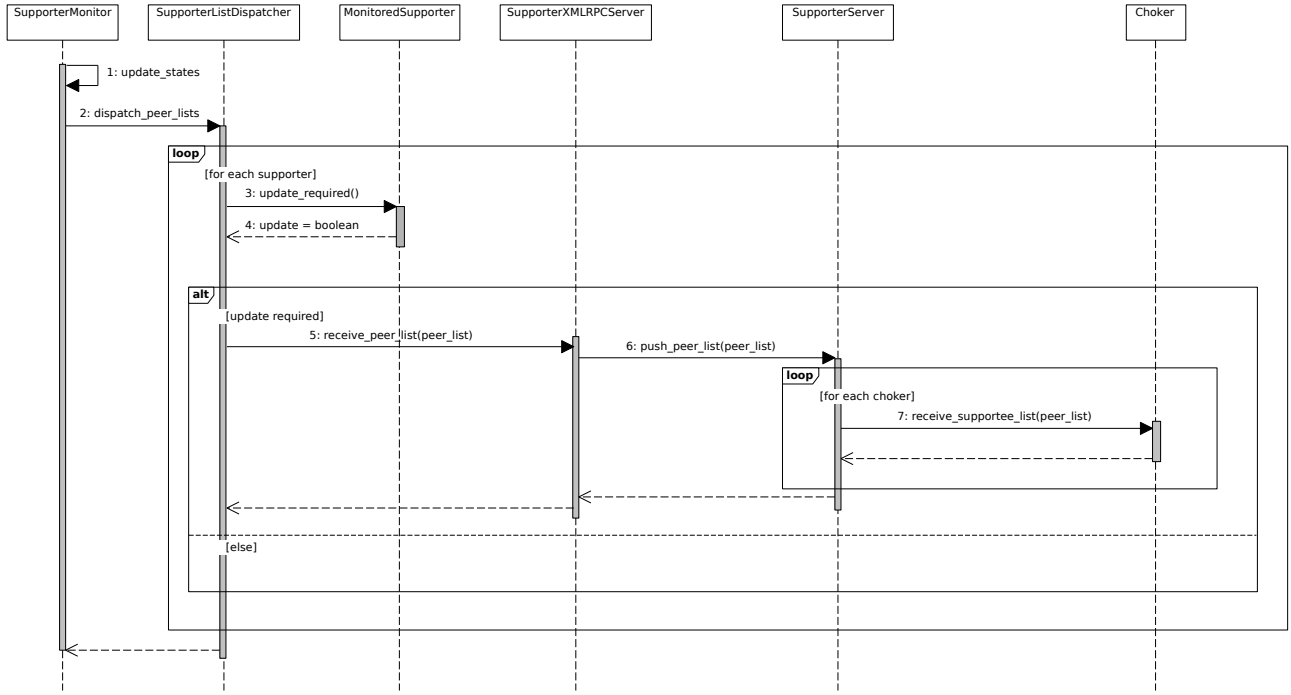


Abbildung 5.5: Zustellung aktualisierter Peer-Listen an korrespondierende Supporter

Die Kommunikation zwischen Überwachungsmechanismus und Supporter wird wie in Abschnitt 5.2.1 beschrieben über eine XML-RPC-Schnittstelle realisiert. Der Überwachungsmechanismus ist dabei der Initiator einer Anfrage und benötigt dementsprechend eine Client-seitige Anbindung für besagte Schnittstelle. Diese Anbindung realisiert die Klasse `SupporteeListDispatcher`. Für jeden Supporter, der sich an der Überwachungskomponente registriert hat, muss ein Proxy unter Verwendung der URI dieses Supporters erzeugt werden. Über diese Proxies erfolgt die Kommunikation mit den Supporter Servern. Die Klasse selbst stellt zwei öffentliche Methoden bereit (Abbildung C.2), welche die XML-RPC-Schnittstelle benutzen, um zu prüfen, welche Supporter noch aktiv im Overlay sind (`query_all_supporters`) und die Versendung von Peer-Listen anzustoßen (`dispatch_peer_lists`). Inaktive Supporter werden entsprechend gekennzeichnet, so dass sie beim nächsten synchronen Zustandswechsel aus der Überwachungskomponente entfernt werden können. Abbildung 5.5 zeigt den sequenziellen Ablauf beim Versenden aktualisierter Peer-Listen an Supporter Server.

Notwendige Modifikationen des NextShare-Kerns

Die Integration der Supporter-Strategie mit NextShare erfordert die Modifikation einiger zentraler Komponenten. NextShare stellt bereits einen internen Tracker mit der Klasse `BaseLib.Core.BT1.BitTornado.track.Tracker` zur Verfügung. Diese zentrale Instanz kann multiple Schwärme bedienen und besitzt Wissen über alle Peers, die sich in diesen Schwärmen aufhalten. Ferner stellt sie über diverse HTTP-Ressourcen eine Schnittstelle zur Kommunikation über HTTP für Peers bereit, deren Erweiterung bereits in Abschnitt 5.2.1 diskutiert wurde. Für die Integration der Supporter-Strategie ist es nötig, bereits bestehende HTTP-Ressourcen durch die in Abschnitt 5.2.1 definierten Ressourcen zu erweitern. Dies geschieht über eine simple Auswahl anhand des Pfades, der mit jeder HTTP-Anfrage mitgeliefert wird. Entsprechend der adressierten HTTP-Ressource erfolgt die Auswertung der HTTP-Anfrage und

Weiterleitung der übermittelten Nachricht an den `SupporterMonitor`. Die Anbindung der neuen HTTP-Ressourcen erfolgt in der Methode `Tracker.get`.

Der interne Tracker verfügt überdies über eine Übersichtsseite im HTML-Format. Diese wurde ebenfalls modifiziert, so dass der aktuelle Zustand des `SupporterMonitor` inklusive aller zu beobachtenden Peers und Supporter Servern dort ebenfalls angezeigt wird. Die Generierung des HTML-Anteils erfolgt in der Klasse `MonitorState`. Die Integration erfolgt in der Methode `Tracker.infopage`.

Modul `Supporter`

Der Supporter Server ist im Wesentlichen ein parametrisierter NextShare Client, der auf Konsolenebene arbeitet. Abbildung 5.6 zeigt die beteiligten Klassen, die sich im Modul `Supporter.py` befinden. Entscheidend ist, die maximale Anzahl von Upload-Slots global auf die Anzahl an Peers zu setzen, die der Supporter unterstützten kann. Geschieht dies nicht, so nimmt NextShare einen Standard-Wert an. Dieser Wert ist für einen regulären Peer zwar sinnvoll gewählt, aber für einen Supporter Server, der typischerweise mit einer erhöhten Bandbreite ausgestattet ist, verteilt sich bei wenigen zulässigen Verbindungen die hohe Bandbreite ineffizient auf wenige Datenübertragungen, obwohl der Supporter wesentlich mehr Peers bedienen könnte.

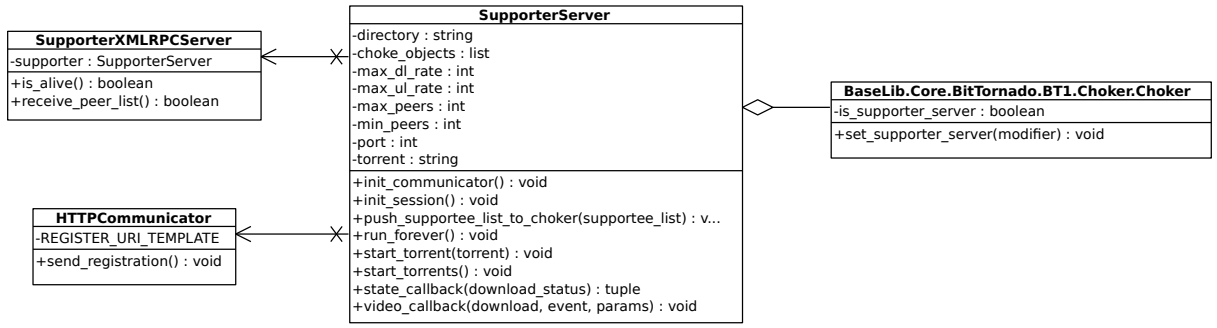


Abbildung 5.6: Klassendiagramm des Moduls `Supporter` (Detailansicht)

Die Anbindung der Kommunikationsschnittstellen

Der Supporter nutzt einen HTTP Client (siehe `HTTPCommunicator`), um sich an der Überwachungskomponente zu registrieren. Die Registrierung ist die einzige Interaktion mit dem Überwachungsmechanismus, der vom Supporter selbst ausgeht. Der Supporter realisiert die in Abschnitt 5.2.1 vorgestellte XML-RPC-Schnittstelle in der Klasse `SupporterXMLRPCServer`. Diese Klasse stellt die notwendigen Methoden für die Kommunikationsrichtung von Überwachungskomponente zu Supporter bereit.

Notwendige Modifikationen des NextShare-Kerns

Die Umsetzung des Verbindungsmanagements geschieht in der Klasse `BaseLib.Core.BT1.BitTornado.Choker`. Diese Klasse realisiert den Choke-Algorithmus und entscheidet darüber, über welche Verbindungen der Supporter Daten sendet. Dies geschieht auf Ebene eines einzelnen Downloads, so dass für jeden Schwarm, an dem der Supporter aktiv teilnimmt, ein dediziertes `Choker`-Objekt existiert. Der Supporter leitet Peer-Listen, die über die XML-RPC-Schnittstelle eingehen, an diese dedizierten `Choker`-Objekte weiter. Die Umsetzung des neuen Verbindungsmanagements erfolgt mit dem nächsten Aktualisieren der Peer-Zustände (vgl. `rechoke`-Methode, Abschnitt 2.2.3). Die Integration des Supporters mit NextShare erfordert es, zwischen der regulären Ausführung des Chokers (eine Kombination aus Tit-for-Tat und Give-to-Get, vgl. Abschnitt 2.2.3 und 2.3.3) und der Restriktion auf unterversorgte Peers zu unterscheiden.

5.3 Implementierung eines Video-on-Demand Client-Emulators

Für die Erprobung des in dieser Arbeit vorgestellten Ansatzes ist es nötig, eine Vielzahl von Clients in einem verteilten Testbed zu starten, die eine Videodatei herunterladen und gleichzeitig die Wiedergabe der Videoinhalte emulieren. Der NextShare-eigene Client kann hierfür nicht verwendet werden, da er eine grafische Oberfläche voraussetzt und überdies keine Statistiken aggregiert, die für die Umsetzung des Bandbreitenmanagements und der Supporter-Strategie benötigt werden.

Die Anforderungen für eine Emulation der Video-Wiedergabe sind minimal:

- Der Video-on-Demand Client Emulator muss über die Konsole gestartet werden können und bezüglich Port und weiterer Parameter konfigurierbar sein.
- Der Emulator muss Statistiken zu den heruntergeladenen Inhalten aggregieren und bei Bedarf an einen entfernten Empfänger über eine standardisierte Schnittstelle versenden können.
- Der Emulator muss fähig sein, mit der Datenerhebungskomponente des Bandbreitenmanagements kommunizieren zu können.
- Der Emulator muss fähig sein, mit der Überwachungskomponente der Supporter-Strategie kommunizieren zu können.
- Die Emulation des Videos erfolgt anhand einer simplen Heuristik: Mit der durchschnittlichen Bitrate in Sekunden des Videos erfolgt das Lesen der Videodaten von dem korrespondierenden Datenstrom. Anzumerken ist an dieser Stelle, dass die Bitrate sich von Frame zu Frame dramatisch ändern kann, so dass die Auswirkungen von Höchstwerten in den Bitraten der Frames nicht von der Emulation berücksichtigt werden. Auf die Abspielzeit hat dies jedoch keine Auswirkungen, so dass diese Vereinfachung wohlwollend in Kauf genommen wird.

5.3.1 Notwendige Erweiterungen der NextShare Code-Basis

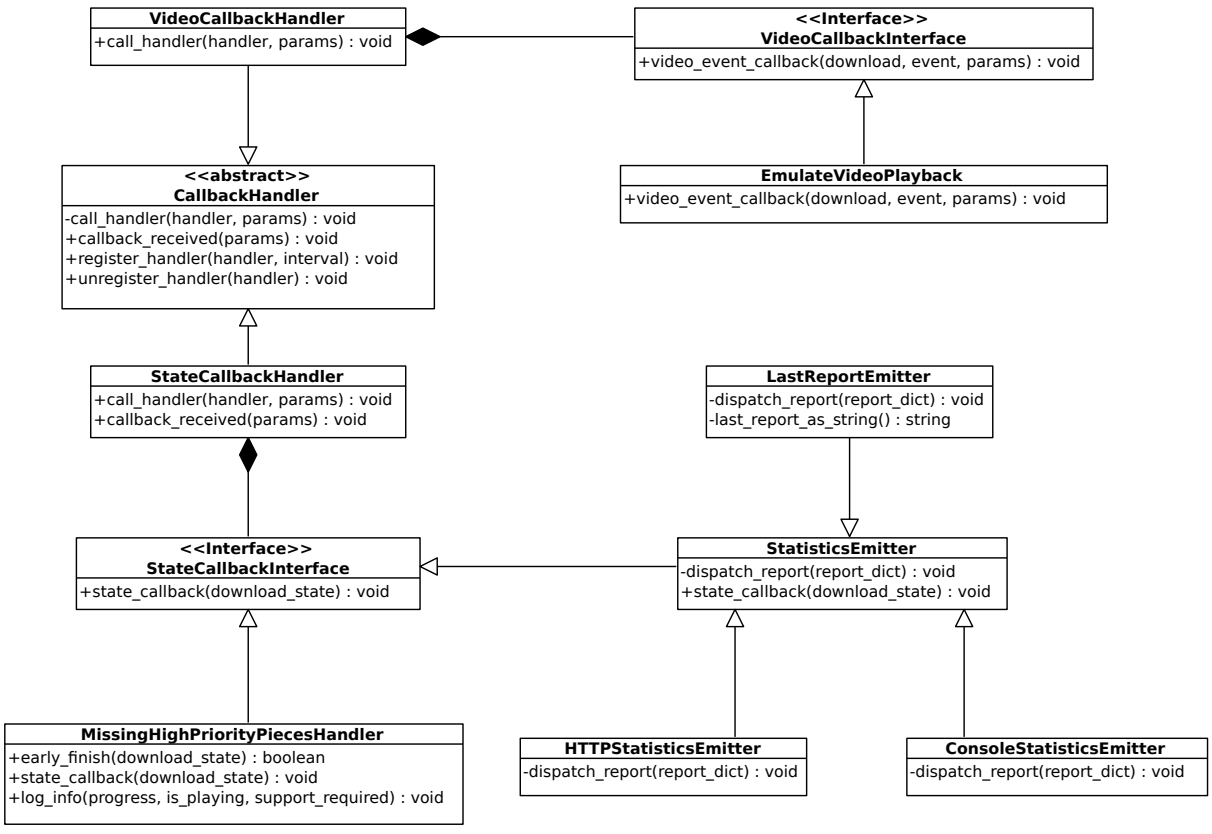


Abbildung 5.7: Klassendiagramm des Moduls `ClientEmulation.CallbackHandlers`

Der NextShare Client wird im Wesentlichen um drei Module erweitert, die alle notwendigen Erweiterungen kapseln:

- Modul `AccessProfile.py`: Dieses Modul beinhaltet die Behandlungsmethode für die XML-RPC-Schnittstelle, die ein Zugangsprofil des Bandbreitenmanagements empfängt und dieses durch client-seitige Ratenlimitierung umsetzt.
- Modul `CallbackHandlers.py`: Dieses Modul beinhaltet die Implementierungen für die im Folgenden angesprochenen Erweiterungen.
- Modul `Client.py`: Dieses Modul nutzt die NextShare API, um einen parametrisierten Client auf Konsolenebene zu starten, der keine grafische Oberfläche benötigt.

Möglichkeiten der Erweiterung

Die NextShare Programmbibliothek bietet software-seitige Mittel an, um eigenen Code in den NextShare-Kern einzuführen, ohne dabei die bestehende Code-Basis modifizieren zu müssen. Im Wesentlichen handelt es sich hierbei um die folgenden Einstiegspunkte:

- `DownloadStartupConfig.set_video_event_callback`: Definiert eine Rücksprungmethode, die aufgerufen wird, sobald ein neuer Video-Event im NextShare-Kern geworfen wurde.
- `Download.set_state_callback`: Definiert eine Rücksprungmethode, die periodisch aufgerufen wird und über die Statistiken zu einem spezifischen Download bezogen werden können.

Der Nachteil dieser Rücksprungmethoden ist, dass pro aktivem Download jeweils nur eine Rücksprungmethode festgelegt werden kann. Des Weiteren ist die Aggregation der Statistiken für einen Download mit einer gewissen Unschärfe behaftet: Die untere Intervallgrenze für Aufrufe an die Rücksprungmethode, die über `set_state_callback` festgelegt wurde, beträgt 1 Sekunde, so dass bspw. Statistiken über Peering-Verbindungen, die zwischen zwei konsekutiven Aufrufen terminiert wurden, nicht länger verfügbar sind. Die Mächtigkeit dieser Einstiegspunkte ist selbstverständlich limitiert, so dass komplexere Modifikationen nicht ohne Eingriffe in die Code-Basis von NextShare erfolgen können. Für die Umsetzung der angeführten Anforderungen ist eine Bereitstellung von spezifischen Rücksprungmethoden allerdings ausreichend.

Abbildung 5.7 zeigt die Klassenhierarchie der client-seitigen Erweiterungen. Um mehrere Rücksprungmethoden von den angeführten Einstiegspunkten in unterschiedlichen Periodika aufzurufen, wurde mit der Klasse `CallbackHandler` eine zusätzliche Indirektionsebene geschaffen. `CallbackHandler` hat konkrete Ausprägungen für Download-Statistiken (`StateCallbackHandler`) und für die Behandlung von Video-Events (`VideoCallbackHandler`). Die Basisklasse `CallbackHandler` stellt Methoden bereit, um entsprechende Handler-Klassen, die Rücksprungmethoden implementieren, zu registrieren. Diese Handler-Klassen implementieren – je nach Typ des Rücksprungs – entweder die Schnittstelle `StateCallbackInterface` oder `VideoCallbackInterface`.

Emulation der Video-Wiedergabe

Die Emulation der Video-Wiedergabe erfolgt in der Klasse `EmulateVideoPlayback`. Die Rücksprungmethode dieser Klasse wird aufgerufen, sobald das Video gestartet wird (Event `VODEVENT_PLAY` und wenn die Wiedergabe pausiert (Event `VODEVENT_PAUSE`) oder wieder angestoßen wird (Event `VODEVENT_RESUME`).

Aggregation client-seitiger Statistiken

Die Aggregation client-seitiger Statistiken erfolgt in der Klasse `StatisticsEmitter`. `StatisticsEmitter` selbst ist eine abstrakte Klasse, deren konkrete Ausprägungen unterschiedliche Wege implementieren, um die aggregierten Daten zu emittieren:

- Ausgabe auf die Konsole (`ConsoleStatisticsEmitter`)
- Ausgabe der letzten verfügbaren Daten (`LastReportEmitter`)

- Versenden an einen entfernten Empfänger über eine HTTP-basierte Schnittstelle (`HTTPStatisticsEmitter`). Im Kontext dieser Arbeit ist der Empfänger die Datenerhebungskomponente des adaptiven Bandbreitenmanagements.

Statistiken werden in einem Python Dictionary gespeichert und können für den Versand per HTTP mit Python-Bordmitteln serialisiert (`pickle`) und komprimiert werden (`zlib`). Appendix D gibt Aufschluss über die clientseitigen Statistiken.

5.3.2 Zusammenspiel mit der Überwachungskomponente der Supporter-Strategie

Die Inferenz, ob Unterstützung notwendig ist oder nicht, erfolgt in der Rücksprungmethode der Klasse `MissingHighPriorityPiecesHandler`. Diese Klasse benutzt eine konkrete Instanz von `TrackerCommunicator` (Implementierung in `Client.py`). `TrackerCommunicator` stellt öffentliche Methoden bereit, um entsprechende Registrierungs- bzw. Support-Nachrichten an die Überwachungskomponente senden zu können.

6 Methodik der Evaluation

Für die Evaluation eines verteilten Systems haben sich in der Forschung zwei hauptsächliche Evaluationsmethoden herauskristallisiert: Die Leistungsbewertung in einer simulierten Laufzeitumgebung oder aber der Einsatz eines Live-Testbeds zur empirischen Bewertung der Performanz. Beide Ansätze unterscheiden sich stark voneinander und haben jeweils Vor- und Nachteile, so dass sich deren Einsatz situationsbedingt begründen lässt.

Grundsätzlich ist eine Simulation dafür geeignet, möglichst schnell eine konkrete Idee umzusetzen, um sie prototypisch erproben zu können. Bedingt durch die dem Simulator inhärente Abstraktion von einer realen Laufzeitumgebung lassen sich Beispielszenarien schnell durchrechnen und analysieren. Aus diesem Grund eignet sich diese Evaluationsmethode insbesondere für die Analyse der Auswirkungen unterschiedlicher Parametrisierungen der Umgebung bzw. des Mechanismus. Die Ergebnisse müssen jedoch generell kritisch betrachtet werden, weil sich das Gesamtsystem durch die starke Abstraktion der Simulation anders verhält, als eine Testumgebung, die durch Hardware gestellt wird. Dennoch eignet sich das einfache Modell eines simulativen Ansatzes, um Tendenzen in den Daten zu erkennen und günstige Parametereinstellungen zu identifizieren.

Die Evaluation unter Zuhilfenahme einer realen Testumgebung, eines sogenannten Live-Testbeds, ist bezüglich der Laufzeit eines Szenarios an die Echtzeit gebunden. Bedingt durch diesen Umstand ist diese Evaluationsmethode deutlich zeitaufwändiger. Die Komplexität der Durchführung steigert sich ebenfalls durch die Installation und Konfiguration eines passenden Systems im Hinblick auf die Applikation, die darauf evaluiert werden soll. Nicht zuletzt müssen Kosten bezüglich der Anschaffung und Instandhaltung der notwendigen Hardware in Kauf genommen werden, wobei heutzutage sicher auch das Cloud Computing eine interessante Alternative zur Bereitstellung eines eigenen Testbed-Clusters ist.

Während Simulationen kostengünstiger und gemessen am zeitlichen Aufwand der Durchführung deutlich attraktiver erscheinen, so lässt sich doch festhalten, dass ein neuer Mechanismus auf langfristige Sicht in einer realen Testumgebung empirisch evaluiert werden muss, um die Einsatzfähigkeit in praktischen Szenarien diskutieren zu können. Aus diesem Grund fokussiert diese Arbeit die Leistungsbewertung des adaptiven Bandbreitenmanagements zusammen mit der Supporter-Strategie in einem solchen Live-Testbed. Die zeitliche Komplexität wird dabei wohlwollend in Kauf genommen. Sofern diese vorliegen, greifen wir auf bereits bekannte Parametrisierungen, die in Simulation ermittelt wurden, zurück. Im Zuge der Vergleichbarkeit sei an dieser Stelle insbesondere auf [10] verwiesen.

6.1 Evaluationsplattform

Die Evaluation greift auf das deutschlandweit verteilte Testbed German-Lab[1] (G-Lab) zurück. G-Lab ist konzipiert als Studien- und Experimentalplattform für das Internet der Zukunft. Die Basis von G-Lab bilden Sun Fire X4150/X4250 Server, die von beteiligten Partneruniversitäten für den Zweck der Evaluation neuer Protokolle oder verteilter Systeme bereitgestellt werden. Die Anbindung an das Netzwerk erfolgt homogen über Gigabit-bandige Netzanschlüsse. Die Maschinen selbst laufen auf der PlanetLab Software[2], die auf einem Linux-Derivat basiert und Virtualisierung nutzt, so dass mehrere Forscher ungehindert auf denselben Maschinen arbeiten und sich die zur Verfügung stehenden Ressourcen teilen können. Der interessierte Leser sei auf [58] verwiesen, um Details zu den Arbeitsschwerpunkten dieses Clusters zu erfahren.

6.2 Topologie und Systemarchitektur

Die zu evaluierenden Szenarien beschränken sich auf eine möglichst einfache Topologie, die sich im Wesentlichen aus drei unterschiedlichen, virtuellen ISP-Domänen und einer Server-Farm zusammensetzt. Virtuell bedeutet nichts

[1] http://www.german-lab.de
[2] http://www.planet-lab.org

anderes, als dass für die verwendeten G-Lab Maschinen unabhängig von ihrem physikalischen Standort eine Gruppierung in mehrere ISP-Domänen auf Basis einer einfachen Zugehörigkeitstabelle erfolgt. Die in Abbildung 6.1 gezeigte Topologie stellt die Grundkonfiguration für alle durchzuführenden Experimente dar. Abweichungen zu der hierin beschriebenen Topologie lassen sich den entsprechenden Experimentbeschreibungen entnehmen, sofern notwendig.

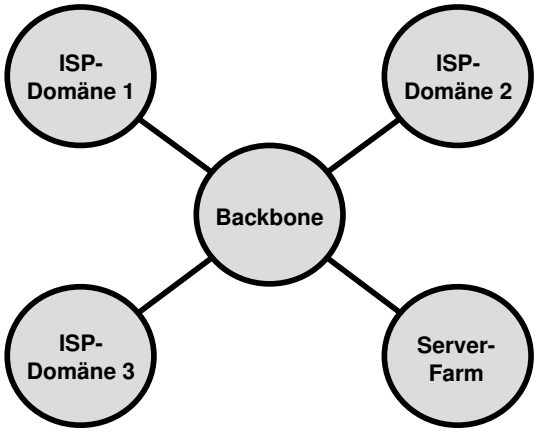


Abbildung 6.1: Vereinfachte Darstellung der virtuellen Topologie

Die Topologie soll die Abbildung eines hybriden Peer-to-Peer Video-on-Demand Systems ermöglichen. Entitäten innerhalb dieses Systems sind der Content Provider, der ISP und letzten Endes die Peers, die sich für die offerierten Inhalte interessieren. Jede dieser Interessengruppen verfolgt unterschiedliche Ziele und agiert dementsprechend.

- Content Provider: Der Content Provider verfolgt das Ziel, seine Daten unter einer Menge von interessierten Benutzern zu verteilen. Der primäre Distributionsweg erfolgt von Peer zu Peer. Hierzu kann sich zu Beginn eines Experiments eine gewisse Anzahl von initialen Peers im System befinden, die den entsprechenden Inhalt bereits vollständig heruntergeladen haben und somit anderen Peers zur Verfügung stellen können. Sofern das Overlay zu einem gegebenem Zeitpunkt nicht leistungsfähig genug ist, können Peers den sekundären Distributionsweg nutzen, um entsprechende Inhalte von Servern zu laden, die der Anbieter mit Hilfe einer adaptiven Server-Strategie verwaltet. Dieser Weg soll lediglich als Ausweichlösung dienen, denn es liegt im Bestreben des Anbieters, die Last auf diesen Servern möglichst gering zu halten. Initiale Peers befinden sich innerhalb der Domäne eines Netzanbieters, während die Server des Content Provider in einem separaten AS lokalisiert sind. Dies spiegelt insbesondere die typische Verteilung im Internet wieder, denn auch dort liegen Server-Farmen eines CDN selten innerhalb der Domäne eines Netzanbieters, sondern meist außerhalb. Durch die Lokalisierung in einem eigenen AS ist jeder Datenverkehr, der zwischen Peer und Server erfolgt, als Inter-ISP-Datenverkehr einzustufen.
- ISP: Aus Gründen der Simplizität beschränkt sich der Einzugsbereich eines ISPs auf ein virtuelles AS (ISP-Domäne). Der Netzanbieter ist letzten Endes die Instanz, die das adaptive Bandbreitenmanagement einsetzt, um Kosten für Inter-ISP-Verbindungen zu reduzieren. Die Topologie sieht aus Komplexitätsgründen drei separate AS vor, die jeweils unter der Kontrolle eines ISPs stehen. Die Aufteilung der Peers auf diese AS erfolgt in Abhängigkeit vom konkreten Szenario und wird in der entsprechenden Beschreibung thematisiert.
- Peers: Die Peers repräsentieren die Benutzer des Systems. Ein Peer ist der Domäne eines ISPs zugeordnet. Ferner bedeutet dies, dass der zugehörige Benutzer bzgl. seines Zugangsprofils unter der Kontrolle des ISPs steht. Nutzt der ISP adaptives Bandbreitenmanagement, dann kann sich das Zugangsprofil des Benutzers im Laufe seines Aufenthalts im System ändern. Für einen Benutzer ist es im Wesentlichen von Belang, dass er gewünschte Inhalte mit einer Mindestdatenrate beziehen kann. Durch die hybride Systemarchitektur soll ihm ermöglicht werden, im Falle von Performanz-Verlusten des P2P-Systems einen Einbruch der Transferrate zu verhindern. Dies gelingt durch die Nutzung von Servern des Content Providers. Diese Möglichkeit soll grundsätzlich jedem Peer im System offenstehen, so dass jeder Teilnehmer des Overlays die in Abschnitt 5.2.1 diskutierte Protokoll-Erweiterung spricht. Heruntergeladene Datenblöcke bietet ein Peer sofort anderen interessierten Peers zum Austausch an. Die Wahl der Zugangsprofile für Peers erfolgt je nach Szenario homogen respektive heterogen. Peers unterscheiden sich in ihrem Verhalten hinsichtlich der Zeit, die sie im System verweilen.

Für konkrete Parametrisierungen und deren mögliche Auswirkungen auf das System sei auf Abschnitt 6.4 verwiesen.

6.3 Testdurchführung

Die Arbeit mit einem weit verteilten Cluster wie dem G-Lab System erfordert einen präzise koordinierten Ablauf der Experimente. Die Koordination erfolgt durch eine Reihe von Python-Skripten, die unterschiedliche Aufgaben übernehmen:

- `glab_master.py`: Dieses Skript läuft auf einem Rechner, der die Rolle des Koordinators übernimmt. Es stellt Möglichkeiten bereit, um eine Vielzahl von G-Lab Maschinen für die Durchführung eines Experiments einzurichten. Ferner koordiniert es die Instanzierung von Prozessen auf einer entsprechenden Maschine und bietet eine API an, mit der ein Benutzer individuelle Szenarien programmatisch konstruieren kann. Das Skript nutzt zur Kommunikation mit G-Lab Maschinen eine XML-RPC Anbindung, die auf Basis von HTTP realisiert ist.
- `glab_slave.py`: Dieses Skript wird von `glab_master.py` auf einer entfernten G-Lab Maschine gestartet und kann fortan über eine XML-RPC Verbindung Steuerungsbefehle erhalten, die es zur Instanzierung diverser Prozesse veranlassen. Das Skript kann ferner darüber Auskunft geben, wieviele Prozesse zu gegebenem Zeitpunkt noch aktiv sind.
- `experiment_*.py`: Individuelle Skripte, welche die programmatische Durchführung der in dieser Arbeit diskutierten Szenarien beinhalten. Diese Skripte nutzen die API von `glab_master.py` und sind der Ausgangspunkt für das Starten eines Experiments.
- `generate_schedule.py`: Dieses Skript generiert einen Ablaufplan, der angibt, zu welchem Zeitpunkt und auf welcher Maschine eine NextShare-Instanz gestartet werden soll. Das Skript kann den Ablaufplan unter Berücksichtigung diverser Ankunftsprozesse erstellen.

Sobald instanziert, laufen die Skripte ohne Beteiligung des Benutzers ab. Da die Evaluationsszenarien in einer Umgebung ablaufen, in der potentiell sehr viele Nebeneffekte auftreten können, muss eine Testwiederholung dafür sorgen, die Ergebnisse mit einer gewissen Konfidenz wiederzugeben. Soweit nicht anders vermerkt, erfolgt eine dreimalige Wiederholung eines Szenarios. Die Darstellung der Ergebnisse erfolgt anhand von Mittelwerten und Standardabweichungen.

6.4 Parameter

Evaluationsszenarien sind über eine Reihe von Parametern konfigurierbar. Die Beschreibung dieser Parameter ist Gegenstand dieses Abschnitts. Unterschiedliche Parametrisierungen haben ebenso unterschiedliche Auswirkungen auf die Szenarien. Aus diesem Grund geht die Beschreibung der Parameter ebenfalls auf zu erwartende Auswirkungen verschiedener Werte ein.

Die Beschreibung der Szenarien in Kapitel 7 und 8 verweist explizit auf die in diesem Abschnitt definierten Parameter. Verzichtet die Szenarienbeschreibung auf einen konkreten Parameter, so ist eine Festlegung für die Testdurchführung nicht notwendig.

6.4.1 Szenarien-bezogene Parameter

Laufzeit des Szenarios

Die Laufzeit des Szenarios gibt an, bis zu welchem Zeitpunkt Peers dem Schwarm beitreten. Überschreitet die Lebenszeit eines Peers die Laufzeit des Szenarios, so hat dies keine Auswirkungen auf den Peer. Ein Peer bleibt grundsätzlich so lange im Schwarm aktiv, wie es durch die Parametrisierung seines Verhaltens vorgegeben ist.

6.4.2 Schwarm-bezogene Parameter

Anzahl Peers im Schwarm

Dieser Parameter beschreibt die Anzahl der Peers, die zu einem gegebenen Zeitpunkt in einem Schwarm aktiv sind. Die Anzahl ist abhängig von dem verwendeten Ankunftsprozess.

Initiale Anzahl an statischen Servern

Ein statischer Server ist ein Peer, der bereits die Inhalte vollständig heruntergeladen hat und sie anderen Peers anbieten kann. Ein statischer Server nimmt bereits zu Beginn eines Evaluationsszenarios aktiv am Schwarm teil. In den quantitativen Studien zur Supporter-Strategie spielt dieser Parameter eine Rolle, um Basisvergleiche in die Leistungsbewertung einzubeziehen. Ein statischer Server verlässt den Schwarm erst nachdem das Szenario beendet ist.

Initiale Anzahl an Supporter Servern

Ein Supporter Server ist ein modifizierter Peer, der bereits die Inhalte vollständig besitzt. Er nimmt bereits zu Beginn eines Evaluationsszenarios am Schwarm teil. Ein Supporter Server befindet sich zunächst in einem inaktiven Zustand. Durch die Allokation eines Peers zu einem Supporter Server wird dieser aktiviert und trägt nun zur Distribution der Daten bei. Supporter Server können durch die Überwachungskomponente der Supporter-Strategie wieder inaktiviert werden, sofern keine weitere Zuteilung eines Peers zu besagtem Supporter Server besteht. Supporter Server verlassen den Schwarm erst nachdem das Szenario beendet ist.

Ankunftsprozess

Der Ankunftsprozess beschreibt, nach welchem zeitlichen Muster neue Peers, insbesondere Leecher, dem Schwarm beitreten. Neu beigetretene Peers nehmen sofort aktiv innerhalb des Schwarms teil. Ein neu beigetretener Leecher beginnt unmittelbar mit dem entsprechenden Download und bietet selbst Daten an, sobald er den ersten Chunk komplettiert hat. Ein neu beigetretener Seeder beginnt unmittelbar mit dem Offerieren seiner Daten.

Der Ankunftsprozess aller Szenarien ist durch einen homogenen Poisson-Prozess beschrieben. Dies ist ein stochastischer Prozess, dessen Zuwächse einer Poisson-Verteilung folgen. Diese Verteilung basiert auf dem Parameter λ, der den Zuwachs pro Zeiteinheit beschreibt. Die Zeiteinheit beträgt in allen Szenarien eine Minute. λ beschreibt damit nichts anderes, als den Erwartungswert für neue Peer-Ankünfte über dem Zeitintervall einer Minute. Die Zeiten zwischen konsekutiven Peer-Ankünften sind exponentialverteilt. Die Anzahl der Peers, die sich zu einem gegebenen Zeitpunkt im Schwarm befinden, kann durch $N(t) = \lambda \cdot \text{Videolänge}$ ermittelt werden. In dieser Form ist $N(t)$ unabhängig von der Absprungrate. Möchte man diese miteinkalkulieren, so genügt es, $N(t)$ mit dem Mittelwert der Absprungrate zu multiplizieren. Anzumerken sei, dass $N(t)$ erst korrekte Werte liefert, wenn das System seine mittlere Arbeitslast erreicht hat. Dies ist dann der Fall, wenn die ersten Peers, die das Video potentiell bis zum Ende sehen, den Schwarm wieder verlassen.

Zugangsprofile der Benutzer

Dieser Parameter beschreibt das Zugangsprofil eines Benutzers, sprich, seine Up- und Download-Bandbreite. Die Wahl der Download-Bandbreite muss mindestens der Bitrate des verwendeten Videos entsprechen, da sich ansonsten inhärent durch den Download-Engpass Verzögerungen ergeben. Das Bandbreitenmanagement modifiziert lediglich die Upload-Bandbreite, nicht jedoch die Download-Bandbreite. Im Rahmen der Vergleichbarkeit, insbesondere der Supporter-Strategie, mit [10] erfolgt die Parametrisierung der Bandbreiten exakt oder nahe den Werten aus dieser Arbeit. In den betrachteten Szenarien ist das Zugangsprofil grundsätzlich asymmetrisch ausgelegt und es gilt: Upload-Bandbreite < Download-Bandbreite.

Video-/Audio-Kodierung

Bezeichner	Qualität	Spielzeit	Auflösung	Video-Codec	Audio-Codec	Dateigröße
480p Testvideo, kurz	SD	5:00	756x540	DivX (476 kb/s)	AAC (64 kb/s)	19,3 MB
480p Testvideo, lang	SD	15:00	756x540	DivX (476 kb/s)	AAC (64 kb/s)	58,0 MB

Tabelle 6.1: Tabellarische Darstellung verwendeter Video-Testdateien

Tabelle 6.1 zeigt die Video-Dateien, die im Rahmen der Leistungsbewertung Verwendung finden. Die Kodierung der Videos ist an SD-Qualität ausgerichtet. Die Leistungsfähigkeit eines konkreten Video-Codecs ist in der Evaluation von geringerer Bedeutung. Der in Abschnitt 5.3 beschriebene Video-on-Demand Client-Emulator interpretiert den empfangenen Datenstrom nicht, sondern prüft lediglich, ob mit einer konstanten Transferrate Daten gelesen werden können (die Transferrate entspricht der gemittelten Bitrate).

Chunk-/Block-Größe eines Torrents

Die Chunk- bzw. Block-Größe[3] hat einen großen Einfluss auf die Performanz des Overlays. Ist sie zu klein gewählt, so steigt der Overhead in Relation zu den Nutzdaten, die pro Transfer getauscht werden, sehr stark an. Auf der anderen Seite können parallele Downloads sehr großer Blöcke dazu führen, dass die Skalierbarkeit des Systems in Mitleidenschaft gezogen wird. Eine sinnvolle Wahl der Chunk-Größe kann in Abhängigkeit der Upload-Kapazität eines Senders festgelegt werden. Nach bestem Wissen des Autors existieren jedoch keine Studien über die Auswirkungen einer derartigen Parametrisierung. Für die Leistungsbewertung erfolgt daher die Annahme, dass ein Wert gleich der Upload-Kapazität des schwächsten Zugangsprofils innerhalb eines konkreten Szenarios sinnvoll ist.

6.4.3 Client-bezogene Parameter

Peer-Verhalten

Dieser Parameter beschreibt das Absprungverhalten eines Peers. Peers können entweder frühzeitig, sprich vor Beendigung der Video-Wiedergabe, den Schwarm verlassen (egoistisches Verhalten), oder aber nach Beendigung der Video-Wiedergabe für eine gewisse Zeit weiterhin aktiv am Schwarm teilnehmen (altruistisches Verhalten). Die konkrete Angabe des Absprungzeitpunkts erfolgt über den Parameter Verweilzeit.

Verweilzeit

Die Verweilzeit gibt an, wie lange sich ein Peer im Schwarm befindet. Die Verweilzeit kann einen konkreten Zeitwert annehmen, oder sich in Abhängigkeit des Peer-Verhaltens berechnen:

- Egoistische Peers: Diese Peers verlassen den Schwarm frühzeitig. Im Mittel verweilt ein Peer über 50% der Videolänge im System. Dies deckt sich mit Beobachtungen in echten Video-on-Demand-Systemen [59]. Die konkrete Verweilzeit berechnet sich nach $t_{\text{Verweil}} = \text{Videolänge} \cdot z$, wobei z aus einer uniformen Gleichverteilung $\mathcal{U}(0,1)$ stammt.
- Altruistische Peers: Diese Peers verweilen länger als notwendig im System. Die Zeit, die sie über die Videolänge hinaus im System bleiben, berechnet sich anhand einer Exponentialverteilung in Abhängigkeit der Videolänge. Die konkrete Verweilzeit berechnet sich zu: $t_{\text{Verweil}} = \text{Videolänge} + z$, mit z nach $\exp\left(\frac{1}{\text{Videolänge}/2}\right)$

[3] Im Kontext der NextShare-Software spricht man nicht von Chunks, sondern von Blöcken.

Seeding-Zeit

Die Seeding-Zeit gibt an, wie lange ein Peer *aktiv* bereits empfangene Daten an andere Peers hochlädt. Ist die Seeding-Zeit in den nachfolgenden Experimenten nicht explizit angegeben, so stimmt sie mit der Verweilzeit eines Peers überein.

Peer-Selektion

Peer-Selektion beschreibt das Verhalten eines Peers hinsichtlich seines Verbindungsmanagements. Der Standard ist die reguläre Implementierung des NextShare-Clients. Das Verbindungsmanagement des Peers kann lokalitätsfördernde Mechanismen (BNS und BU) einsetzen, um bevorzugt mit anderen Peers aus derselben ISP-Domäne Daten auszutauschen. Die Tests greifen auf Implementierungen von BNS und BU zurück, die im Rahmen des SmoothIT-Projekts entstanden sind. Speziell für BNS gibt es hier die Besonderheit, dass die Peer-Menge, die der Tracker liefert, erst auf Seiten des Peer hinsichtlich Lokalität gefiltert wird.

6.4.4 Supporter-bezogene Parameter

Upstream-Kapazität des Supporter Servers

Da ein Supporter Server mehrere Peers bedienen muss, ist die Upstream-Kapazität höher als bei regulären Peers. Die konkrete Parametrisierung richtet sich grob nach der Bitrate multipliziert mit der maximalen Anzahl zu unterstützender Peers pro Supporter Server. Ein Supporter kann somit bei voller Belegung jeden Peer potentiell mit einer Transferrate bedienen, die der Bitrate des Videos entspricht.

Maximale Anzahl zu unterstützender Peers

Dieser Parameter beschreibt die Anzahl der maximal zu unterstützenden Peers. Die Parametrisierung ist über alle nachfolgenden Evaluationsszenarien konstant auf den Wert 4 festgelegt. Die Untersuchung der Supporter-Strategie [10] hat gezeigt, dass unter diesem Wert die besten Resultate erzielt wurden.

6.4.5 Bandbreitenmanagement-bezogene Parameter

Upload-Multiplikationsfaktor

Dieser Parameter gibt an, mit welchem Faktor die Upstream-Kapazität eines zu befördernden Peers multipliziert werden soll. Dieser Wert sollte innerhalb eines vernünftigen Rahmen bleiben. Für die nachfolgenden Experimente zur Leistungsfähigkeit des adaptiven Bandbreitenmanagements ist dieser Faktor konstant mit dem Wert 2,0 parametrisiert.

Anteil zu befördernder Peers

Dieser Parameter gibt an, wieviele Peers aus der Gesamtmenge aller Peers einer mit Bandbreitenmanagement verwalteten ISP-Domäne mit einer höheren Upload-Bandbreite ausgestattet werden sollen. Ist der Wert jedoch zu klein gewählt, dann sind die Effekte des Bandbreitenmanagements nicht ersichtlich. Ist er hingegen zu groß gewählt, so muss ein ISP verhältnismäßig viele Ressourcen für die Beförderung von Peers aufwenden. Eine Variation dieses Parameters kann im Rahmen einer empirischen Evaluation nur schwer erfolgen. Der Anteil über alle Bandbreitenmanagement-bezogenen Evaluationen ist deshalb konstant auf 20% festgelegt. In Anbetracht einer Promotion der Upstream-Kapazität um den Faktor 2,0 erscheint diese Wahl vernünftig.

6.5 Ausgabemetriken

Durch die Evaluationsszenarien möchte man den Einfluss auf die Systemperformanz der vorgestellten Mechanismen zeigen. Die Systemperformanz wird anhand einer Reihe von Ausgabemetriken gemessen. Die nachfolgenden Paragraphen stellen diese Ausgabemetriken vor und erläutern ggf. ihre Berechnung.

Akkumulierter Datenverkehr nach Klasse

Die folgenden Ausgabewerte beschreiben den akkumulierten Datenverkehr. Sie dienen als Vergleichsbasis dafür, ob durch Einsatz der in dieser Arbeit vorgestellten Mechanismen eine Reduktion hinsichtlich bestimmter Datenverkehrsklassen erreicht werden konnte. Je nach Evaluationsszenario kann man die nachfolgend definierten Klassen für eine ISP-Domäne, oder aber akkumuliert über alle ISP-Domänen vergleichend gegenüberstellen. Im Rahmen der Evaluation erfolgt eine Unterscheidung zwischen:

- Inter-ISP-Upload: Transfervolumen über Inter-ISP-Verbindungen (ausgehend), akkumuliert über alle betrachteten Peers.
- Inter-ISP-Download: Transfervolumen über Inter-ISP-Verbindungen (eingehend), akkumuliert über alle betrachteten Peers.
- Intra-ISP-Gesamt: Transfervolumen über Intra-ISP-Verbindungen, akkumuliert über alle betrachteten Peers.

Startup-Verzögerung

Die Startup-Verzögerung beschreibt die Zeit zwischen dem Starten des korrespondierenden Downloads und dem Starten der Video-Wiedergabe in Sekunden. Die Verzögerung ergibt sich dadurch, dass zunächst der Wiedergabepuffer des Clients vollständig gefüllt sein muss, bevor die Wiedergabe erfolgen kann. Die Startup-Verzögerung ist dementsprechend kleiner, je schneller die notwendigen Datenblöcke von anderen Peers geliefert werden können und somit ein Indiz für die Leistungsfähigkeit des Overlays. Die Startup-Verzögerung wird folgendermaßen ermittelt:

$$t_{\text{Startup}} = t_{\text{Wiedergabe}} - t_{\text{Start}}$$

Akkumulierte Stall-Zeit

Liegt die aktuelle Download-Rate für ein Video unterhalb dessen Bitrate, so leert sich der Wiedergabepuffer schneller, als dass er gefüllt werden kann. Sobald der Wiedergabepuffer leer ist, muss der Client die Wiedergabe des Videos anhalten und zunächst dafür sorgen, dass dieser wieder gefüllt ist. Oszilliert die Download-Rate um den Wert der Bitrate mit starken Abweichungen, so kann es durchaus vorkommen, dass diese Unterbrechungen häufiger auftreten. Der Ausgabewert beschreibt die akkumulierte Zeitspanne über die Dauer der Unterbrechungen. Die akkumulierte Stall-Zeit wirkt sich negativ auf die Erlebnisgüte des Benutzers aus und ist somit charakteristisch für eine schlechte Performanz des Overlays. Sie ist folgendermaßen definiert:

$$t_{\text{Stalling}} = \sum_i t_{\text{Wiederaufnahmezeit}_i} - t_{\text{Anhaltezeit}_i}$$

Gesamtverzögerung

Die Gesamtverzögerung errechnet sich aus der Startup-Verzögerung und der akkumulierten Stalling-Zeit. Sie ist folgendermaßen definiert:

$$t_{\text{Gesamtv.}} = t_{\text{Startup}} + t_{\text{Stalling}}$$

Server-Last

Dieser Ausgabewert beschreibt den gesamten Upload eines Supporter Servers. Durch den Vergleich unterschiedlicher Konfigurationen eines Szenarios kann ermittelt werden, ob sich durch die verwendeten Mechanismen tatsächlich eine Reduktion der Server-Last einstellt oder nicht.

6.6 Verlässlichkeit und Generalisierbarkeit der Ergebnisse

Insbesondere der Mechanismus zum Bandbreitenmanagement ist sehr stark an die Güte der Identifikationsmetriken gekoppelt. Die Identifikationsmetriken bewerten einen Peer anhand der Statistiken, die er an die Datenerhebung sendet. Diese Information kann unverlässlich sein, so dass ein Peer günstiger erscheint, als er sich tatsächlich verhält. Stattet das Bandbreitenmanagement den Peer nun mit einem erhöhten Zugangsprofil aus, so verstärken sich vermutlich negative Effekte, die man durch Anwendung des Mechanismus eigentlich zu reduzieren versuchte (bspw. könnte das Inter-ISP-Transfervolumen ansteigen). Der verlässliche Austausch von Peer-Statistiken ist nicht Gegenstand der vorliegenden Arbeit. Es wird angenommen, dass sich alle Peers innerhalb des Testbeds verlässlich verhalten und korrekte Informationen an die Datenerhebungskomponente senden.

Abschnitt 6.2 erklärt die Zusammenstellung der Topologie über virtuell definierte ISP-Domänen. Die Topologie wird pro Testlauf stets aus Rechnern einer bestimmten Domäne (bspw. nur aus der Domäne der TU Darmstadt) gestellt. Dies hat einen einfachen Grund. Das G-Lab Testbed ist mit der Restriktion behaftet, dass nur mit einer maximalen Transferrate zwischen verschiedenen Domänen Daten fließen dürfen. Dieses Limit ist im Vergleich zu der tatsächlichen Transferrate über Inter-ISP-Verbindungen um bis zu einem Faktor von 10 kleiner. Es ist daher nicht möglich, eine ISP-Domäne bspw. durch Rechner der TU Darmstadt, und eine andere ISP-Domäne durch Rechner der Universität Kaiserslautern zu stellen. Die Topologie verteilt sich im Grunde auf ein LAN (Local Area Network), in welchem Effekte wie Verzögerung und Jitter auf Netzwerkebene deutlich geringer ausfallen, als wenn man auf mehreren, tatsächlichen ISP-Domänen arbeiten würde. Dies wirkt sich natürlich auch auf die Performanz der Peers in den Testläufen aus. Die Ergebnisse sind daher als Tendenz zu verstehen, da sie nicht unbedingt auf eine reale Situation generalisierbar sind.

Gesendete Peer-Statistiken verursachen selbst eine gewisse Last auf Netzwerkverbindungen. Theoretisch kann sich dies auf die Ergebnisse auswirken. Der Einfluss der Peer-Statistiken ist nur schwer messbar. Die nachfolgenden Experimente zeigen allerdings, dass der Overhead durch Peer-Statistiken im Vergleich zum gesamten Transfervolumen gering ist und daher nur einen marginalen Einfluss auf die Ergebnisse haben kann.

7 Evaluation der Supporter-Strategie

Zunächst soll der Frage nachgegangen werden, wie sich die Supporter-Strategie als Komponente zur adaptiven Server-Allokation gegenüber klassischen Server-Strategien verhält. Den Analysen liegt stets der gleiche Datenbestand zugrunde. Die Datenbeschaffung erfolgte durch 3-fache Wiederholung der entsprechenden Testläufe. Die Studie zeigt die Resultate für die nachfolgend beschriebenen Experimente.

- Einfluss der Ankunftsrate auf Ausgabemetriken: Die Analyse prüft den Einfluss der Ankunftsrate auf die Ausgabemetriken Stalling- und Startup-Verzögerung sowie Server- und Peer-Last. Die Ankunftsrate wird in den Testläufen sukzessive erhöht, beginnend bei einer mittleren Ankunft von 12 Peers/Min. bis hin zu einer mittleren Ankunft von 36 Peers/Min. Die Erhöhung erfolgt in 4er-Schritten und ist damit feingranular genug, um einzelne Konfigurationen voneinander abgrenzen zu können.
- Entwicklung der Zustandsallokation über die Dauer einer Testdurchführung: Die Analyse soll zeigen, ob die Assoziation von Peers mit einem Zustand über die Dauer einer Testdurchführung charakteristische Merkmale zeigt, die einen Rückschluss auf das Systemverhalten und die Systemperformanz liefern können. Die Analyse geht dabei auf die einzelnen Server-Strategien separat ein. Die Abhängigkeit der Zustandsallokation zur Ankunftsrate wird ebenfalls exemplarisch aufgezeigt. Die Darstellung der Ergebnisse erfolgt für ein konkretes Szenario. Eine wiederholte Testdurchführung findet nicht statt.
- Basisvergleich der Strategien für ausgewählte Szenarien: Die Analyse zeigt einen Vergleich der Performanz der unterschiedlichen Server-Strategien in ausgewählten Szenarien. Die Auswahl der Szenarien basiert auf den Tests zum Einfluss der Ankunftsrate. Durch diese Tests ist gut ersichtlich, welche Szenarien sich besonders für einen direkten Vergleich eignen. Die Analyse basiert auf demselben Datensatz, der für die Analyse des Einflusses der Ankunftsrate verwendet wurde. Der direkte Vergleich ermöglicht allerdings eine genauere Betrachtung.
- Eine abschließende Betrachtung vergleicht die Ergebnisse der vorliegenden Arbeit (Testbed) mit denen aus [10] (Simulation).

Parameter	**Konfiguration**
Laufzeit des Szenarios	30 Minuten
Videolänge	5 Minuten
Bitrate	540 kb/s
Verfügbare Server	**Je nach Strategie variierend**
Anzahl Peers pro Supporter	4
Server-Kapazität (Upstream)	2048 kb/s
Peer-Zugangsprofile (Upstream)	Low (256 kb/s), Mid (512 kb/s), High (1024 kb/s)
Peer-Zugangsprofil (Downstream)	2048 kb/s
Zugangsprofil-Verteilung	30% Low, 50% Mid, 20% High
Ankunftsrate	**Poisson-Prozess, je nach Szenario variierend**
Peer-Verhalten	Absprung bei 50% der Videolänge im Mittel
Block-Größe	32 kB

Tabelle 7.1: Grundkonfiguration für die Testszenarien der Supporter-Strategie

Die Konfiguration der für die Testdurchführung notwendigen Parameter zeigt Tabelle 7.1. Die Angaben dieser Tabelle sind als Grundkonfiguration zu verstehen. Sofern nötig, zeigen die nachfolgenden Abschnitte Abweichungen zu dieser Konfiguration. Die zugrunde liegende Topologie entspricht der Beschreibung in Abschnitt 6.2. Die Topologie spielt in den nachfolgenden Analysen jedoch eine untergeordnete Rolle.

Parameter	Wert
Verfügbare Server	{1,4,7} statische Server oder 10 Supporter
Ankunftsrate	Poisson-Prozess, 12 bis 36 Peers/Min.

Tabelle 7.2: Abweichungen zur Grundkonfiguration (vgl. Tabelle 7.1) des Supporter-Szenarios

7.1 Einfluss der Ankunftsrate auf Ausgabemetriken

Die Konfiguration der Szenarien entspricht der in Tabelle 7.1, mit den Modifikationen, die in Tabelle 7.2 zusätzlich beschrieben sind. Die entscheidende Variable ist die Ankunftsrate. Sie wird, beginnend von einer Ankunft von 12 Peers/Min., in 4er-Schritten bis hin zu einer Ankunft von 36 Peers/Min. variiert. Die Tests stellen die Strategien der statischen Server und des Supporters vergleichend gegenüber. Es ist zu erwarten, dass die Analyse die folgenden Aussagen bestätigt:

- Stalling-Effekte prägen sich mit zunehmender Ankunftsrate stärker aus.
- Stalling-Effekte können durch Bereitstellung weiterer Server reduziert werden.
- Startup-Verzögerungen prägen sich mit zunehmender Ankunftsrate stärker aus.
- Startup-Verzögerungen können durch Bereitstellung weiterer Server reduziert werden.
- Die Server-Last statischer Server liegt am theoretischen Limit, da davon auszugehen ist, dass sie unentwegt mit anderen Peers in Kontakt stehen und Daten verteilen.
- Die Supporter-Strategie realisiert einen Kompromiss zwischen zunehmender Server-Last und reduzierter Stalling-Verzögerung.

7.1.1 Maximale Anzahl versorgbarer Peers

Ankunftsrate λ	**Peers im Schwarm** $N(t)$	**Anzahl versorgbarer Peers** $M(t)$		
		Für 1 Server	**Für 4 Server**	**Für 7 Server**
12 Peers/Min.	30	**24**	34	43
16 Peers/Min.	40	**31**	41	50
20 Peers/Min.	50	**38**	**47**	57
24 Peers/Min.	60	**44**	**54**	64
28 Peers/Min.	70	**51**	**61**	71
32 Peers/Min.	80	**58**	**68**	**77**
36 Peers/Min.	90	**65**	**75**	**84**

Tabelle 7.3: Anzahl der Peers, die von div. statischen Servern und den Peers im Schwarm potentiell versorgt werden können. Die Annahme ist, dass die Upstream-Kapazität eines Peers zu 80% ausgelastet ist [60]. Der Prefetch-Faktor f des Give-to-Get-Protokolls ist mit $1,2$ bemessen.

Ein einfaches analytisches Modell zur Berechnung der maximalen Anzahl von Peers, die mit einer gewissen Mindestdatenrate durch die Upstream-Kapazitäten von Servern und Peers versorgt werden können, stützt die Argumentation in diesem Abschnitt. Das Modell berücksichtigt die Ankunfts- und Absprungrate sowie die individuellen Zugangsprofile der Peers und Server.

Sei S die Menge aller Server im System und u_s die individuelle Upstream-Kapazität eines Servers s. Dann kann man die gesamte Upstream-Kapazität aller Server unter der Annahme, dass die Auslastung idealerweise bei 100% liegt, folgendermaßen ausdrücken:

$$U_S = \sum_{s \in S} u_s$$

Sei P die Menge aller Zugangsprofile und $p(m)$ die Wahrscheinlichkeit eines Zugangsprofils m, sowie u_m die Upstream-Kapazität des korrespondierenden Zugangsprofils. Die Upstream-Kapazität aller Peers im System kann man unter Berücksichtigung eines Auslastungsfaktors ρ[1] folgendermaßen darstellen:

$$U_P(t) = \rho \cdot \sum_{m \in M} p(m) \cdot N(t) \cdot u_m$$

Sei b die Bitrate des gewünschten Videos und f der Prefetching-Faktor, den das System nutzt, um Datenblöcke mit niedriger Priorität zu beziehen. Die Mindestdatenrate d, die ein Peer einhalten muss, um ein Video unterbrechungsfrei wiedergeben zu können, ist:

$$d = b \cdot f$$

Die Anzahl der Peers, die das System zu einem gegebenen Zeitpunkt mit der Mindestdatenrate versorgen kann, ist dann:

$$M(t) = \frac{U_S + U_P(t)}{d}$$

Tabelle 7.3 zeigt die Auswertung von $M(t)$ für die Strategien mit statischen Servern unter Berücksichtigung einer wachsenden Ankunftsrate. Zu erkennen ist, dass mit wachsender Ankunftsrate die Funktion $M(t)$ (bei allen Strategien) langsamer wächst als $N(t)$ und je nach Strategie ab einer bestimmten Ankunftsrate unterschreitet. Der Bedarf der Peers kann in diesem Fall nicht mehr durch das System bewältigt werden. Die entsprechenden Fälle sind in Tabelle 7.3 durch Fettschrift hervorgehoben.

7.1.2 Einfluss auf die Stalling-Verzögerung

Die Untersuchung der Metriken in Abhängigkeit der Ankunftsrate beginnt zunächst mit der Analyse der Stalling-Verzögerung. Die Abbildungen zeigen die 95%-Perzentil der Stalling-Verzögerung. Die Darstellung der 50%-Perzentile entfällt, da diese über alle Strategien und Ankunftsraten hinweg bei 0.0 Sekunden liegen.

Die Szenarien, in denen ein statischer Server eingesetzt wurde, um Chunks zu verteilen, verhalten sich – wie erwartet – relativ ungünstig. Die Stalling-Verzögerung ist generell sehr hoch, allerdings muss man dazu sagen, dass sie trotz wachsender Ankunftsraten relativ stabil bleibt. Bereits der erste Wert (vgl. Abbildung 7.1, 12 Peers/Min.) zeigt eine sehr hohe Standardabweichung, so dass der nachfolgende Abstieg für die Ankunftsraten 16 Peers/Min und 20 Peers/Min. nicht zwangsweise einen systembedingten Grund haben muss. Im Bereich von 24 Peers/Min. bis 36 Peers/Min. steigt die Stalling-Verzögerung wieder leicht an, allerdings verbietet auch hier die teilweise sehr hohe Standardabweichung in den Resultaten eine genauere Analyse. Der Grund dafür liegt zum Einen in der schlechten Performanz des Servers, zum Anderen aber auch an der Instabilität des 95%-Perzentils, wenn sich nur wenige Peers über die Dauer des Experiments im System befanden. Dies wirkt sich vor allem bei Szenarien mit niedrigen Ankunftsraten aus. Hier kann man beobachten, dass bei 12 Peers/Min. · 30 Min. nur die Ergebnisse von insgesamt 360 Peers in die Auswertung einfließen und der Bereich um die 95%-Perzentil von einer sehr hohen Dynamik geprägt ist. Die unzureichende Performanz eines einzigen statischen Servers stellt sich in allen Konfigurationen dar. Betrachtet man die Anzahl der Peers, die von Server und den Peers im Schwarm potentiell versorgt werden können, so ist dies nicht weiter verwunderlich. Tabelle 7.3 zeigt, dass der Bedarf Peers selbst bei einer niedrigen Ankunftsrate von 12 Peers/Min. nicht gedeckt ist. Dies führt unweigerlich zu Stalling-Effekten, die sich über den Evaluationszeitraum von 30 Minuten hin erstrecken.

[1] Eine Studie des Coolstreaming Video-on-Demand Systems hat gezeigt, dass die Auslastung des Peer-Upstreams in diesem System um 80% liegt [60]. Das analytische Modell greift auf diesen Wert zurück, da die Auslastung des Upstreams in P2P Video-on-Demand Systemen bedingt durch die Block-Diversität im Vergleich zu reinen P2P-Downloads etwas geringer ausfällt.

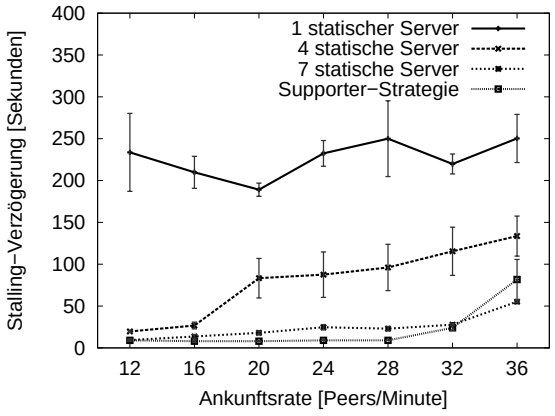


Abbildung 7.1: Einfluss der Ankunftsrate auf die Stalling-Verzögerung, betrachtet über alle Server-Strategien

Vier statische Server können den Bedarf der Peers bis zu einer Ankunftsrate von 16 Peers/Min. relativ gut abdecken. Die Stalling-Verzögerung ist in diesem Bereich zwar höher als in einer Konfiguration mit sieben statischen Servern oder der Supporter-Strategie, jedoch ist diese erhöhte Verzögerung gemessen an der Videolänge von 5 Minuten nicht besonders signifikant. Die Performanz des Overlays verschlechtert sich jedoch deutlich ab einer Ankunftsrate von 20 Peers/Min. und wächst mit zunehmenden Ankunftsraten weiter an. Dies kann durch die limitierte Gesamt-Upload-Kapazität der Server und Peers im Schwarm erklärt werden. Tabelle 7.3 liefert hier ebenfalls einen Richtwert für den Einbruch der Overlay-Performanz. Dieser liegt im analytischen Modell bei 20 Peers/Min. und deckt sich mit den Ergebnissen in Abbildung 7.1.

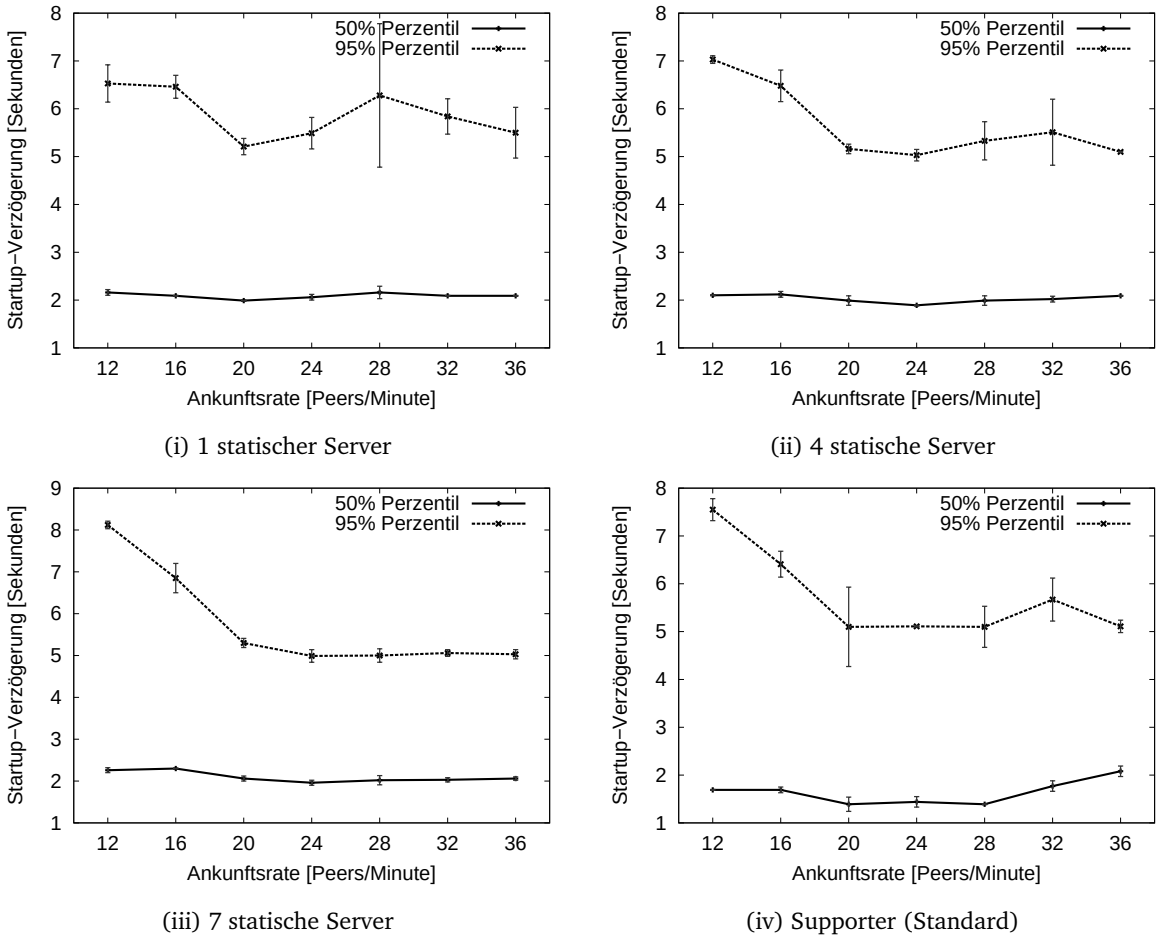
Die Tendenz beim Einsatz von sieben statischen Servern ist zwar noch vorhanden, allerdings ist die Reduktion der Stalling-Verzögerung bei weitem nicht so signifikant, gemessen an dem Schritt von einem Server zu vier Servern. Die Leistungsfähigkeit sieben Server liegt nur etwas hinter der Performanz der Supporter-Strategie zurück. Erst ab einer Ankunftsrate von 36 Peers/Min. zeigt die Variante mit sieben statischen Servern einen leichten Vorsprung gegenüber dem Supporter. Im direkten Vergleich mit vier statischen Servern schneidet die Strategie mit sieben statischen Servern erst ab einer Ankunftsrate von 20 Peers/Min. signifikant besser ab. Diese Aussage wird wieder durch das analytische Modell (vgl. Tabelle 7.3) gestützt.

Am Verlauf der Linie, die mit der Supporter-Strategie korrespondiert (vgl. Abbildung 7.1), ist deutlich zu erkennen, dass der Supporter eine konstant gute Dienstgüte realisiert. Die Stalling-Verzögerung erhöht sich erst ab einer Ankunftsrate von 32 Peers/Min. Dennoch ist hier zu sagen, dass die Supporter-Strategie bezüglich ihrer Performanz alle anderen Strategien schlägt. Die etwas schlechtere Performanz im Szenario mit einer Ankunftsrate von 36 Peers/Min. lässt sich vermutlich durch Messungenauigkeiten erklären, da in diesem Fall eine relativ hohe Standardabweichung vorliegt. Eine Strategie mit sieben statischen Servern erscheint an dieser Stelle bezüglich der Stalling-Verzögerung günstiger, jedoch müssen an dieser Stelle noch andere Ausgabemetriken berücksichtigt werden, um die verschiedenen Strategien in ihrer Gesamtheit beurteilen zu können.

7.1.3 Einfluss auf die Startup-Verzögerung

Abbildung 7.2 zeigt den Median und das 95%-Perzentil bezüglich der Startup-Verzögerung für alle untersuchten Strategien. Auffällig ist zunächst, dass sich alle Strategien ähnlich verhalten. Abbildung 7.2(i) zeigt die Ergebnisse für die Konfiguration mit einem statischen Server. Der Median ist hier sehr stabil und bewegt sich konstant um einen mittleren Wert von ca. 2 Sekunden. Das 95%-Perzentil fängt zunächst hoch an (bei ca. 6,5 Sekunden), erfährt dann ab einer Ankunftsrate von 20 Peers/Min. eine Reduktion bezüglich der Startup-Verzögerung auf ca. 5 Sekunden. Dieser Wert wird über die sukzessive ansteigenden Ankunftsraten grob gehalten. Die hohen Standardabweichungen bei dem Szenario mit einer Ankunftsrate von 28 Peers/Min. deuten auf einen statistischen Ausreißer hin.

Die Abbildungen 7.2(ii)-(iv) zeigen dieselbe Tendenzen auf dem Median und dem 95%-Perzentil. Die Startup-Verzögerung ist zunächst leicht erhöht, sinkt dann aber ab einer Ankunftsrate von 20 Peers/Min. signifikant ab und pendelt sich auf einen nahezu konstanten Wert um den Bereich 5 Sekunden ein.



(i) 1 statischer Server

(ii) 4 statische Server

(iii) 7 statische Server

(iv) Supporter (Standard)

Abbildung 7.2: Einfluss der Ankunftsrate auf die Startup-Verzögerung, betrachtet über alle Server-Strategien

Dieses Verhalten lässt sich wie folgt erklären. Jeder Server geht eine Anzahl von k Verbindungen mit Peers ein. Die maximale Upstream-Kapazität teilt sich nach TCP Fair Share ungefähr gleichmäßig auf die einzelnen k Verbindungen auf. Diese Annahme kann getroffen werden, da alle Peers im Schwarm dieselbe Downstream-Kapazität aufweisen. Betrachtet man nun das Zeitintervall $[t_0; t_1]$ mit Hinblick auf unterschiedliche Ankunftsraten $\lambda_1 = 12$ Peers./Min und $\lambda_2 = 24$ Peers/Min., dann lässt sich zweifelsohne sagen, dass zum Zeitpunkt t_1 bei einer mittleren Ankunft von λ_2 sich deutlich mehr Peers im System befinden, als bei einer mittleren Ankunft von λ_1. Dies wirkt sich auf die Anzahl der Verbindungen k aus. Bei einer mittleren Ankunft von λ_2 ist die Verbindungszahl k höher, dafür sind aber die Transferraten über individuelle Verbindungen niedriger. Dies hat den Effekt, dass die Server langsamer Datenblöcke an Peers verteilen können. Die Fortschrittsgeschwindigkeit in der Video-Wiedergabe nimmt ab, dafür kann allerdings der Bedarf an den ersten Datenblöcken neu hinzukommender Peers vom Schwarm größtenteils aufgefangen werden. Die Wiedergabe des Videos startet früher, allerdings ist Stalling bezüglich der Datenblöcke, die weit entfernt von der Abspielposition liegen, ein Problem. Man beachte, dass sich diese Beobachtungen mit der vorhergenden Betrachtung der Stalling-Verzögerung decken. Dieser Effekt wird jedoch mit weiter zunehmender Ankunftsrate mitigiert, da hierdurch nur eine weitere Verschiebung von Server- auf Peer-Last realisiert wird. Das heißt nichts anderes, als dass ab einer gewissen Ankunftsrate die Server bezüglich der Verteilung der ersten Datenblöcke entlastet werden und somit mehr Kapazität haben, um die Peers mit Daten zu versorgen, die gerade in Ermangelung neuer Datenblöcke das Video pausieren müssen. Das hat zur Folge, dass die globale Fortschrittsgeschwindigkeit in der Wiedergabe des Videos mit zunehmender Ankunftsrate wieder ansteigt.

Abbildung 7.3 zeigt eine Überlagerung der 95%-Perzentile aller untersuchter Strategien. Deutlich ist zu erkennen, dass alle Strategien sich annähernd gleich verhalten. Abweichungen gibt es bezüglich der Szenarien mit einer Ankunftsrate von 12 Peers/Min. bzw. 16 Peers/Min. Interessanterweise zeigt in diesem Bereich die Strategie mit einem statischen Server die besten Resultate. Alle anderen Strategien zeigen geringfügig höhere Werte. Ein Trend ist hier

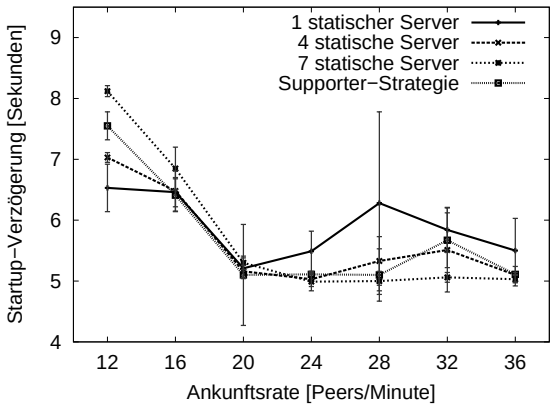


Abbildung 7.3: Einfluss der Ankunftsrate auf die Startup-Verzögerung. Die Darstellung zeigt die 95%-Perzentile der Startup-Verzögerung.

nicht zu erkennen. Da sich die Abweichungen in einem sehr kleinen Rahmen von ein bis zwei Sekunden bewegen, können durchaus einzelne statistische Ausreißer für eine leichte Verschiebung entlang der y-Achse sorgen. Dies geht zumindest mit der Beobachtung einher, dass die Standardabweichungen im Vergleich zu der 95%-Perzentile relativ groß sind. Systembedingt lässt sich dieses Verhalten nicht erklären. Im Gegenteil: Es steht konträr zu der Annahme, dass bei steigender Gesamt-Server-Kapazität die Startup-Zeit reduziert werden müsste.

7.1.4 Einfluss auf die Server- und Peer-Last

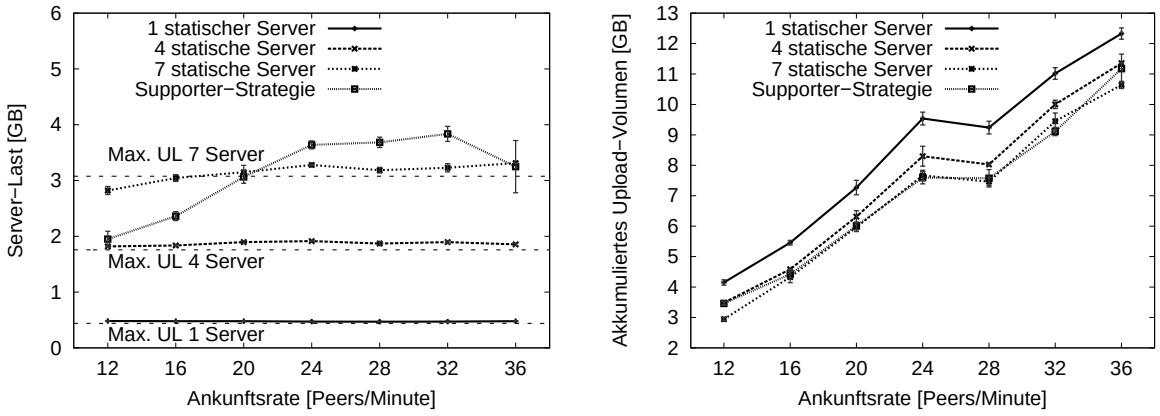
Abbildung 7.4(i) zeigt die Server-Last in Abhängigkeit der Ankunftsrate. Zunächst ist zu beobachten, dass die maximal theoretische Upload-Kapazität der einzelnen statischen Server-Strategien gut umgesetzt wird. Die Umsetzung erfolgt – sowohl für Server als auch für Peers – über einen NextShare-internen Mechanismus zur Ratenkontrolle. Die Ratenkontrolle kann nicht 100%ig exakt erfolgen, sondern schwingt um den Zielwert, so dass es zu einer gewissen Abweichung kommt. Während der tatsächliche maximale Upload bei einem statischen Server nur knapp über der theoretischen Grenze liegt, die Abweichung also gering ist, erhöht sich dieser Fehler bei der Überlagerung der Upload-Volumen mehrerer Server. Dies ist nicht weiter verwunderlich, führt allerdings auch nicht zu einer signifikanten Verzerrung der Ergebnisse.

Interessant ist der Bereich der Ankunftsraten 12 Peers/Min. bis 20 Peers/Min. Bei 12 Peers/Min. ist deutlich zu erkennen, dass die Supporter-Strategie nur geringfügig mehr Server-Last erzeugt, als die Strategie mit vier statischen Servern. Die Server-Last nimmt für die Supporter-Strategie mit wachsender Ankunftsrate stetig zu. Bringt man diese Resultate in Bezug zu dem Einfluss der Ankunftsrate auf die Stalling-Verzögerung, so stellt man fest, dass die Supporter-Strategie tatsächlich nur die erforderliche Server-Last generiert, so dass Stalling-Effekte weitestgehend vermieden werden.

Ab einer Ankunftsrate von 24 Peers/Min. wächst die Server-Last bezüglich der Supporter-Strategie über die Server-Last anderer Verfahren hinaus. Die Reduktion der Server-Last im Szenario mit einer Ankunftsrate von 36 Peers/Min. kann als statistischer Ausreißer betrachtet werden, zumal die Standardabweichung signifikant höher ist, als in den anderen Szenarien.

Das akkumulierte Upload-Volumen der Peers (siehe Abbildung 7.4(ii)) zeigt einen charakteristischen Verlauf. Gemäß der Erwartung steigt das Upload-Volumen annähernd linear mit einer wachsenden Ankunftsrate. Erwartungsgemäß ist das Upload-Volumen der Peers am höchsten, wenn nur ein statischer Server eingesetzt wird. Bei mehreren statischen Servern wird die Last entsprechend von den Peers genommen und auf die Server umverteilt (vgl. Abbildung 7.4(i)).

Interessant ist, dass die Supporter-Strategie bis auf das Szenario mit einer mittleren Ankunft von 32 Peers/Min. durch die Strategien mit vier bzw. sieben Server flankiert wird. Dies zeigt, dass die Supporter-Strategie den Tradeoff zwischen der Reduktion von Server-Last und der Einhaltung einer hinreichenden Dienstgüte sucht. Insbesondere



(i) Server-Last in Abhängigkeit der Ankunftsrate

(ii) Peer-Last in Abhängigkeit der Ankunftsrate

Abbildung 7.4: Einfluss der Ankunftsrate auf die Server- und Peer-Last, jeweils betrachtet über alle Strategien. Die horizontalen Linien in Abbildung (i) zeigen das theoretisch maximale Upload-Volumen über die Evaluationsdauer einer bestimmten Strategie. Durch die NextShare-interne Ratenkontrolle zeigt sich ein gewisser Fehler im tatsächlichen Upload-Volumen.

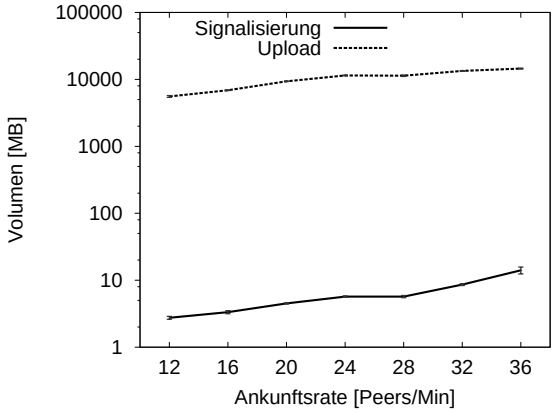


Abbildung 7.5: Einfluss der Ankunftsrate auf das Volumen Supporter-bezogener Kommunikation. Die Grafik zeigt im Vergleich dazu den gesamten Upload (Server und Peers).

bei einer Ankunftsrate von 12 Peers/Min. und 24 Peers/Min. fällt das Upload-Volumen der Peers bei Verwendung der Supporter-Strategie mit vier respektive sieben statischen Servern zusammen. Dies zeigt zusätzlich, dass sich diese beiden Szenarien für einen direkten Vergleich der Strategien untereinander eignen.

7.1.5 Einfluss auf den Signalisierungs-Overhead

Abbildung 7.5 zeigt die Auswirkungen einer steigenden Ankunftsrate auf den Signalisierungs-Overhead und den gesamten Server- und Peer-seitigen Upload. Der Signalisierungs-Overhead zeigt einen nahezu linearen Anstieg. Der Anstieg des Server- und Peer-seitigen Upload-Volumens nimmt ab einer Ankunftsrate von 28 Peers/Min. deutlich zu, so dass beide Kurven bei einer Ankunftsrate von 36 Peers/Min. sich bereits angenähert haben. Der Unterschied in den Einheiten beträgt den Faktor 1024, so dass in diesem Fall die Kontrollnachrichten der Supporter-Strategie $\approx 1/1000$ des gesamten Uploads ausmachen. Folgt man der Tendenz, so kann man sagen, dass der Signalisierungs-Overhead in Relation zu dem gesamten Upload-Volumen zunehmend vernachlässigbar ist.

7.1.6 Diskussion

Bei der Betrachtung der Stalling-Verzögerung hat man deutlich gesehen, dass der Vorzug einer Strategie mit mehreren statischen Servern gemessen an einer Strategie, die weniger statische Server einsetzt, ab einem gewissen Punkt nicht mehr gerechtfertigt ist. So ist bspw. die Reduktion der Stalling-Verzögerung beim Schritt von einem statischen Server auf vier statische Server enorm. Beim Schritt von vier auf sieben statische Server ist diese Einsparung nicht mehr ganz so stark ausgeprägt bzw. erscheint nur in einem Szenario mit deutlich höherem Peer-Bedarf signifikant. An dieser Stelle spielt die Supporter-Strategie ihre Stärken aus. Wachsender Bedarf kann durch die Zuschaltung weiterer Supporter kompensiert werden. Die Performanz der Supporter-Strategie bleibt fast durchweg stabil und dies bei einer vergleichbar guten Server-Last.

Die erwarteten Ergebnisse stimmen dementsprechend mit den tatsächlichen Ergebnissen bis auf das abweichende Systemverhalten bei der Auswirkung der Ankunftsrate auf die Startup-Verzögerung überein. Das Systemverhalten bezüglich der Startup-Verzögerung konnte im Zuge der Evaluation erklärt werden.

7.2 Entwicklung von Zustandsallokationen an der Überwachungskomponente

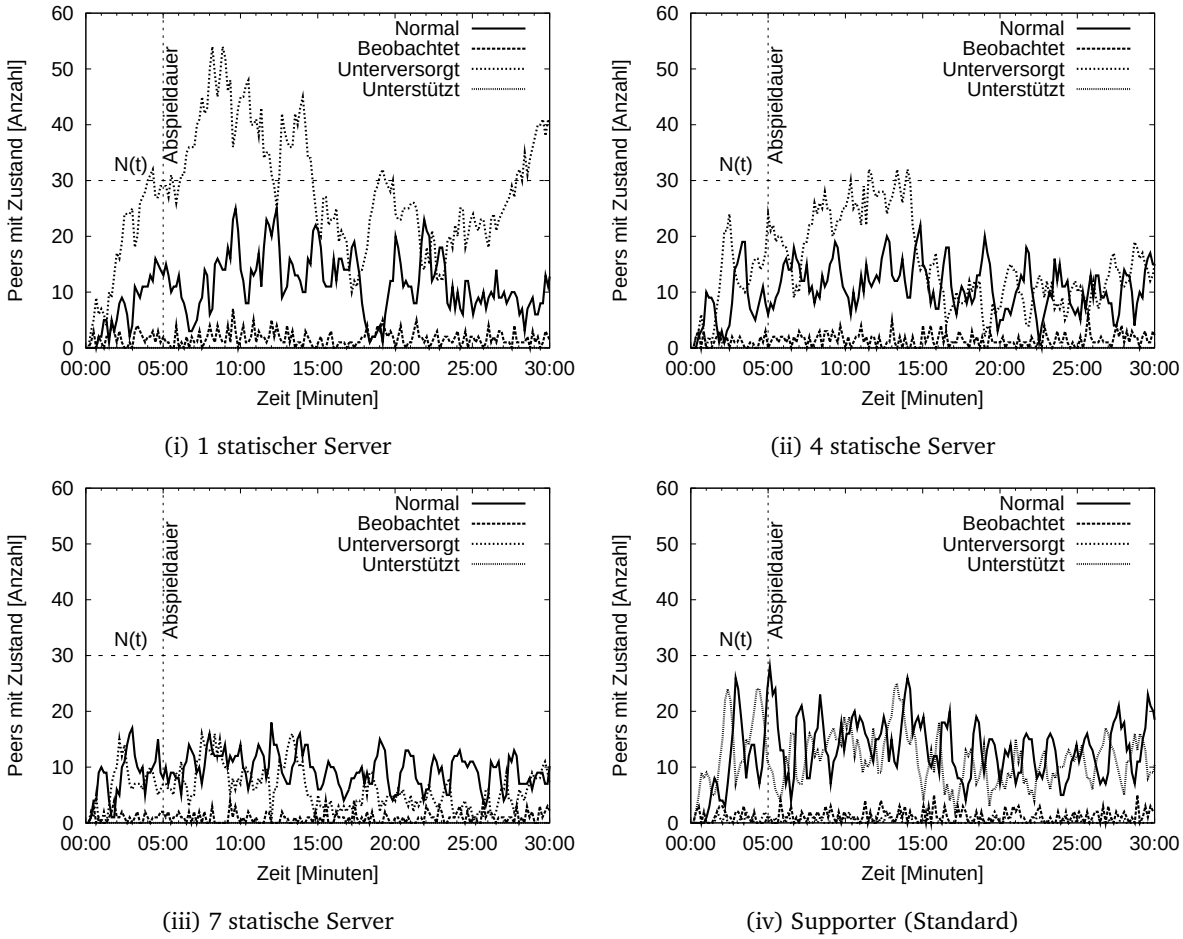
Die Analyse zeigt den qualitativen Verlauf der Zustandsallokationen an der Überwachungskomponente der Supporter-Strategie. Zur direkten Vergleichbarkeit führen Peers das Supporter-Protokoll mit der Überwachungskomponente durch, auch wenn sich keine Supporter im Schwarm befinden. Dies erlaubt einen Vergleich zwischen dem *Unterversorgt*-Zustand und dem *Unterstützt*-Zustand zwischen den Strategien der statischen Server und der Supporter-Strategie. Anzumerken ist, dass sich nicht alle Peers, die sich im Schwarm befinden, an der Überwachungskomponente registriert haben müssen. Dies wird insbesondere bei der Betrachtung von Szenarien deutlich, die über mehrere Server verfügen, so dass unterversorgte Peers weniger häufig auftreten. Durch die Analyse sind direkte Rückschlüsse auf die Gesamtperformanz des Systems möglich.

Parameter	**Wert**
Verfügbare Server	{1,4,7} statische Server oder 10 Supporter
Ankunftsrate	Poisson-Prozess, 12 und 24 Peers/Min..

Tabelle 7.4: Abweichungen zur Grundkonfiguration (vgl. Tabelle 7.1) des Supporter-Szenarios

Die Konfiguration der Szenarien entspricht der in Tabelle 7.1, mit den Modifikationen, die in Tabelle 7.4 zusätzlich beschrieben sind. Die Untersuchung der Zustandsallokation erfolgt mit 12 Peers/Min. und 24 Peers/Min., so dass sich der etwaige Einfluss der Ankunftsrate in den Resultaten zeigen sollte. Es ist zu erwarten, dass die Analyse die folgenden Aussagen bestätigt:

- Die Zustandsallokation in einem Szenario mit Supporter-Strategie sollte ein funktionierendes Protokoll zeigen.
- Unter Verwendung der Supporter-Strategie erhalten zu keinem Zeitpunkt mehr als maximal 40 Peers Unterstützung. Benötigen dennoch mehr Peers Unterstützung, so verweilen diese im *Unterversorgt*-Zustand, was an einem temporären Anstieg der Unterversorgt-Kurve zu erkennen wäre.
- Unter Verwendung der Supporter-Strategie sollte in Szenarien mit niedriger Ankunftsrate die Beobachtet- und Unterversorgt-Kurven auf einem konstant niedrigen Niveau bleiben.
- Die Testdurchführung mit einem bzw. vier statischen Servern sollte zeigen, dass diese Server zu leistungsschwach sind, um eine gute Performanz des Schwarms zu gewährleisten. Dies sollte sich insbesondere durch eine stetig ansteigende Unterversorgt-Kurve darstellen. Beim Einsatz mehrerer Server und gleicher Ankunftsrate sollte die Unterversorgt-Kurve einen flacheren Anstieg zeigen.
- Nach der Länge der Wiedergabedauer des Video hat das System zum ersten Mal eine konstante Arbeitslast erreicht. Ab dieser Marke sollte sich der Gesamtzustand des Systems geringfügig verbessern.



(i) 1 statischer Server

(ii) 4 statische Server

(iii) 7 statische Server

(iv) Supporter (Standard)

Abbildung 7.6: Vergleich der Entwicklung von Zustandsallokationen über alle Strategien bei einer mittleren Ankunft von 12 Peers/Min.. Die vertikale Linie zum Zeitpunkt 5 Minuten zeigt die Abspieldauer des Videos. Die Videolänge korreliert mit dem Zeitpunkt, zu dem dass System seine konstante Arbeitslast erreicht hat.

7.2.1 Vergleich der Strategien bei einer Ankunftsrate von 12 Peers/Min.

Abbildung 7.6 stellt die Entwicklung der Zustandsallokation für einen, vier und sieben statische(n) Server(n) der Supporter-Strategie vergleichend gegenüber. Vor der Diskussion dieses Vergleichs ist zunächst eine erfreuliche Erkenntnis festzuhalten. Abbildung 7.6(iv) zeigt, dass das Supporter-Protokoll gut funktioniert. Die Kurven für beobachtete und unterversorgte Peers sind relativ niedrig. Dies bedeutet, dass die entsprechenden Zustandswechsel gemäß der Protokollspezifikation erfolgen. Sofern es freie Supporter-Plätze gibt, erfolgt die unmittelbare Zuweisung an Peers, was durch die leichte Oszillation der Unterversorgt-Linie erkennbar ist. Interessant ist auch die Tatsache, dass die Unterstützt-Kurve ein stetiges Auf und Ab zeigt. Das spricht dafür, dass einzelne Peers, die unmittelbar vor der Unterbrechung der Video-Wiedergabe stehen, Unterstützung erhalten und diese wieder abgeben, sofern sie nicht mehr benötigt wird. Nach Protokolldefinition fallen unterstützte Peers zurück in den *Normal*-Zustand. Die Normal-Kurve wird aus diesem Grund maßgeblich von den Peers gebildet, die Unterstützung erhalten haben und diese nun nicht mehr benötigen. Daher ist die Normal-Kurve in Bezug zur Unterstützt-Kurve entlang der x-Achse leicht verschoben. Die korrekten Zustandswechsel lassen sich überdies auch in den Abbildungen 7.6(i)-(iii) erkennen. Hier ist die Beobachtet-Kurve ebenfalls sehr gering, so dass Peers nach der Zulassungskontrolle der Supporter-Strategie offensichtlich korrekt in den Unterversorgt-Zustand übergehen.

Bevor das System eine konstante Arbeitslast erreicht hat (ca. um 5 Minuten), steigt der Bedarf der Peers stetig an. Dies zeigt vor allem Abbildung 7.6(i) deutlich, da hier nur ein statischer Server Datenblöcke initial verteilt. Die Anzahl unterversorgter Peers steigt daher rapide an und da sich aufgrund der limitierten Kapazität dieser

Strategie Stalling-Effekte ausbilden, erreicht das System einen Höchstwert an unterversorgten Peers erst an der 9-Minuten-Marke. Dass hier Stalling-Effekte die Ursache sind, ist offensichtlich, denn der Erwartungswert für die Anzahl Peers im Schwarm liegt bei diesem Szenario bei ca. 30 Peers. Durch Stalling-Effekte wird der Absprung der Peers hinausgezögert und es kommt zu dem in Abbildung 7.6(i) gezeigten Systemverhalten. Durch die längere Systemzeit der Peers, die Stalling-Effekte erfahren haben, können diese Peers mehr von den Datenblöcken, die sie bereits besitzen, an andere Peers weitergeben. Der Systemzustand verbessert sich daher zunächst nach der 9-Minuten-Marke, pendelt sich aber ca. ab der 19. Minute auf die typische Arbeitslast bei dieser Ankunftsrate ein.

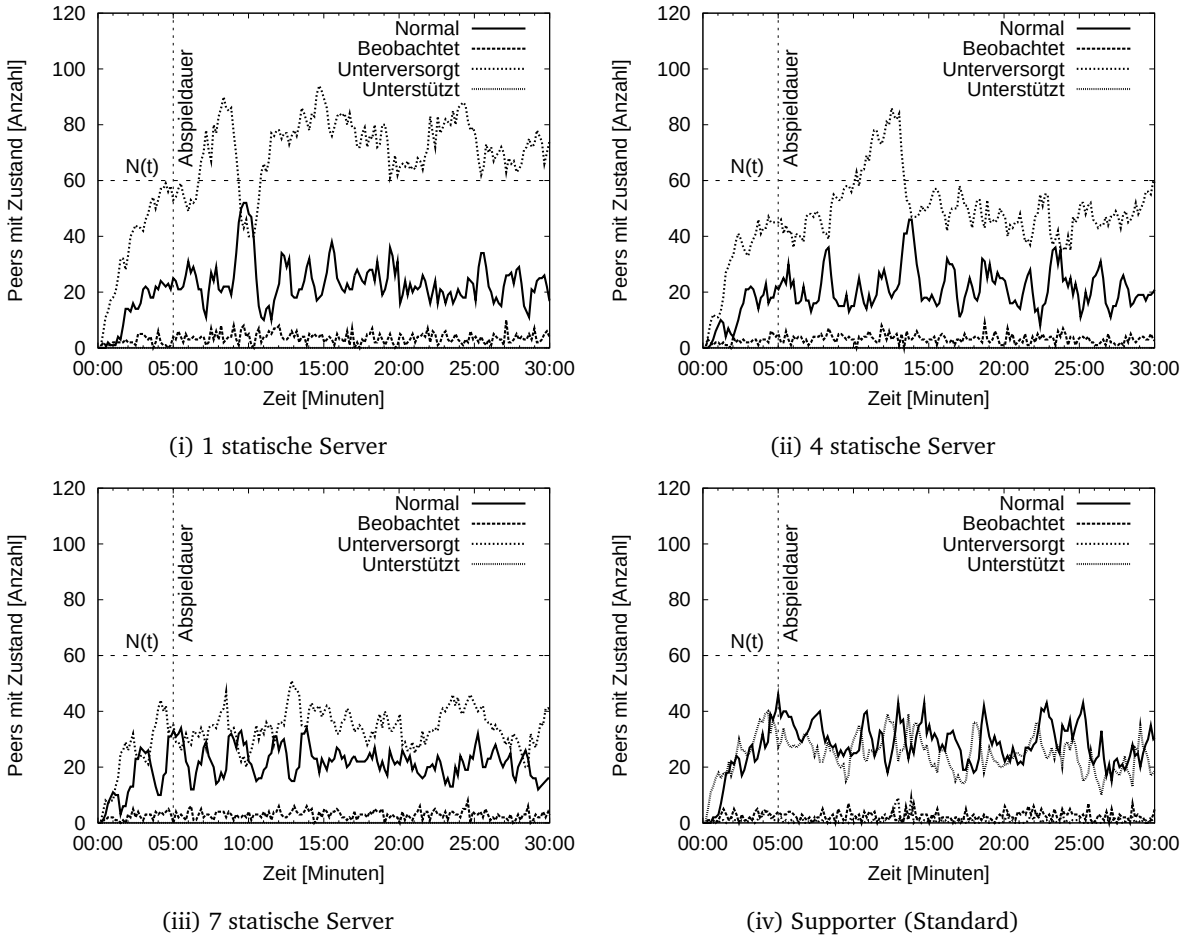
Diese Effekte erkennt man auch in Abbildung 7.6(ii). Die Grafik zeigt das selbe Szenario, mit dem Unterschied, dass nun vier statische Server seit Beginn des Experiments Daten verteilen können. Es ist deutlich zu erkennen, dass die Mehrzahl an Servern für eine Entlastung des Systems sorgen. Der Höchstwert stellt sich allerdings auch hier erst hinter der 5-Minuten-Marke ein, überschreitet jedoch den Erwartungswert von 30 Peers im Mittel bei der vorherrschenden Ankunftsrate nur wenige Male (siehe die Zeitpunkte um 12 und 15 Minuten). Die Lastsituation prägt sich auch zeitlich deutlich geringer aus, als im Fall mit nur einem statischen Server (vgl. das Intervall [8:00;14:30] in Abbildung 7.6(ii) mit dem Intervall [4:00;30:00] in Abbildung 7.6(i)).

Sieben statische Server sorgen erwartungsgemäß für eine weitere Verbesserung des Systemzustands. Abbildung 7.6(iii) zeigt, dass die Server den Peer-Bedarf sehr gut stellen können, so dass sich Defizite durch unterversorgte Peers nicht stark ausbilden. Überhaupt sorgen sieben statische Server dafür, dass ein Großteil der Peers hinreichend mit Daten versorgt wird. Dies ist daran erkennbar, dass die Unterversorgt-Linie über die gesamte Dauer der Testdurchführung deutlich unterhalb des Erwartungswertes von 30 Peers im Mittel liegt.

Abbildung 7.6(iv) zeigt die Entwicklung der Zustandsallokation unter Verwendung der Supporter-Strategie. Im Fokus steht hier nun die Unterstützt-Linie. Sie oszilliert über die gesamte Dauer der Testdurchführung, überschreitet allerdings niemals den Erwartungswert von 30 Peers im Mittel. Dies spricht dafür, dass Stalling-Effekte im Vergleich zu einem bzw. vier statischen Servern erheblich reduziert werden. Die Unterversorgt-Linie hält ein konstant niedriges Niveau, was darauf schließen lässt, dass Peers, denen Stalling-Effekte bevorstehen, Unterstützung erhalten. Das ist erklärbar, da die Anzahl der potentiell verfügbaren Supporter-Plätze 40 beträgt. Dieser Wert wird nie erreicht, so dass unterversorgte Peers sofort einem Supporter zugewiesen werden können. Die häufigen An- und Abstiege der Unterstützt-Kurve signalisieren, dass bei einem hohen Peer-Bedarf mehr als vier Supporter aktiv gewesen sind. Zu Zeiten, in denen der Peer-Bedarf durch das Overlay gut aufgefangen werden kann, sind deutlich weniger Supporter aktiv.

7.2.2 Vergleich der Strategien bei einer Ankunftsrate von 24 Peers/Min.

Erhöht man die Ankunftsrate auf 24 Peers/Min. im Mittel, so verstärken sich die zuvor beschriebenen Effekte. Abbildung 7.7 zeigt den Vergleich der Strategien untereinander. Die erwartete Anzahl von Peers im Schwarm liegt in diesem Szenario bei ca. 60 im Mittel. Grafik 7.7(i) zeigt, dass ein statischer Server dem Peer-Bedarf über die gesamte Dauer der Testdurchführung nicht gerecht werden kann, da die Anzahl der unterversorgten Peers nahezu konstant massiv über der 60-Peer-Schwelle liegt. Dies deutet auf eine Vielzahl von Stalling-Effekten hin, was die in Abschnitt 7.1 gezeigte Auswertung bestätigt. Abbildung 7.7(ii) zeigt, dass vier statische Server den Peer-Bedarf zwar besser auffangen können, es aber dennoch zu Stalling-Effekten kommt. Es bildet sich ein ähnliches Systemverhalten heraus, wie es Abbildung 7.7(i) im Falle einer Ankunftsrate von 12 Peers/Min. und einem statischen Server zeigt. Der Peer-Bedarf kann allerdings nach der ersten Hürde (Abstieg der Unterversorgt-Linie um 12 Minuten) besser durch das System gestemmt werden. Sieben statische Server fangen den Peer-Bedarf deutlich besser auf. Stellt man diese Grafik jedoch in Bezug zu der in Abschnitt 7.1 durchgeführten Analyse, dann erkennt man, dass dennoch nicht unerhebliche Stalling-Effekte auftreten. Abbildung 7.7(iv) kann man entnehmen, dass die Supporter-Strategie auch in diesem Szenario eine durchweg gute Leistung zeigt. Da sich im Mittel 60 Peers nach der 5-Minuten-Grenze im System befinden, die Unterstützt-Linie zusammen mit der Unterversorgt-Linie jedoch nie diesen Wert überschreitet, befürchten einige Peers gar keine Stalling-Effekte. Die Anzahl der Supporter-Plätze beträgt hier 40. Die Grafik zeigt auch, dass zu gegebenen Zeitpunkten (bspw. um 4, 14 und 22 Minuten) alle Supporter aktiv sind und keine freien Plätze mehr besitzen. Zu diesen Zeitpunkten zeigt sich auch eine temporäre Erhöhung der Unterversorgt-Linie. Demnach würden gerne weitere Peers Unterstützung erhalten, da sie unmittelbar Stalling-Effekte befürchten. Dieser Systemzustand prägt sich immer nur kurzfristig aus, so dass man davon ausgehen kann, dass die Supporter-Strategie durch die inhärente Zulassungskontrolle eine gute Verteilung der Un-



(i) 1 statische Server

(ii) 4 statische Server

(iii) 7 statische Server

(iv) Supporter (Standard)

Abbildung 7.7: Vergleich der Entwicklung von Zustandsallokationen über alle Strategien bei einer mittleren Ankunft von 24 Peers/Min.. Die vertikale Linie zum Zeitpunkt 5 Minuten zeigt die Abspieldauer des Videos. Die Videolänge korreliert mit dem Zeitpunkt, zu dem dass System seine konstante Arbeitslast erreicht hat.

terstützung erreicht. Dafür sprechen auch die Ergebnisse aus der Analyse der Stalling-Verzögerung in Abschnitt 7.1.

7.2.3 Diskussion

In erster Linie zeigt die Analyse der Zustandsallokation, dass das Supporter-Signalisierungsprotokoll funktioniert und Peers, die in den *Unterstützt*-Zustand wechseln, tatsächlich Unterstützung erhalten. Der gesamte Systemzustand konnte durch den Einsatz der Supporter-Strategie deutlich gebessert werden. So ist die Performanz beim Einsatz von einem bzw. vier statischen Servern sprunghaft, während der Verlauf der Zustände unter Verwendung der Supporter-Strategie ein deutlich gleichmäßigeres Bild zeigt. Bei einer Ankunftsrate von 12 Peers/Min. zeigt dies die Strategie mit sieben statischen Servern. Erhöht man die Ankunftsrate, so verschlechtert sich jedoch deren Performanz deutlich. Zu Beginn der Analyse wurde die Annahme getroffen, dass sich nach den ersten 5 Minuten (entspricht der Videolänge) eine annähernd konstante Arbeitslast einstellt. Dies konnte in den Resultaten nur bedingt nachvollzogen werden. Zum Einen sorgen Stalling-Effekte dafür, dass sich zu diesem Zeitpunkt deutlich mehr Peers im System befinden, als es der Erwartungswert vorschlägt. Zum Anderen sorgt die Absprungrate dafür, dass Peers ihren Datenbestand nicht längerfristig verteilen können. Betrachten Peers das Video immer bis zum Ende, so sollte sich ein gänzlich anderes Verhalten herausstellen.

Parameter	Wert
Verfügbare Server	{1,4,7} statische Server oder 10 Supporter
Ankunftsrate	Poisson-Prozess, 12 und 24 Peers/Min..

Tabelle 7.5: Abweichungen zur Grundkonfiguration (vgl. Tabelle 7.1) des Supporter-Szenarios

7.3 Basisvergleich

Die Konfiguration der Szenarien entspricht der in Tabelle 7.1, mit den Modifikationen, die in Tabelle 7.5 zusätzlich beschrieben sind. Die entscheidende Variable ist die Ankunftsrate. Die nachfolgende Analyse betrachtet die verschiedenen Server-Strategien in einem Szenario mit einer niedrigeren Ankunftsrate von 12 Peers/Min. und mit einer höheren Ankunftsrate von 24 Peers/Min im Hinblick auf deren Performanz. Ziel der Untersuchung ist der Nachweis, dass die Supporter-Strategie einen Kompromiss zwischen Reduktion der Server-Last und Reduktion von Stalling-Effekten realisiert. Zu erwarten ist:

- Bei einer Ankunftsrate von 12 Peers/Min. ist der Peer-Bedarf verhältnismäßig niedrig. Vorangegangene Tests haben gezeigt, dass die statischen Server generell eine sehr hohe Auslastung ihrer Upstream-Kapazität zeigen. Die Analyse sollte veranschaulichen, dass die Supporter-Strategie effektiv Server-Last einsparen kann, ohne dabei Stalling-Effekte zu vermehren.
- Bei einer Ankunftsrate von 24 Peers/Min. ist der Peer-Bedarf deutlich höher. Die Analyse soll zeigen, dass die Supporter-Strategie deutlich mehr Peers mit Daten versorgen muss, um Stalling-Effekte zu reduzieren.

7.3.1 Vergleich bei einer Ankunftsrate von 12 Peers/Min.

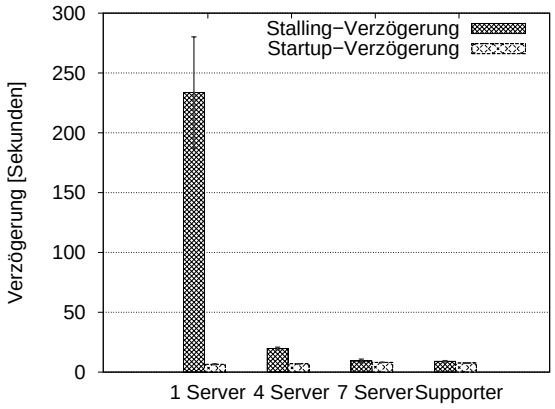
Tabelle 7.6 zeigt die Ergebnisse des Basisvergleichs bei einer Ankunftsrate von 12 Peers/Min. Während sich die Startup-Verzögerungen nur unmerklich voneinander unterscheiden, zeigen sich Stalling-Effekte vor allem bei einem bzw. vier statischen Servern. In diesem Szenario ist insbesondere der Vergleich der Supporter-Strategie mit vier statischen Servern interessant. Beide Strategien weisen einen annähernd gleichen Wert bezüglich der Server-Last auf, jedoch unterscheidet sich die Stalling-Verzögerung auf dem 95%-Perzentil enorm. Die Supporter-Strategie erreicht diesbzgl. eine Reduktion um den Faktor 2,19. Dieses Ergebnis ist durchaus beachtlich, besagt es doch, dass die Supporter-Strategie durch die inhärente Zulassungskontrolle eine bessere Verteilung von Daten erreicht, als dies vier statische Server vermögen, die konstant mit einer hohen Datenrate – dafür willkürlich – Daten an ihre Nachbarn im Schwarm senden. Die Analyse der Zustandsallokation in Abschnitt 7.2 hat gezeigt, dass sich die Anzahl aktiver Supporter der Performanz des Overlays anpasst. Erst dadurch ist es möglich, bei niedriger Server-Last eine konstant gute Dienstgüte zu erhalten. Dies spiegelt sich im Vergleich der Supporter-Strategie mit sieben statischen Servern wieder. Die Server-Last für sieben statische Server ist um den Faktor 1,45 höher, allerdings liefert die Supporter-Strategie auch hier ein besseres Ergebnis bezüglich der Stalling-Verzögerung.

Server-Strategie	Stalling (95%)	Startup (95%)	Peer-Upload	Server-Last
Statisch (1 Server)	233,62 s	6,53 s	4,15 GB	0,48 GB
Statisch (4 Server)	19,72 s	7,03 s	3,49 GB	1,82 GB
Statisch (7 Server)	9,37 s	8,12 s	2,94 GB	2,82 GB
Supporter-Strategie (10)	9,01 s	7,55 s	3,46 GB	1,95 GB

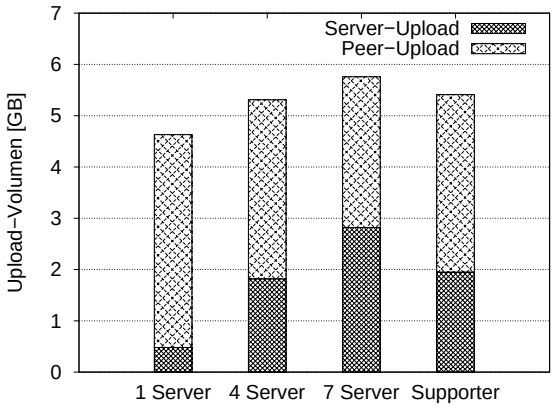
Tabelle 7.6: Direkter Vergleich der Server-Strategien bei einer Ankunftsrate von 12 Peers/Min.

7.3.2 Vergleich bei einer Ankunftsrate von 24 Peers/Min.

Tabelle 7.7 zeigt Ergebnisse bei einer mittleren Ankunft von 24 Peers/Min. Die Startup-Verzögerungen sind in diesem Szenario ebenfalls vernachlässigbar, da sie sich nur unmerklich voneinander unterscheiden. Das Upload-

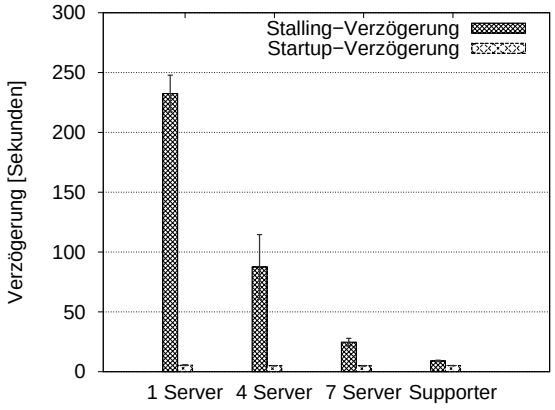


(i) Stalling- und Startup-Verzögerung

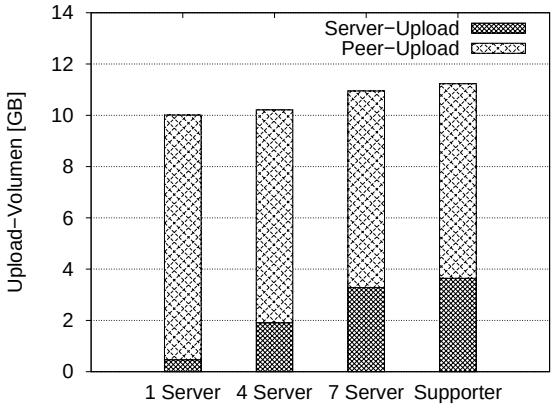


(ii) Server-Last und Upload-Volumen von Peers

Abbildung 7.8: Vergleich der Strategien untereinander bei einer Ankunftsrate von 12 Peers/Min.



(i) Stalling- und Startup-Verzögerung



(ii) Server-Last und Upload-Volumen von Peers

Abbildung 7.9: Vergleich der Strategien untereinander bei einer Ankunftsrate von 24 Peers/Min.

Volumen der statischen Server ist durch die Anzahl der Server limitiert. Dies drückt sich bei einer höheren Ankunftsrate durch ein steigendes Upload-Volumen der Peers aus. Je mehr Upload-Volumen die Server bereitstellen können, desto geringer fällt das Upload-Volumen der Peers aus. Die Supporter-Strategie muss bedingt durch die erhöhte Ankunftsrate mehr Peers mit Daten versorgen, wodurch sein Upload-Volumen sehr stark im Vergleich zur vorherigen Betrachtung ansteigt. Tatsächlich liegt die Supporter-Strategie bezüglich der Server-Last deutlich über den anderen Strategien. Den Grund hierfür liefern vermehrt auftretende Stalling-Effekte, die sich insbesondere bei den statischen Servern ausbilden. Die Supporter-Strategie wirkt diesen Effekten durch die Aktivierung zusätzlicher Supporter entgegen. Konsequenterweise steigt die Server-Last, dafür aber wird die Stalling-Verzögerung auf einem konstant niedrigen Niveau gehalten. Im direkten Vergleich mit sieben statischen Servern erreicht die Supporter-Strategie bezüglich der Stalling-Verzögerung eine Reduktion um den Faktor 2,74.

7.3.3 Diskussion

Die Ergebnisse stimmen mit der Erwartungshaltung überein: Die Supporter-Strategie ist in der Lage, den erwarteten Kompromiss zwischen Reduktion der Server-Last und Reduktion der Stalling-Verzögerung umzusetzen. Unter Verwendung dieser Strategie kann in Zeiten niedriger Last effektiv Server-Last eingespart werden, ohne dabei ein Verlust an Dienstgüte in Kauf nehmen zu müssen. In Zeiten hoher Last ist die Supporter-Strategie in der Lage, die

Server-Strategie	Stalling (95%)	Startup (95%)	Peer-Upload	Server-Last
Statisch (1 Server)	232,46 s	5,49 s	9,54 GB	0,47 GB
Statisch (4 Server)	87,51 s	5,03 s	8,3 GB	1,91 GB
Statisch (7 Server)	24,69 s	4,99 s	7,67 GB	3,28 GB
Supporter-Strategie (10)	9,01 s	5,11 s	7,59 GB	3,64 GB

Tabelle 7.7: Direkter Vergleich der Server-Strategien bei einer Ankunftsrate von 24 Peers/Min.

Dienstgüte auf einem guten Niveau zu halten, obgleich dann die Server-Last ansteigt. Das zeigt vor allem, dass der Supporter insbesondere dann gut einsetzbar ist, wenn der genaue Bedarf unbekannt ist.

Tabelle 7.8 zeigt für die Strategien der statischen Server die Faktoren, um die sich die Stalling-Verzögerung im Vergleich mit der Supporter-Strategie reduziert hat. Die Resultate sind enorm und zeigen, dass die Supporter-Strategie durchaus Potential hat.

Server-Strategie	Verbesserung der Stalling-Verzögerung um Faktor	
	bei 12 Peers/Min.	bei 24 Peers/Min.
Statisch (1 Server)	25,93	25,80
Statisch (4 Server)	2,19	9,71
Statisch (7 Server)	1,04	2,74

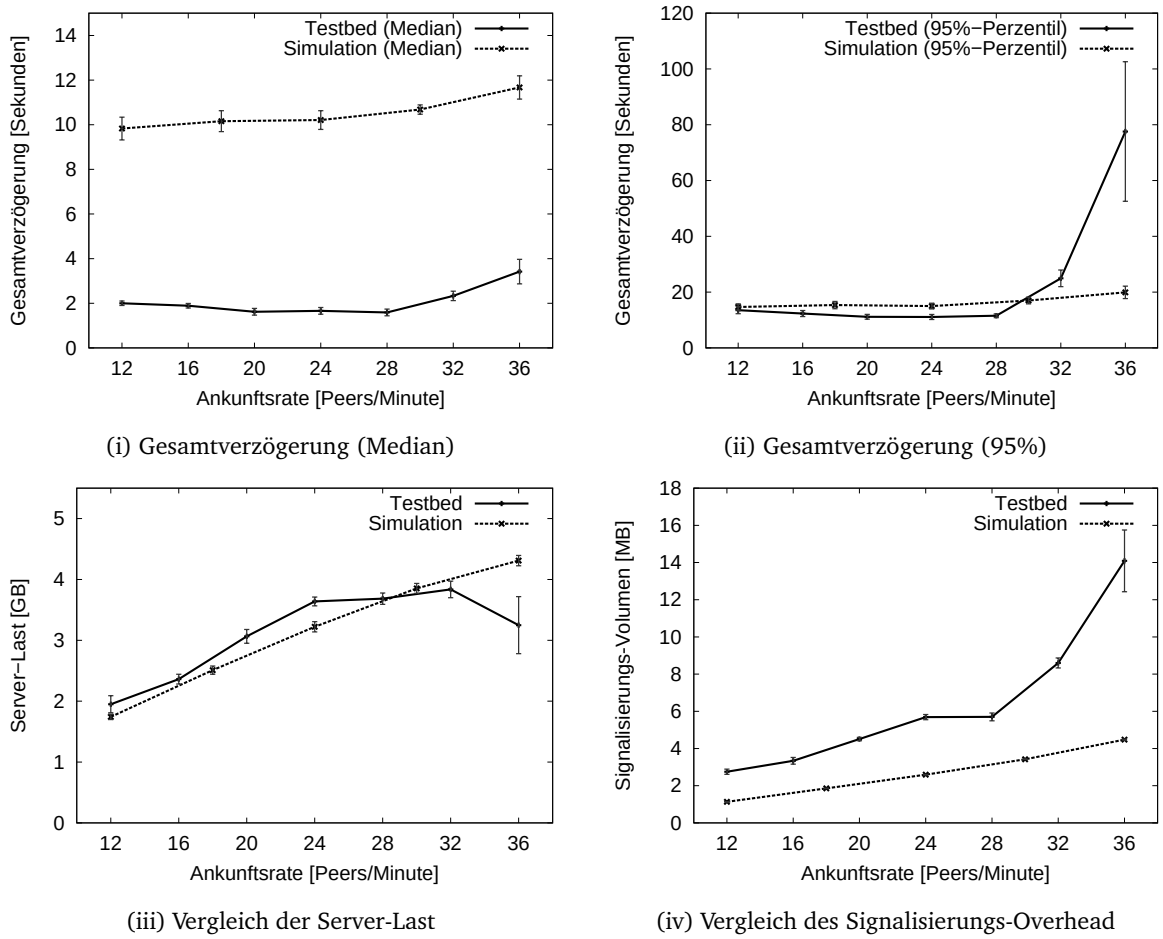
Tabelle 7.8: Reduktionsfaktoren bzgl. der Stalling-Verzögerung auf dem 95%-Perzentil der statischen Server im Vergleich mit der Supporter-Strategie

Da die Supporter-Strategie nur dann Daten an Peers sendet, wenn es unbedingt erforderlich ist, sinkt das transferierte Datenvolumen erheblich. Insbesondere für kleine Anbieter von Inhalten, die bspw. auf Cloud Computing zurückgreifen, um ihre Inhalte zu vertreiben, ist dies interessant. In der Regel erfolgt die Abrechnung bei Diensten des Cloud Computing über das transferierte Datenvolumen. Ist dies auf ein notwendiges Minimum beschränkt, so senken sich die Kosten des Anbieters.

7.4 Vergleich mit *Adaptive Server Allocation for Peer-Assisted Video-on-Demand (Pussep et al.)*

Die Supporter-Strategie ist bereits in [10] durch Simulation erprobt worden. Das zugrunde liegende Szenario der Simulation entspricht der in Tabelle 7.1 gezeigten Basiskonfiguration mit einer Ankunftsrate von 12 Peers/Min. Der Vergleich auf diesem Szenario kann allerdings nur schwer geführt werden, da sich die Implementierungen bzgl. der Messergebnisse für Stalling- und Startup-Verzögerung sehr stark voneinander unterscheiden.

Betrachtet man jedoch die Gesamtverzögerung, so lässt sich ein gemeinsamer Trend in der Performanz beider Implementierungen erkennen. Die Ankunftsrate variiert bei dieser Untersuchung im Bereich von 12 Peers/Min. bis 36 Peers/Min. Den Vergleich auf dem Median zeigt Abbildung 7.10(i). Beide Kurven zeigen einen ähnlichen Verlauf, allerdings unterscheiden sich die individuellen Messergebnisse teilweise um einen Faktor von bis zu 5. Die Gesamtverzögerung fällt für 50% der Peers in der Simulation demnach deutlich höher aus. Die konkreten Ergebnisse zur Stalling-Verzögerung sind in [10] mit 0.0s auf sowohl dem Median und dem 95%-Perzentil angegeben. Die negativen Auswirkungen einer steigenden Peer-Last zeigen sich in der Simulation im Prinzip nur durch die Startup-Verzögerung. Dies zeigt, dass obwohl ein gemeinsamer Trend in der Leistung beider Implementierungen zu erkennen ist, sich die Systeme stark voneinander unterscheiden. Abbildung 7.10(ii) zeigt den Vergleich auf dem 95%-Perzentil. Ein gemeinsamer Trend ist auch hier wieder zu erkennen. Die Implementierung im Simulator [10] zeigt ein konstant gutes Verhalten, während die im Rahmen der vorliegenden Arbeit erfolgte Implementierung deutlich an Performanz ab einer Ankunftsrate von 32 Peers/Min. einbüßt. Vermutlich handelt es sich bei der deutlich schlechteren Performanz bei einer mittleren Peer-Ankunft von 36 Peers/Min. um einen statistischen Ausreißer. Die Standardabweichung ist in diesem Szenario sehr hoch, so dass man davon ausgehen kann, dass die Messwerte nicht die tatsächliche Performanz wiedergeben.



(i) Gesamtverzögerung (Median)

(ii) Gesamtverzögerung (95%)

(iii) Vergleich der Server-Last

(iv) Vergleich des Signalisierungs-Overhead

Abbildung 7.10: Vergleich der Supporter-Strategie der vorliegenden Arbeit (Testbed) mit den Ergebnissen der Leistungsbewertung in [10] (Simulation).

Abbildung 7.10(iii) zeigt, dass die Ergebnisse der vorliegenden Arbeit ebenfalls die Tendenz in der Server-Last aus [10] bestätigen. Beide Kurven liegen sogar eng beieinander, wobei insbesondere der Anstieg der Testbed-Kurve fast linear ist. Die kleinen Abweichungen zum Verlauf der Simulations-Kurve lassen sich durch die etwas höhere Standardabweichung erklären, die für eine Evaluation in einem Testbed nichts ungewöhnliches ist. Das Szenario mit einer mittleren Ankunft von 36 Peers/Min. zeigt auch in der Analyse der Server-Last wieder erhebliche Schwankungen. Dass die Standardabweichung erneut sehr hoch ist, stützt die zuvor getroffene Vermutung eines statistischen Ausreißers in diesem Fall.

Abbildung 7.10(iv) zeigt schließlich den Signalisierungs-Overhead für beide Evaluationsmethoden. Die Ergebnisse in [10] gehen von einer konstanten Nachrichtengröße von 1500 Bytes aus. Die Nachrichten der Supporter-Implementierung für NextShare sind deutlich kleiner, so dass der Vergleich auf Basis der Anzahl an Nachrichten für beide Verfahren, multipliziert mit der mittleren Größe einer Nachricht der Supporter-Implementierung, erfolgt. Da sich die Größen der unterschiedlichen Nachrichtentypen leicht voneinander unterscheiden, muss die gewichtete mittlere Größe berechnet werden. Als Grundlage hierfür dient ein Evaluationsszenario mit einer mittleren Ankunft von 24 Peers/Min. Tabelle 7.9 schlüsselt die Berechnung auf. Die mittlere Größe liegt bei 131,28 Byte. Vergleicht man nun die Kurven der Ergebnisse aus Testbed und Simulation, so lässt sich auch hier wieder ein gemeinsamer Trend feststellen. Der Signalisierungs-Overhead in Simulation steigt linear mit zunehmender Ankunftsrate. Die Ergebnisse der Evaluation im Testbed zeigen einen quasi-linearen Verlauf, der erneut durch den Ausreißer bei einer Ankunftsrate von 36 Peers/Min. unterbrochen wird. Der Signalisierungs-Overhead im Testbed fällt in den meisten Fällen um ca. den Faktor 3 höher aus. Die Parametrisierung der Supporter-Strategie in der Simulation ist bzgl. der Sende-Rate von `request_support`-Nachrichten nicht ersichtlich. Da dieser Nachrichtentyp allerdings den Groß-

Nachrichtentyp	Größe	Anzahl Nachrichten	Anteil Nachrichten
`register_supporter`	166 Byte	10	0,022%
`register_peer`	139 Byte	1999	4,424%
`request_support`	131 Byte	41333	91,475%
`abort_support`	129 Byte	1843	4,079%
Summe	-	45185	100%
Gewichtetes Mittel	131,28 Byte		

Tabelle 7.9: Berechnung der gewichteten mittleren Größe einer Nachricht der Supporter-Implementierung anhand der Anzahl der Nachrichten in einem Szenario mit mittlerer Ankunft von 24 Peers/Min.

teil der Nachrichten stellt, kann man davon ausgehen, dass sich die Sende-Rate in beiden Evaluationsmethoden voneinander unterscheidet.

8 Evaluation des adaptiven Bandbreitenmanagements

Die Leistungsbewertung des in dieser Arbeit vorgestellten Mechanismus zum adaptiven Bandbreitenmanagement geht der Frage nach, wie sich der Ansatz im Vergleich zu anderen Mechanismen verhält. Insbesondere soll die Evaluation zeigen, ob die Interessen aller beteiligten Interessengruppen durch den Mechanismus gefördert werden können. Die Datenbeschaffung erfolgte pro Experiment durch 3-fache Wiederholung der korrespondierenden Testläufe. Eine empirische Untersuchung des Einflusses unterschiedlicher Parametrisierungen ist durch die Evaluation im Testbed nur bedingt möglich. Entsprechende Reduktionen bzw. Annahmen bzgl. der Parametrisierungen des Mechanismus erfolgen – sofern nötig – im jeweiligen Abschnitt zur entsprechenden Analyse. Die Studie zeigt die Resultate für die nachfolgend beschriebenen Untersuchungen:

- Güte der Identifikationsmetriken: Den Einstieg in die Evaluation liefert die Untersuchung der in Abschnitt 4.3.1 vorgestellten Identifikationsmetriken. Die Untersuchung betrachtet die Metriken zunächst in Isolation. Die Bewertung der Qualität einer Metrik erfolgt in instrumentierten Testbed-Läufen. Ziel ist es, herauszufinden, ob die Metriken die *korrekten* Peers identifizieren können. Die Analyse betrachtet die Metriken abschließend in Kombination, speziell vor dem Hintergrund einer möglichen Parametrisierung des adaptiven Bandbreitenmanagements für die nachfolgenden Experimente.
- Genereller Einfluss des adaptiven Bandbreitenmanagements: Dieses Experiment zeigt den Einfluss des adaptiven Bandbreitenmanagements bei alleiniger Verwendung und in Kombination mit den in Abschnitt 3.3 beschriebenen Mechanismen BNS und BU. Die Analyse vergleicht diese BM-Varianten gegen zwei unterschiedliche Basiskonfigurationen hinsichtlich der Ausgabemetriken Stalling- und Startup-Verzögerungen, Server-Last und Verkehrsvolumen nach Inter- und Intra-ISP-Datenverkehr. Innerhalb dieses Experiments setzt jeder ISP den Mechanismus des adaptiven Bandbreitenmanagements um. Die Supporter-Strategie stellt die erforderliche Server-Komponente.
- Einfluss des adaptiven Bandbreitenmanagements im Falle eines Early Adopters: Diese Untersuchung widmet sich dem Einfluss des Mechanismus, wenn ihn nur ein ISP umsetzt. Die Analyse vergleicht die Leistung hinsichtlich der Ausgabemetriken Stalling- und Startup-Verzögerungen, Server-Last und Verkehrsvolumen nach Inter- und Intra-ISP-Datenverkehr auf Basis der verschiedenen ISPs.

8.1 Güte der Identifikationsmetriken

Dieser Abschnitt widmet sich zunächst der Analyse der in Kapitel 4 vorgestellten Identifikationsmetriken. Die Konfiguration der für die Testdurchführung notwendigen Parameter zeigt Tabelle 8.1. Die zugrunde liegende Topologie entspricht der Beschreibung in Abschnitt 6.2, mit der Ausnahme der Reduktion auf zwei ISP-Domänen und einer Server-Farm.

Die Identifikationsmetriken bewerten das Verhalten der Peers unter verschiedenen Aspekten. So würde bspw. die Metrik $S(t)$ Peers mit einem hohen Gütewert versehen, sofern sie über die Dauer der Video-Wiedergabe hinaus im Schwarm bleiben und dabei weiter Daten an andere Peers verteilen. Das Experimentdesign nutzt dies aus und berücksichtigt durch eine entsprechende Konfiguration, dass sich verschiedene Peer-Gruppen bilden. Diese Peer-Gruppen unterscheiden sich in der Variation der Konfiguration eines Parameters, bspw. in dem konkreten Wert für die Verweilzeit. Durch die Parametrisierung ist zu erwarten, dass eine Peer-Gruppe sich innerhalb des Schwarms besser verhält und von der zu untersuchenden Metrik generell durch bessere Gütewerte bewertet wird. Beispiel: Betrachtet man die besten 10% aller bewerteten Peers, dann sollte sich herausstellen, dass eine bestimmte Gruppe in dieser Untermenge besonders häufig vertreten ist. Der Nachweis der Identifikationsgüte erfolgt dementsprechend nach dem Anteil der *korrekt* identifizierten Peers bei einer variierenden Anzahl von zu befördernden Peers zwischen 1% und 100%.

Parameter	Konfiguration
Laufzeit des Szenarios	30 Min.
Videolänge	5 Min.
Bitrate	540 kb/s
Topologie	**2 ISP-Domänen, 1 Server-Farm**
Verfügbare Server	3 (statisch)
Server-Kapazität (Upstream)	**2048 kb/s**
Peer-Zugangsprofile (Upstream)	**Mid (1024 kb/s), High (2048 kb/s)**
Peer-Zugangsprofile (Downstream)	**Mid (1024 kb/s), High (2048 kb/s)**
Zugangsprofil-Verteilung	**50% Mid, 50% High**
Ankunftsrate	Poisson-Prozess, 24 Peers/Min.
Peer-Verhalten	Absprung nach Verweilzeit
Verweilzeit	**10 Minuten**
Seeding-Zeit	$\frac{1}{2}\times$**,** $\mathbf{1}\times$**,** $\mathbf{2}\times$ **Videolänge**
Seeding-Zeit-Verteilung	**33%** $\frac{1}{2}\times$**, 33%** $\mathbf{1}\times$**, 33%** $\mathbf{2}\times$
Peer-Selektion	**Standard**
Block-Größe	32 kB

Tabelle 8.1: Grundkonfiguration der Testszenarien zu den Identifikationsmetriken

8.1.1 Untersuchung der Auslastung der Upload-Kapazität $U(t)$

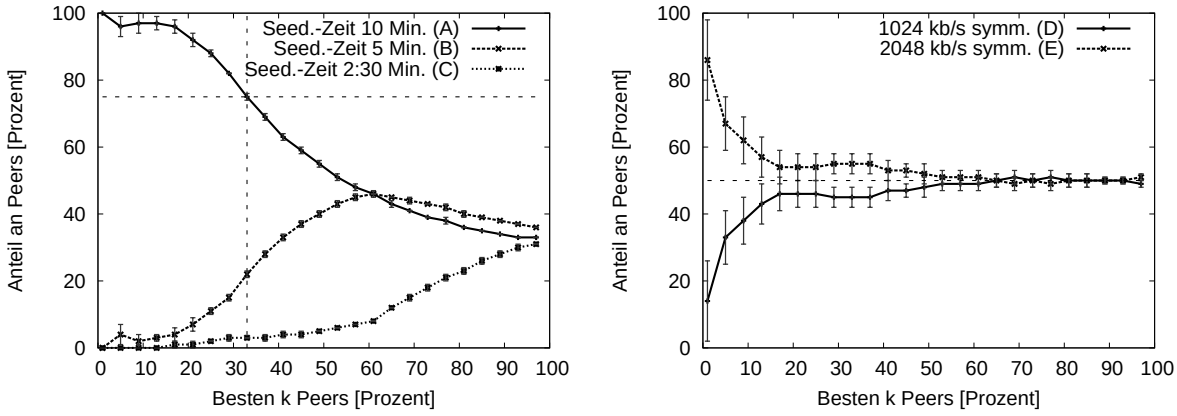
Dieser Abschnitt analysiert die Identifikationsgüte der Metrik $U(t)$. Die Konfiguration des Szenarios entspricht der Grundkonfiguration aus Tabelle 8.1. Diese Konfiguration unterteilt die Gesamtanzahl der Peers nach der Verweilzeit und dem Zugangsprofil. Die Untersuchung geht von der Annahme aus, dass jeder Peer für insgesamt 10 Minuten im System verweilt, aber nach einer gewissen Verweilzeit das aktive Weitergeben von Daten unterbindet.

Die Gruppen nach Verweilzeit teilen sich gleichmäßig auf, so dass die Anzahl der Peers pro Gruppe ungefähr 1/3 der Gesamtzahl der Peers entspricht. Die Gruppen beinhalten Peers mit den folgenden Eigenschaften:

- Gruppe A: Peers dieser Gruppe bleiben für insgesamt 10 Minuten im Schwarm und senden konstant Daten an andere Peers über diese Zeit. Es ist zu erwarten, dass diese Gruppe von der Identifikationsmetrik $U(t)$ am besten bewertet wird.
- Gruppe B: Peers dieser Gruppe bleiben für insgesamt 10 Minuten im Schwarm, senden allerdings nur über eine Dauer von insgesamt 5 Minuten Daten an andere Peers. Es ist zu erwarten, dass die Identifikationsmetrik $U(t)$ diese Peers schlechter bewertet als Peers der Gruppe A und besser als Peers der Gruppe C. Der Anteil von Peers dieser Gruppe unter den am besten bewerteten Peers sollte sich erst ab einer Selektion von 33% signifikant erhöhen.
- Gruppe C: Peers dieser Gruppe bleiben für insgesamt 10 Minuten im Schwarm, senden allerdings nur über eine Dauer von insgesamt 2:30 Minuten Daten an andere Peers. Es ist zu erwarten, dass die Identifikationsmetrik $U(t)$ Peers dieser Gruppe generell schlechter bewertet, als Peers der Gruppen A und B. Der Anteil von Peers dieser Gruppe unter den Besten sollte sich dementsprechend erst ab einer Selektion von 66% der bewerteten Peers signifikant erhöhen.

Die Gruppen nach Zugangsprofil teilen sich ebenfalls gleichmäßig auf, so dass die Anzahl der Peers pro Gruppe ungefähr der Hälfte der Gesamtzahl aller Peers entspricht. Die Gruppen beinhalten Peers mit den folgenden Eigenschaften:

- Gruppe D: Diese Gruppe beinhaltet Peers, deren Bandbreiten symmetrisch auf 1024 kb/s festgelegt sind.
- Gruppe E: Diese Gruppe beinhaltet Peers, deren Bandbreiten symmetrisch auf 2048 kb/s festgelegt sind.



(i) Identifikationsgüte von $U(t)$

(ii) Verhalten bzgl. Peer-Heterogenität

Abbildung 8.1: Analyse der Identifikationsmetrik Auslastung der Upload-Kapazität $U(t)$

Da die Identifikationsmetrik $U(t)$ unabhängig von der Bandbreite und damit der Heterogenität der Peers arbeitet, ist zu erwarten, dass die Verteilung von Peers beider Gruppen unter den Besten bei 50% liegt und damit der ursprünglichen Verteilung entspricht.

Abbildung 8.1(i) zeigt die Auswertung der Gruppen nach Verweilzeit. Es ist deutlich zu erkennen, dass Peers aus Gruppe *A* bei einer Auswahl der besten 33% aller Peers einen deutlichen Vorzug durch die Metrik erhalten. Dennoch stellen sich bereits kleine Fehler ein, denn unter den besten 33% Peers befinden sich nur zu 75% Peers aus Gruppe *A*. Die restlichen 25% teilen sich Gruppe *B* (22%) und *C* (3%), wobei Peers aus Gruppe *B* deutlich überwiegen. Nach Selektion der besten 33% wächst der Anteil von Peers aus Gruppe *B* stark an und erreicht seinen Höchstpunkt – wie zu erwarten – bei 62%. Dies entspricht einer Abweichung von der ursprünglichen Verteilung um lediglich 5% (67% aller Peers gehören zu den Gruppen *A* und *B*). Erst nach den besten 62% aller Peers steigt der Anteil von Gruppe *C* signifikant an. Die Kurven der Gruppen *A* und *B* nehmen ab den besagten Grenzwerten von ca. 33% und 62% wieder ab. Nach der ursprünglichen Verteilung ($\approx 1/3$ aller Peers pro Gruppe) muss der Anteil von Peers aus schlechter bewerteten Gruppen zunehmen, was zur Folge hat, dass der Anteil gut bewerteter Peers im gleichen Verhältnis abnimmt.

Abbildung 8.1(ii) zeigt die Auswertung der Gruppen nach Zugangsprofil. Die Grafik zeigt, dass die Peer-Heterogenität erst ab einer Auswahl der 15% besten Peers einigermaßen gewahrt ist (45% aus Gruppe *D*, 55% aus Gruppe *E*). Verringert man den Anteil der besten Peers, so überwiegt der Anteil der Peers mit hohem Zugangsprofil (Gruppe *E*). Die Ursache hierfür ist nicht in der Metrik zu suchen, sondern in der Funktionsweise des P2P-Systems selbst. Der NextShare-Client nutzt bei der Verteilung von Video-Daten das G2G-Protokoll (vgl. Abschnitt 2.3.3) und dieses bevorzugt bei Verbindungsherstellung Peers, die sich als gute Weiterleiter ausgezeichnet haben. Durch ein stärkeres Zugangsprofil kann ein Peer mengenmäßig mehr Daten weiterleiten, als ein Peer mit schwächerem Zugangsprofil. Das lastet die Upload-Kapazität dieser Peers stärker aus, so dass sich nach $U(t)$ folgerichtig diese Peers mit einem höheren Anteil unter den besten Peers befinden. Der Effekt ist allerdings nicht stark, da sich eine Gleichverteilung bereits annähernd bei Selektion der besten 15% einstellt.

Tabelle 8.2 zeigt den Anteil für jede Gruppe an der Gesamtzahl der Peers und liefert die Mittelwerte der Bewertungen durch die Metrik. Zu erkennen ist, dass die individuellen Peer-Bewertungen im Durchschnitt relativ niedrig ausfallen. Das liegt vermutlich an der ohnehin guten Performanz des Systems in diesem Szenario. In der Praxis würde man in solch einem Fall die Beförderung von Peers zusätzlich von einem Grenzwert abhängig machen, so dass eine Beförderung von zuvielen mittelmäßigen Peers auszuschließen ist. Der durchschnittliche Gütewert für Gruppe *D* und *E* liegt ungefähr gleich auf. Die geringen Standardabweichungen bestätigen, dass die Mittelwerte der Bewertungen pro Gruppe relativ stabil sind.

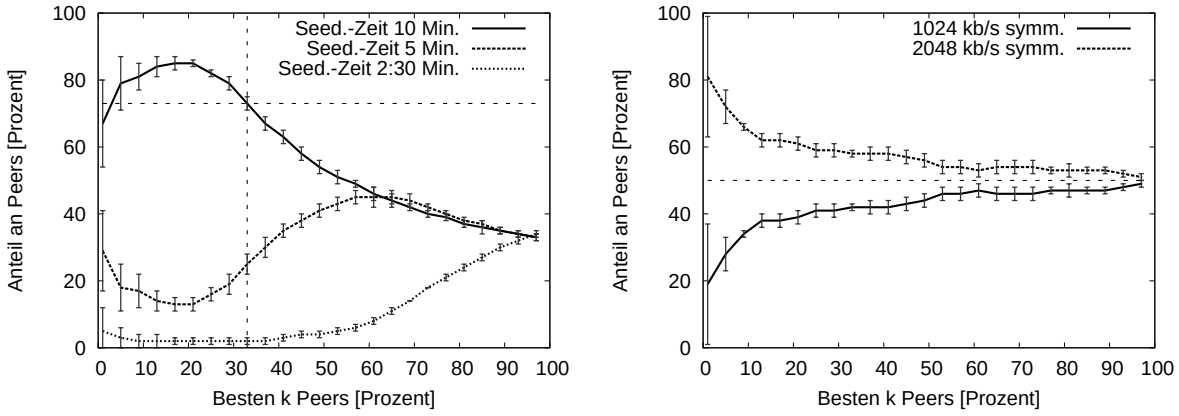
Insgesamt ist das Ergebnis zufriedenstellend, denn trotz der hohen Dynamik eines P2P-Systems wurden größtenteils die Peers identifiziert, die man zu identifizieren hoffte. Dass durch die Funktionsweise des G2G-Protokolls Peers mit höherer Bandbreite gewissermaßen bevorzugt behandelt werden, leistet dem Mechanismus eines adaptiven Band-

Gruppen nach Verweilzeit				
Gruppe	**Größe (Mittelw.)**	**Anteil an Gesamtzahl**	$U(t)$ (Mittelw.)	$U(t)$ (Standardabw.)
Gruppe *A* (10 Min.)	259 Peers	33%	0,27	0,01
Gruppe *B* (5 Min.)	282 Peers	36%	0,17	0,00
Gruppe *C* (2:30 Min.)	238 Peers	31%	0,11	0,00
Gruppen nach Zugangsprofil				
Gruppe	**Größe (Mittelw.)**	**Anteil an Gesamtzahl**	$U(t)$ (Mittelw.)	$U(t)$ (Standardabw.)
Gruppe *D* (1024 kb/s)	384 Peers	49%	0,18	0,00
Gruppe *E* (2048 kb/s)	395 Peers	51%	0,19	0,01

Tabelle 8.2: Anteil individueller Gruppen an der Gesamtzahl mit durchschnittlichem Gütewert

breitenmanagements keinen Abbruch. Im Gegenteil: Auch wenn die Identifikationsmetrik $U(t)$ Peer-Heterogenität grundsätzlich nicht beachten soll, so ist der positive Effekt durch die Beförderung eines Peers mit stärkerem Zugangsprofil nach aller Wahrscheinlichkeit höher, als bei Beförderung eines Peers mit schwächerem Zugangsprofil.

8.1.2 Untersuchung des Verhältnisses zwischen Upload und Download $S(t)$



(i) Identifikationsgüte von $S(t)$

(ii) Verhalten bzgl. Peer-Heterogenität

Abbildung 8.2: Analyse der Identifikationsmetrik Upload-Download-Verhältnis $S(t)$

Gegenstand dieses Abschnitts ist die Untersuchung der Identifikationsgüte der Metrik $S(t)$. Die Konfiguration des Szenarios entspricht der Grundkonfiguration aus Tabelle 8.1. Die durch die Konfiguration entstehenden Gruppen sind daher äquivalent mit denen aus der Analyse von $U(t)$ in Abschnitt 8.1.1. Dadurch ist insbesondere ein Vergleich der Identifikationsgüte zwischen beiden Metriken möglich. Es gilt die selbe Erwartungshaltung bzgl. der Ergebnisse pro Gruppe, wie bereits in Abschnitt 8.1.1 beschrieben.

Abbildung 8.2(i) zeigt, dass der prinzipielle Verlauf der Kurven denen aus Abbildung 8.1(i) ähnelt. Jedoch erreicht der Anteil von Gruppe *A* einen deutlich niedrigeren Höchstwert (86%) als bei der Bewertung nach der Upload-Auslastung (100%). Betrachtet man den Anteil von Gruppe *A* unter den besten 33% Peers, so liegt dieser bei ca. 73%. Im Vergleich zur Bewertung nach der Upload-Auslastung $U(t)$ ist der Wert an dieser Stelle ebenfalls geringer. Der Verlauf der Kurven, welche die Gruppen *B* und *C* repräsentieren, erinnert ebenfalls an die vorhergehende Betrachtung. Dass sich die Bewertung der Peers so verhält ist logisch, denn die Peer-Verteilung, und damit auch die individuellen Gruppengrößen, ist identisch mit der Analyse aus Abschnitt 8.1.1.

Abbildung 8.2(ii) zeigt die Auswertung der Gruppen nach Zugangsprofil. Die Grafik zeigt, dass die Heterogenität der Peers erst ab einer Selektion von 50% der besten Peers gewahrt ist (45% aus Gruppe *D*, 55% aus Gruppe

E). Im Vergleich zu den Ergebnissen nach Zugangsprofil bei einer Bewertung durch die Upload-Auslastung $U(t)$ zeigt sich ein deutlich schlechteres Verhalten (vgl. Abschnitt 8.1.1). Verringert man den Anteil der besten Peers auf weniger als 50%, steigt der Fehler bzgl. der Peer-Heterogenität stark an. Die Metrik scheint Peers mit einem höheren Zugangsprofil zu bevorzugen. Da die Berechnungsformel für $S(t)$ grundsätzlich unabhängig von der Heterogenität arbeitet, ist auch hier der Grund wieder in der Funktionsweise des G2G-Protokolls (vgl. Abschnitt 8.1.1) zu suchen.

Die Ergebnisse zeigen, dass nach $S(t)$ größtenteils die erhofften Peers identifiziert werden. Allerdings ist der Fehler in der Identifikationsgüte größer, als dies unter Berechnung von Gütewerten nach Metrik $U(t)$ der Fall ist. Ab einer Selektion von 70% der besten Peers, besteht zwischen den beiden Metriken jedoch kaum noch ein Unterschied. In der Praxis ist man aller Voraussicht nach nicht daran interessiert, 70% der Gesamtheit aller Peers innerhalb einer ISP-Domäne mit einem erhöhten Zugangsprofil auszustatten.

Gruppen nach Verweilzeit											
Selektion in %	**Idealer Anteil**			**Nach $U(t)$**			**Nach $S(t)$**			**Fehler**	
	Gr. A	Gr. B	Gr. C	Gr. A	Gr. B	Gr. C	Gr. A	Gr. B	Gr. C	$U(t)$	$S(t)$
33%	100%	0%	0%	75%	22%	3%	73%	25%	3%	25%	27%
69%	50%	50%	0%	41%	44%	15%	42%	44%	14%	15%	14%

Tabelle 8.3: Betrachtung des Anteils fehlerhaft identifizierter Peers an den Übergangsstellen 33% und 69% bei Gruppierung nach Verweilzeit

Dennoch muss festgehalten werden, dass $S(t)$ die Gruppen deutlich schlechter voneinander separiert. Diese Tatsache zeigt sich insbesondere bei der Selektion einer geringen Menge von Peers (Bereich 1%-30%). In diesem Bereich zeigt Metrik $U(t)$ (vgl. Abschnitt 8.1.1) eine deutlich bessere Separierung. Idealerweise befinden sich unter den besten 33% nur Peers aus Gruppe A, resp. unter den besten 69% nur Peers aus den Gruppen A und B. Tabelle 8.3 zeigt den absoluten Fehler an diesen Stellen vergleichend für beide Metriken bei der Separierung der Gruppen nach Verweilzeit an. Man erkennt, dass bei einer Auswahl der besten 33% Peers $U(t)$ den geringeren Fehler zeigt. Betrachtet man den Fehler bei einer Auswahl der besten 69% Peers, so kehrt sich dies um: Beide Metriken liefern annähernd identische Ergebnisse, allerdings scheint hier $S(t)$ besser abzuschneiden als $U(t)$.

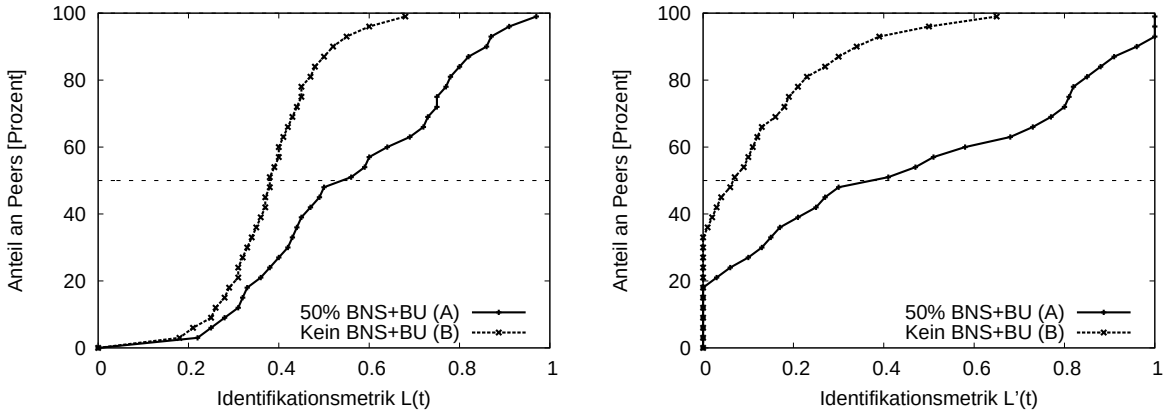
Die Tatsache, dass $S(t)$ eine schlechtere Separierung zwischen den Gruppen nach Verweilzeit und den Gruppen nach Zugangsprofil im Bereich 1%-30% zeigt, ist als grundlegendes Ergebnis festzuhalten. Auf Grund dieses Ergebnisses verzichtet die Bewertung der Peers in den folgenden Experimenten auf eine Bewertung nach $S(t)$, da Metrik $U(t)$ in der Lage ist, dieselben Peer-Gruppen in dem gewünschten Zielbereich mit einem geringeren Fehler zu identifizieren.

8.1.3 Untersuchung der Lokalitätsmetrik $L(t)$ und $L'(t)$

Parameter	**Konfiguration**
Topologie	3 ISP-Domänen
Server-Kapazität (Upstream)	1024 kb/s
Peer-Zugangsprofil (Upstream)	512 kb/s
Peer-Zugangsprofil (Downstream)	2048 kb/s
Zugangsprofil-Verteilung	-
Verweilzeit	1× Videolänge
Peer-Selektion	50% in AS 1 BNS+BU, Rest Standard

Tabelle 8.4: Abweichungen zur Grundkonfiguration (vgl. Tabelle 8.1) des Identifikationsmetriken-Szenarios

Die Analyse der Lokalitätsmetriken $L(t)$ und $L'(t)$ (vgl. Abschnitt 4.3.1) ist Gegenstand dieses Abschnitts. Die Konfiguration der Evaluationsszenarien basiert auf Tabelle 8.1 mit den Modifikationen, die in Tabelle 8.4 zusätzlich beschrieben sind. Insbesondere ist der Vergleich der Identifikationsgüte zwischen $L(t)$ und $L'(t)$ angestrebt. $L(t)$



(i) CDF zu $L(t)$ (ii) CDF zu $L'(t)$

Abbildung 8.3: Vergleich der Peer-Bewertungen anhand der Lokalitätsmetrik $L(t)$ (i) und $L'(t)$ (ii)

verspricht rein intuitiv bereits gute Werte. $L'(t)$ ist durch eine komplexere Funktion beschrieben, schließt allerdings die Bewertung von zuviel Lokalität mit in die Berechnung ein. Der Vergleich soll zeigen, ob die Identifikationsgüte beider Lokalitätsmetriken mind. gleich gut ist.

Die Topologie besteht aus drei AS. Das Peer-Verhalten variiert zwischen den Peers, die unterschiedlichen ISPs angehören. So nutzen im Mittel 50% der Peers aus AS 1 lokalitätsfördernde Mechanismen. Alle anderen Peers verhalten sich regulär. Die Gesamtzahl der Peers pro ISP ist gleichverteilt. Die Güte der Lokalitätsmetriken $L(t)$ und $L'(t)$ wird anhand des Anteils identifizierter Peers der nachfolgend beschriebenen Gruppen gemessen:

- Gruppe *A*: Peers dieser Gruppe gehören ISP 1 an und befinden sich in AS 1. 50% dieser Peers nutzen BNS und BU (vgl. Abschnitt 3.3). Es ist zu erwarten, dass der mittlere Gütewert durch Berechnung anhand von $L(t)$ und $L'(t)$ für Peers dieser Gruppe höher ausfällt, als für Peers aus Gruppe *B*. Der akkumulierte Inter-ISP-Datenverkehr über allen Peers dieser Gruppe sollte ebenfalls deutlich geringer ausfallen, als für Peers aus Gruppe *B*.
- Gruppe *B*: Peers dieser Gruppe gehören ISP 1 und ISP 2 an und befinden sich in den entsprechenden AS. Keiner dieser Peers nutzt BNS und BU. Es ist zu erwarten, dass der mittlere Gütewert durch Berechnung anhand von $L(t)$ und $L'(t)$ für Peers dieser Gruppe geringer ausfällt, als für Peers aus Gruppe *A*.

Abbildung 8.3(i) zeigt zunächst die CDF (Cumulative Distribution Function) der Gütewerte aller Peers nach $L(t)$. Jeder Peer aus Gruppe *B* steht im Mittel mit 2/3 Peers aus einer anderen ISP-Domäne in Kontakt. Dies spiegelt die CDF gut wieder, für 50% der Peers in Gruppe *B* liegt der Lokalitätswert bei max. 0,38. Für Peers der Gruppe *A* gilt dies nicht. Dadurch, dass 50% dieser Peers BNS und BU anwenden, stellen sie bevorzugt Verbindungen zu anderen Peers innerhalb ihrer ISP-Domäne her. Die CDF lässt dies deutlich erkennen, da bereits 50% der Peers aus Gruppe *A* einen Lokalitätswert von maximal 0,54 aufweisen. Diese positive Tendenz zeigt sich über den gesamten Verlauf der CDF zu Gruppe *A*. Abbildung 8.3(ii) zeigt die CDF nach $L'(t)$ auf denselben Daten. Der Verlauf der Kurven hat daher dieselbe Tendenz. Die Grafik zeigt jedoch, dass die Metrik Peers der Gruppe *A* stärker von Peers der Gruppe *B* abgrenzt.

Bisher ist lediglich der prinzipielle Einfluss von BNS und BU auf die Lokalitätswerte gezeigt worden. Im Folgenden soll der Frage nachgegangen werden, ob die Metrik in der Lage ist, von den Peers aus Gruppe *A* diejenigen zu identifizieren, die lokalitätsfördernde Mechanismen einsetzen. Die Resultate bzgl. der Identifikationsgüte für $L(t)$ und $L'(t)$ zeigt Abbildung 8.4. Beide Metriken zeigen ein nahezu identisches Verhalten. Ab einer Selektion der besten 50% aller Peers fällt der Anteil korrekt identifizierter Peers stark ab. Dies ist logisch, da die ursprüngliche Verteilung der Peers vorsieht, dass lediglich 50% BNS und BU anwenden. Unter den besten 50% aller Peers befinden sich, bei Anwendung beider Metriken, 92% der gewünschten Peers.

Die Ergebnisse zeigen, dass beide Lokalitätsmetriken in der Lage sind, mit einem akzeptablen Fehler die gewünschten Peers zu identifizieren. Der Vergleich zwischen $L(t)$ und $L'(t)$ hat gezeigt, dass in den folgenden Experimenten

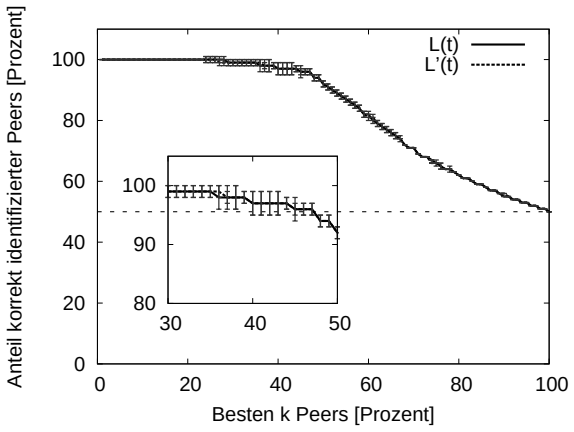


Abbildung 8.4: Analyse der Identifikationsgüte der Lokalitätsmetrik $L(t)$ und $L'(t)$

$L'(t)$ eingesetzt werden kann, ohne dabei an Identifikationsgüte im allgemeinen Fall einzubüßen. Die Verwendung von $L'(t)$ ist vorzuziehen, da diese Lokalitätsmetrik Problemfälle in Bezug auf zuviel Lokalität behandelt.

8.1.4 Kombinationsmöglichkeiten

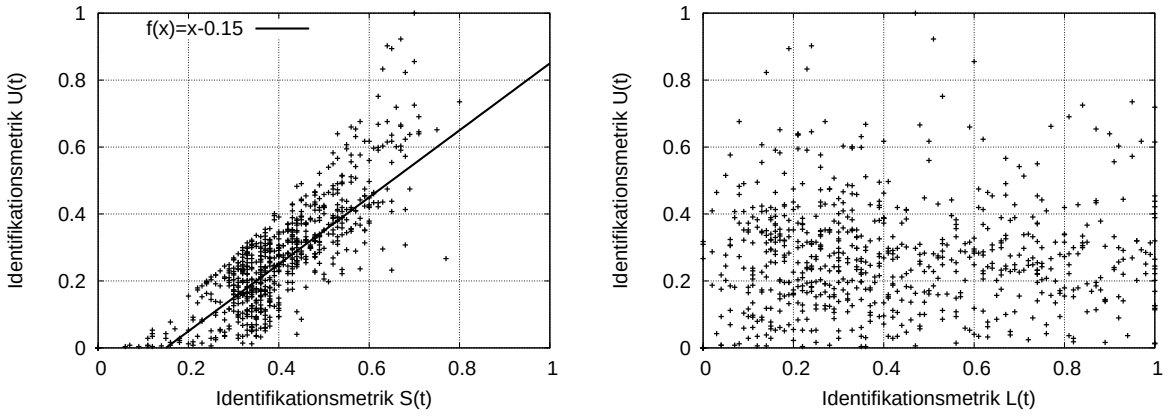
Die durchgeführten Tests treffen einige Annahmen bzgl. der Verteilung von Peer-Eigenschaften, insbesondere in Bezug auf Zugangsprofile, Peer-Selektionsstrategie und Aufteilung nach ISP-Domäne. Durch die limitierte Größe des Testbeds kann diese Verteilung nicht exakt einer solchen entsprechen, die man in der Praxis vorfindet. Dies hat ebenfalls Einfluss auf die Verteilung der Gütewerte, welche die Identifikationsmetriken liefern. Aus diesem Grund ist es schwierig, im Rahmen der vorliegenden Arbeit treffende Aussagen zur Kombinierbarkeit unterschiedlicher Metriken zu treffen.

Grundsätzlich sind die Identifikationsmetriken auf mehrere Arten kombinierbar. Zwei konkrete Möglichkeiten stellt Abschnitt 4.3.1 vor. Insbesondere die stufenweise Kombination der Metriken scheint interessant und soll im Zuge der nachfolgenden Experimente Verwendung finden. Bei dieser Kombinationsmöglichkeit erfolgt eine Filterung der Gesamtheit aller bewerteter Peers zuerst nach einer primären Identifikationsmetrik. Dies liefert eine reduzierte Peer-Menge. Anhand einer sekundären Identifikationsmetrik filtert man diese Menge und erhält als Resultat die best-bewerteten Peers anhand beider Metriken.

Die bisherigen Ergebnisse haben gezeigt, dass sich bei der Identifikation von Peers hinsichtlich ihrer Aktivität im Schwarm $U(t)$ etwas besser verhält als $S(t)$. Dass eine Kombination dieser beiden Metriken keinen signifikanten Informationsgewinn liefert, zeigt Abbildung 8.5(i). In der Abbildung erkennt man, dass sich nahezu alle Gütewert-Paare um die Kurve $f(x) = x - c$, mit $c = 0{,}15$ ansiedeln. Grundsätzlich ist man natürlich an Peers interessiert, die bzgl. beiden Metriken einen guten Wert versprechen (also nahe einer Diagonalen liegen). Liegen jedoch *alle* Gütewert-Paare nahe an einer solchen, linear ansteigenden Kurve, dann zeigen beide Identifikationsmetriken ein ähnliches Bewertungsverhalten und damit auch eine ähnliche Identifikationsgüte (vgl. Ergebnisse aus Abschnitt 8.1.2). Der Informationsgewinn durch die Kombination der Metriken ist daher gering. Die Parameterbelegung $c = 0{,}15$ gibt in diesem Fall lediglich an, dass $U(t)$ die Peers generell etwas schlechter bewertet als $S(t)$.

Betrachtet man die Kombination von $L(t)$ mit $U(t)$, so stellt sich ein grundsätzlich anderes Verhalten heraus. Die Gütewert-Paare verteilen sich nicht entlang einer linear ansteigenden Kurve, sondern großflächig über die Ebene. Durch die Kombination dieser beiden Metriken ist durchaus ein Informationsgewinn zu erzielen: Würde man nur anhand von $U(t)$ die besten Peers selektieren, so hätte man kein Indiz über das Lokalitätsverhalten der Peers. Filtert man die resultierende Peer-Menge erneut, aber diesmal nach $L(t)$, so erhält man als Resultat Peers mit gutem Upload- und Lokalitätsverhalten.

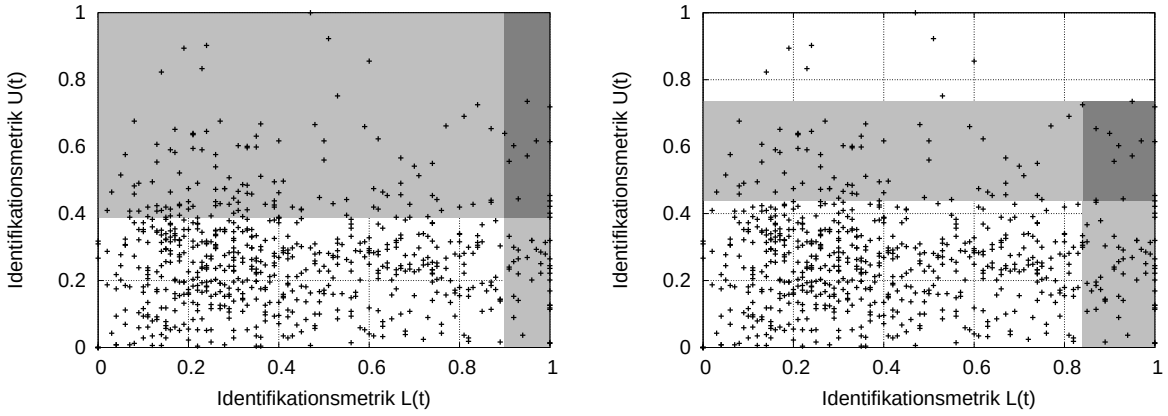
Die Reihenfolge einer solchen, stufenbasierten Filterung kann von entscheidender Bedeutung sein. Durch die primäre Selektion können bereits Peers aus der Ergebnismenge gefiltert werden, an denen man interessiert ist, die man aber erst durch die sekundäre Selektion identifizieren würde. Dies soll anhand eines Beispiels verdeutlicht werden.



(i) Verteilung nach $S(t)$ und $U(t)$

(ii) Verteilung nach $L(t)$ und $U(t)$

Abbildung 8.5: Verteilung von Gütewerten in Abhängigkeit div. Identifikationsmetriken



(i) Primär $U(t)$ (20%), sekundär $L(t)$ (10%)

(ii) Primär $L(t)$ (10%), sekundär $U(t)$ (20%)

Abbildung 8.6: Die Reihenfolge der Selektion hat Einfluss auf die resultierende Menge der best-bewerteten Peers. Die hellgrauen Flächen zeigen jeweils den Bereich, der von einer Metrik selektiert werden würde. Die dunkelgrauen Flächen zeigen die resultierende Ergebnismenge.

Abbildung 8.6 zeigt die Wirkungsweise der Funktion und den Einfluss auf die resultierende Menge beispielhaft für die Kombinationsmöglichkeiten von $U(t)$ mit $L(t)$:

(i) Primäre Selektion nach $U(t)$ (20% der besten Peers), sekundäre Selektion nach $L(t)$ (10% der besten Peers): Durch diese Reihenfolge identifiziert man zuerst Peers, die sich durch gutes Upload-Verhalten bewährt haben. Die sekundäre Selektion dient dazu, unter diesen Peers diejenigen zu identifizieren, die lokale Verbindungen bevorzugen.

(ii) Primäre Selektion nach $L(t)$ (10% der besten Peers), sekundäre Selektion nach $U(t)$ (20% der besten Peers): Durch diese Reihenfolge identifiziert man zuerst Peers, die ein gutes Lokalitätsverhalten zeigen. Dadurch können bereits gute Uploader aus der resultierenden Menge gefiltert werden, so dass die sekundäre Selektion nicht mehr das gewünschte Ergebnis liefern mag.

Dieses Problem besteht grundsätzlich, unabhängig von der Reihenfolge eingesetzter Identifikationsmetriken. Eine primäre Selektion durch Identifikationsmetrik $U(t)$ erscheint sinnvoll, da man im Sinne einer guten Performanz zunächst die Peers identifizieren möchte, die möglichst viel von ihren Ressourcen innerhalb des Schwarms einsetzen.

8.1.5 Diskussion

Die Ergebnisse zur individuellen Identifikationsgüte der Metriken zeigen, dass $U(t)$ annähernd dieselben Peers identifiziert, wie $S(t)$. Dies könnte sich anders verhalten, wenn die Metriken mit einer größeren Heterogenität hinsichtlich der Zugangsprofile einzelner Peers konfrontiert würden. Innerhalb des selben Szenarios zeigte sich jedoch durch einen geringer ausgebildeten Klassifikationsfehler, dass $U(t)$ eine bessere Gruppen-Separierung erreicht als $S(t)$. Aus diesem Grund sehen die nachfolgenden Experimente von einer Verwendung der Metrik $S(t)$ ab.

Bzgl. der Kombinationsmöglichkeiten der Metriken lässt sich sagen, dass insbesondere innerhalb des Testbeds die Kombination von $U(t)$ mit $S(t)$ keinen Informationsgewinn zeigt, da beide Metriken die Gesamtheit der Peers ähnlich bewerten. Die Kombination von $U(t)$ mit der Lokalitätsmetrik $L(t)$ stellt jedoch durchaus einen Gewinn dar: Anhand einer stufenweisen Filterung durch diese Metriken können Peers identifiziert werden, die sich sowohl bzgl. ihres Upload- als auch ihres Lokalitätsverhaltens als günstig erwiesen haben.

Es bedarf detaillierteren Auswertungen, um eine allgemeingültige Aussage für die Identifikationsgüte in der Praxis ableiten zu können. Eine solche Analyse liegt aufgrund ihrer Komplexität jedoch außerhalb der vorliegenden Arbeit. Dennoch liefert die Bewertung der Identifikationsmetriken erste Hinweise darauf, wie sich die Metriken innerhalb des Testbeds verhalten.

8.2 Genereller Einfluss des Bandbreitenmanagement

Dieser Abschnitt bewertet den generellen Einfluss des adaptiven Bandbreitenmanagements (BM). Der Vergleich des Mechanismus erfolgt gegen unterschiedliche Basisfälle. Die Kombination von BM mit BNS und BU ist ebenfalls Gegenstand der nachfolgenden Analyse. Innerhalb jedes Szenarios wird die Supporter-Strategie als sekundärer Distributionspfad genutzt. Die Basisfälle umfassen:

- Basisfall A: Diese Konfiguration sieht vor, dass sich alle Peers im Schwarm regulär verhalten.
- Basisfall B: Diese Konfiguration sieht vor, dass 50% der Gesamtheit aller Peers (pro ISP-Domäne) BNS und BU (vgl. Abschnitt 3.3) verwenden.

Ein Vergleich des BM gegen diese Basisfälle erfolgt anhand von zwei unterschiedlichen Konfigurationen. Zunächst soll der alleinige Einsatz von BM erprobt werden. In diesem Szenario verhalten sich alle Peers bzgl. der Peer-Selektion regulär. Im Anschluss erfolgt die Untersuchung des BM in Kombination mit lokalitätsfördernden Mechanismen (BM+Lok.). Tabelle 8.5 zeigt die Grundkonfiguration für diese Szenarien.

Das Ziel dieser Untersuchung ist, herauszufinden, ob und in welcher Konfiguration sich Vorteile für Content Provider (Reduktion der Server-Last), ISP (Reduktion des Inter-ISP-Volumens) und Benutzer (Aufrechterhaltung einer gewissen Dienstgüte) erkennen lassen. Die Erwartungshaltung hinsichtlich dieser Ziele ist nachfolgend beschrieben:

- Basisfall A zeigt aufgrund der Verwendung der Supporter-Strategie bereits eine gute Dienstgüte durch niedrige Abspielverzögerungen. Das Inter-ISP-Volumen ist relativ hoch, da keiner der Peers lokalitätsfördernde Mechanismen einsetzt. Die Server-Last bewegt sich für die Supporter-Strategie in einem angemessenen Rahmen.
- Basisfall B zeigt durch die Verwendung von BNS und BU eine Reduktion hinsichtlich des Inter-ISP-Volumens. Die Dienstgüte kann sich verschlechtern oder auf demselben Niveau bleiben, wie in Basisfall A. Eine potentielle Verschlechterung ließe sich durch die Präferenz von ISP-lokalen Peers begründen, da u. U. nicht alle Daten zum erforderlichen Zeitpunkt bezogen werden können, oder die akkumulierte Download-Rate aufgrund zu schwacher Uploader in der lokalen ISP-Domäne nicht der Bitrate des Videos entspricht. Die Server-Last ist auf einem ähnlichen Niveau wie in Basisfall A.
- Durch die alleinige Verwendung von BM erhöht sich die akkumulierte Upstream-Kapazität aller Peers. Dadurch sollte sich eine spürbare Verbesserung hinsichtlich Abspielverzögerungen einstellen. Da eine Priorisierung hinsichtlich Verbindungen aus der lokalen ISP-Domäne nicht erfolgt, sollte sich das Inter-ISP-Volumen in etwa auf dem gleichen Niveau befinden, wie in Basisfall A.

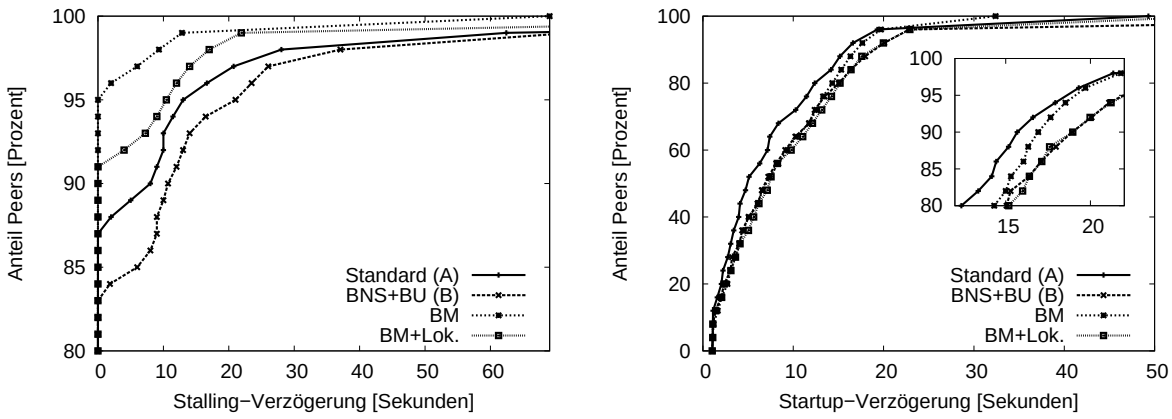
Parameter	Konfiguration
Laufzeit des Szenarios	90 Min.
Videolänge	15 Min.
Bitrate	540 kb/s
Topologie	3 AS, 1 Server-Farm
Verfügbare Server	10 (Supporter)
Server-Kapazität (Upstream)	2048 kb/s
Peer-Zugangsprofile (Upstream)	512 kb/s
Peer-Zugangsprofile (Downstream)	2048 kb/s
Zugangsprofil-Verteilung	Homogen
Ankunftsrate	Poisson-Prozess, 8 Peers/Min.
Peer-Verhalten	Egoistisch (75%), Altruistisch (25%)
Verweilzeit (Egoistisch)	50% der Videolänge im Mittel
Verweilzeit (Altruistisch)	Exponentialverteilt mit Mittelwert Videolänge/2
Peer-Selektion	**Nach Experiment variierend**
BNS-Selektion	90% lokale Verbindungen gewünscht
Einsatz von BM	**Jeder ISP**
Zeit zw. Bandbreitenzuteilungen	5 Min.
Anzahl zu befördender Peers	20% der Peers pro AS
Upstream-Multiplikationsfaktor	2,0
Block-Größe	32 kB

Tabelle 8.5: Grundkonfiguration der Testszenarien zum adaptiven Bandbreitenmanagement

- Die Verwendung von BM in Kombination mit BNS und BU sollte die Vorzüge der einzelnen, zuvor betrachteten Konfigurationen kombinieren:
  - Basisfall A: Gute Dienstgüte aufgrund Supporter-Strategie
  - Basisfall B: Reduktion des Inter-ISP-Volumens
  - Nur BM: Gute Dienstgüte, weniger Server-Last

8.2.1 Einfluss auf Abspielverzögerungen

Abbildung 8.7(i) zeigt die CDFs über der Stalling-Verzögerung aller betrachteten Peers. Anhand den CDFs ist deutlich zu erkennen, dass die BM-Varianten durchweg die beste Performanz erzielen. Speziell bei ausschließlicher Verwendung von BM wächst die Stalling-Verzögerung erst jenseits des 95%-Perzentils und erreicht einen Spitzenwert von 69 Sekunden. Dieser Wert liegt deutlich unter den Spitzenwerten der anderen Mechanismen: 186 Sekunden (Basisfall A), 222 Sekunden (Basisfall B) und 146 Sekunden (BM mit Lokalität). Der Verlauf der CDF für BM mit Lokalität zeigt einen geringfügig schlechteren Verlauf als für reines BM, der sich erst ab dem 95%-Perzentil signifikant vergrößert. Dennoch zeigt die Kombination von BM mit Lokalität im Vergleich zu den Basisfällen ein besseres Verhalten bzgl. der Stalling-Verzögerung. Die Performanz der Basisfälle fällt im Vergleich zu den BM-Varianten ab. Dennoch muss betont werden, dass durch den Einsatz der Supporter-Strategie bereits die Basisfälle eine annehmbare Dienstgüte aufweisen. Basisfall A erreicht bspw. für 95% der Gesamtheit aller Peers eine maximale Stalling-Verzögerung von ca. 13 Sekunden. Gemessen an der Länge des Testvideos ist dies eine relative Verzögerung von 13 Sekunden/900 Sekunden = 1,44%. Dass die Stalling-Verzögerung unter alleiniger Ausnutzung von BNS und BU (Basisfall B) nicht besser werden kann, erscheint logisch. Dass sich selbst in einem verhältnismäßig kleinen Testbed bereits eine signifikante Verschlechterung gegenüber Basisfall A zeigt, ist ein interessantes Ergebnis. Es spricht dafür, dass lokalitätsfördernde Mechanismen zwar das Potential haben, Inter-ISP-Datenverkehr zu reduzieren, sich aber durch diese Reduktion negative Effekte bzgl. der Performanz zeigen können.



(i) Stalling-Verzögerungen

(ii) Startup-Verzögerungen

Abbildung 8.7: CDF zu den Stalling-Verzögerungen der Peers (i) und ihren Startup-Verzögerungen (ii)

Die Evaluation der Supporter-Strategie hat bereits gezeigt, dass sich die Startup-Verzögerung weitestgehend stabil über variierende Ankunftsraten und bzgl. unterschiedlicher Server-Strategien verhält (vgl. Abschnitt 7.1.3). Diese Tatsache lässt sich anhand den CDFs in Abbildung 8.7(ii) ebenfalls beobachten. Speziell Basisfall B und die BM-Varianten zeigen bis hin zum 85%-Perzentil ein nahezu identisches Verhalten. Die Startup-Verzögerung für Basisfall A ist erst ab dem 98%-Perzentil höher als unter Verwendung von BM. Die Abweichungen zwischen den Mechanismen sind marginal und zeigen keine besondere Signifikanz. Dass Basisfall A (Standard) am besten abschneidet, ist – in Analogie zur Evaluation der Supporter-Strategie – aus Sicht des Systems nicht zu erklären.

Abspielverzögerungen				
Verzögerung	**Basisfall A**	**Basisfall B**	**Nur BM**	**BM mit Lokalität**
Stalling (50%)	0,0 s ± 0,0 s	0,0 s ± 0,0 s	0,0 s ± 0,0 s	0,0 s ± 0,0 s
Stalling (95%)	13,13 s ± 2,68 s	19,68 s ± 2,31 s	1,24 s ± 1,57 s	9,63 s ± 1,46 s
Startup (50%)	4,84 s ± 0,35 s	7,45 s ± 1,12 s	6,34 s ± 1,08 s	7,24 s ± 0,15 s
Startup (95%)	18,6 s ± 0,16 s	30,14 s ± 12,35 s	18,48 s ± 0,66 s	21,96 s ± 0,21 s
Gesamt (50%)	6,29 s ± 0,25 s	9,5 s ± 1,52 s	7,03 s ± 0,75 s	8,19 s ± 0,06 s
Gesamt (95%)	23,84 s ± 0,77 s	41,26 s ± 16,49 s	18,96 s ± 0,63 s	24,62 s ± 1,93 s

Tabelle 8.6: Aufstellung der absoluten Abspielverzögerungen (Mittelwert ± Standardabweichung) nach Typ der Verzögerung und für jede Konfiguration

Tabelle 8.6 zeigt die absoluten Verzögerungen hinsichtlich Stalling-, Startup- und Gesamtverzögerung. Bei der Auswertung der Ergebnisse zeigte sich, dass die Startup-Verzögerung einen erheblichen Anteil an der Gesamtverzögerung ausmacht. Dies fällt insbesondere bei der Analyse der BM-Varianten auf. Sie zeigen eine konstant niedrige Stalling-Verzögerung im Vergleich zu den anderen Verfahren, so dass die Gesamtverzögerung fast ausschließlich durch die Startup-Verzögerung zustande kommt (vgl. Tabelle 8.6). Bis auf die Ausnahme der Startup-Verzögerung bei Basisfall B sind die Standardabweichungen relativ gering. Dies lässt den Schluss zu, dass die Ergebnisse die tatsächliche Performanz der Mechanismen gut wiedergeben. Da die Unterschiede bzgl. der Startup-Verzögerung nicht signifikant sind und ihr Einfluss auf die Gesamtverzögerung sehr groß ist, lässt sich die Leistung der Mechanismen am besten anhand der Stalling-Verzögerung zeigen. Die absoluten Unterschiede in der Stalling-Verzögerung sind zwischen den Basisfällen und den BM-Varianten auf dem 95%-Perzentil durchaus signifikant. So erreicht der alleinige Einsatz von BM im Vergleich zu Basisfall A eine Reduktion um etwa den Faktor 10,6 (13,13 Sekunden/1,24 Sekunden). Ein direkter Vergleich von Basisfall B und BM mit BNS und BU ist ebenfalls möglich. Der Reduktionsfaktor durch den Einsatz von BM mit BNS und BU liegt bei etwa 2 (19,68 Sekunden/9,63 Sekunden) und ist damit nicht ganz so hoch wie im vorhergehenden Fall. Dennoch zei-

gen diese Ergebnisse, dass eine spürbare Verbesserung durch den Einsatz von BM (in allen Varianten) möglich ist.

8.2.2 Einfluss auf die Server-Last

Tabelle 8.7 zeigt den potentiellen Gewinn bzgl. der Server-Last bei Verwendung des adaptiven Bandbreitenmanagements. Der Vergleichswert von Basisfall A erscheint vernünftig. Geht man von der Gesamtkapazität über die Testdauer aus, dann haben die 10 Supporter-Server eine gesamte, mittlere Auslastung von $5{,}47\ \text{GB}/(10 \cdot 90\ \text{min} \cdot 256\ \text{kB/s}) = 0{,}415$. Gemittelt betrachtet würde man also mit 5 der 10 Supporter Servern auskommen. Das sind durchaus Werte, die sich mit den Ergebnissen der Supporter-Evaluation in ähnlichen Szenarien decken.

Konfiguration	**Server-Last**	
	Absolutwert	Einsparung
Basisfall A	5,47 GB ± 0,20 GB	-
Basisfall B	5,37 GB ± 0,16 GB	1,83%
Nur BM	3,43 GB ± 0,19 GB	37,36%
BM mit Lokalität	3,88 GB ± 0,16 GB	28,98%

Tabelle 8.7: Absolute Server-Last (Mittelwert ± Standardabweichung) und prozentuale Angabe der Einsparung bzgl. der Server-Last im Vgl. zu Basisfall A

Basisfall B zeigt eine marginale Reduktion der Server-Last im Vergleich zu Basisfall A. Die Notwendigkeit für den Einsatz der Server scheint durchaus gegeben zu sein, denn wie die vorhergehende Betrachtung gezeigt hat, verschlechtern sich die Abspielverzögerungen bei dieser Konfiguration. Dass die Server-Last nicht ansteigt, liegt evtl. daran, dass Peers sich mit einem zugeteilten Supporter aufgrund des lokalitätsfördernden Mechanismus nicht schnell genug verbinden.

Die BM-Varianten zeigen eine durchweg gute Reduktion der Server-Last. Durch die Ausstattung ausgewählter Peers mit einer höheren Upstream-Kapazität erreicht das System eine bessere Effizienz hinsichtlich der Verteilung von Daten. Das sorgt dafür, dass eine erhöhte Anzahl von Peers versorgt bleibt. Die Anzahl unterversorgter Peers ist geringer als in den Basisfällen A und B. Dies reduziert gleichzeitig die Häufigkeit einer Server-seitigen Unterstützung, was sich letzten Endes durch eine deutlich niedrigere Server-Last in den Ergebnissen präsentiert. Beim Vergleich mit den Basisfällen muss beachtet werden, dass die Supporter-Strategie bereits einen guten Kompromiss zwischen Performanz des Schwarms und Server-Last erzielt (vgl. Ergebnisse aus Kapitel 7). Dass die Server-Last durch den Einsatz von BM um weitere 37,36% reduziert werden kann, zeigt, dass der potentielle Gewinn durch eine Kombination dieser Mechanismen durchaus signifikant ist. Bei der Verwendung von BM mit lokalitätsfördernden Mechanismen ist der Gewinn in der Reduktion der Server-Last nicht ganz so ausprägt, wie bei der ausschließlichen Verwendung von BM. Die Performanz des Schwarms verhält sich in diesem Fall analog zu Basisfall B. Der Unterschied ist, dass die lokale Beförderung ausgewählter Peers anderen Peers aus derselben ISP-Domäne zu Gute kommt. Dies wirkt der Verschlechterung des Versorgungszustands ISP-lokaler Peers entgegen, was sich nicht nur positiv auf die Server-Last auswirkt, sondern auch auf Abspielverzögerungen (vgl. Abschnitt 8.2.1).

8.2.3 Einfluss auf das Transfervolumen nach Verkehrsklasse

Tabelle 8.8 zeigt die Aufschlüsselung des Datenvolumens nach Intra-ISP-, Inter-ISP-Download- und Inter-ISP-Upload-Volumen für jede untersuchte Konfiguration. Wie zu erwarten, verursacht Basisfall A das höchste Inter-ISP-Volumen, sowohl bzgl. eingehendem (Download) als auch ausgehendem (Upload) Datenverkehr. Im Vergleich dazu erwirkt Basisfall B durch den Einsatz von BNS und BU eine signifikante Reduktion bzgl. des Inter-ISP-Datenverkehrs. Diese Reduktion ist jedoch mit Nachteilen bzgl. anderer Ausgabemetriken (vgl. Abschnitt 8.2.1) behaftet.

Konfiguration	Intra-ISP	Inter-ISP-Download		Inter-ISP-Upload	
	Absolutwert	Absolutwert	Einsparung	Absolutwert	Einsparung
Basisfall A	13,21 GB	18,46 GB	-	13,19 GB	-
Basisfall B	21,84 GB	11,20 GB	39,33%	5,99 GB	54,59%
Nur BM	12,72 GB	16,21 GB	12,19%	13,01 GB	1,36%
BM mit Lokalität	22,24 GB	9,03 GB	51,08%	5,41 GB	58,98%

Tabelle 8.8: Absolutes Datenvolumen (Mittelwert) und prozentuale Angabe der Einsparung bzgl. des Datenvolumens im Vgl. zu Basisfall A

Der alleinige Einsatz von BM verspricht nur eine geringfügige Verbesserung, die sich hauptsächlich auf den Download über Inter-ISP-Verbindungen auswirkt. Die Reduktion ist durch die erhöhte Upstream-Kapazität aller Peers innerhalb einer ISP-Domäne erklärbar:

1. Das G2G-Protokoll bewertet einen benachbarten Peer anhand seines Weiterleitungsverhaltens. Ist ein solcher benachbarter Peer durch das BM mit einer höheren Upstream-Kapazität ausgestattet worden, so kann er potentiell mehr Daten anderen Peers zur Verfügung stellen und sich als besserer Weiterleiter im Vergleich zu regulären Peers etablieren. Das heißt, dass auch ohne lokalitätsfördernde Mechanismen ein Peer mit höherer Bandbreite attraktiver erscheinen kann, als ein regulärer Peer. Die Analyse der Identifikationsmetriken $U(t)$ und $S(t)$ hinsichtlich der Heterogenität (vgl. Abschnitte 8.1.1 und 8.1.2) bestätigen die präferentielle Wahl von Peers mit einem höheren Zugangsprofil. Die Entscheidung, ob ein Peer mit einem regulären oder beförderten Peer Daten austauschen soll, kann somit zu Gunsten des beförderten Peers entschieden werden.
2. Eine andere Erklärung ist in dem Versorgungszustand der Peers zu finden. Durch die Erhöhung der Upstream-Kapazität ausgewählter Peers können benachbarte Peers mit einer höheren Transferrate Daten beziehen. Dadurch verbessert sich potentiell der Versorgungszustand der Peers, die Daten mit einem beförderten Peer austauschen. Dies reduziert die Server-Last auf den Supportern und somit auch den Inter-ISP-Datenverkehr über Verbindungen zur Server-Farm.

Der Einsatz von BM in Kombination mit BNS und BU zeigt das beste Verhalten unter allen betrachteten Konfigurationen. Basisfall B hat bereits gezeigt, dass durch BNS und BU eine signifikante Reduktion hinsichtlich des Inter-ISP-Volumens erreicht werden kann. Die Kombination mit dem BM steigert die Reduktion sowohl des eingehenden als auch ausgehenden Inter-ISP-Volumens um zusätzliche 12% resp. 4%.

8.2.4 Diskussion

Die Ergebnisse zeigen, dass die einführend definierte Erwartungshaltung im Allgemeinen erfüllt wurde. Lediglich die Startup-Verzögerung zeigt ein nicht erklärbares Verhalten. Jedoch wurde dies bereits bei der Evaluation der Supporter-Strategie festgestellt und ist im Kontext der Gesamtheit der Ergebnisse vernachlässigbar. Die einzelnen Ergebnisse der vorangegangenen Betrachtung sind zusammengefasst:

1. Durch den Einsatz von BM (mit oder ohne Lokalität) erreicht man eine spürbare Verbesserung der Abspielverzögerungen, insbesondere bzgl. Stalling- und Gesamtverzögerung.
2. Die Vergleichsszenarien nutzen bereits den Supporter-Server und erreichen somit einen guten Kompromiss zwischen Leistung und Server-Last. Setzt man BM in Kombination mit der Supporter-Strategie ein, so zeichnet sich eine zusätzliche Reduktion der Server-Last um bis zu 37% ab, ohne dabei Dienstgüte einzubüßen.
3. Reduktionen hinsichtlich des Inter-ISP-Datenverkehrs zeigen sich hauptsächlich durch den Einsatz von BM mit lokalitätsfördernden Mechanismen.

Betrachtet man diese Ergebnisse in ihrer Gesamtheit, so lässt sich feststellen, dass die Ziele von Benutzer, Content Provider und ISP durch den Einsatz der BM-Varianten erfüllt werden können. Insbesondere die Kombination von BM mit lokalitätsfördernden Mechanismen erscheinen interessant, da hierbei der Gewinn für den ISP prägnanter ist, als durch den alleinigen Einsatz von BM. Diesen Gewinn erreicht das BM, ohne dabei Dienstgüteansprüche

der Benutzer zu vernachlässigen. Eher im Gegenteil: Der Einsatz von BM hat gezeigt, dass insbesondere Stalling-Verzögerungen signifikant reduziert werden können.

8.3 Einfluss des Bandbreitenmanagements im Falle eines Early Adopters

Im vorhergehenden Abschnitt hat man gesehen, dass der Einsatz von BM in allen ISP-Domänen den Gesamtzustand des Systems bzgl. Abspielverzögerungen, Server-Last und Inter-ISP-Datenverkehr verbessern kann. Ziel dieses Abschnitts ist nun die Untersuchung, wie stark sich diese Effekte ausprägen, wenn nur ein einziger ISP das BM umsetzt. Dieser ISP wird im Rahmen der folgenden Betrachtung als Early Adopter[1] adressiert. Der Vergleich hinsichtlich der Ausgabemetriken erfolgt nicht direkt von Mechanismus zu Mechanismus, sondern zwischen den ISPs untereinander. Die Konfiguration des Szenarios orientiert sich an der Grundkonfiguration aus Tabelle 8.5, mit den Modifikationen, die in Tabelle 8.9 zusätzlich beschrieben sind. Zu erwarten ist:

Parameter	**Konfiguration**
Einsatz von BM	Nur 1 ISP (Early Adopter), restliche ISPs nicht
Peer-Selektion	50% der Peers des Early Adopters, Peers anderer ISPs nicht

Tabelle 8.9: Abweichungen zur Grundkonfiguration (vgl. Tabelle 8.5) des Early-Adopter-Testszenarios

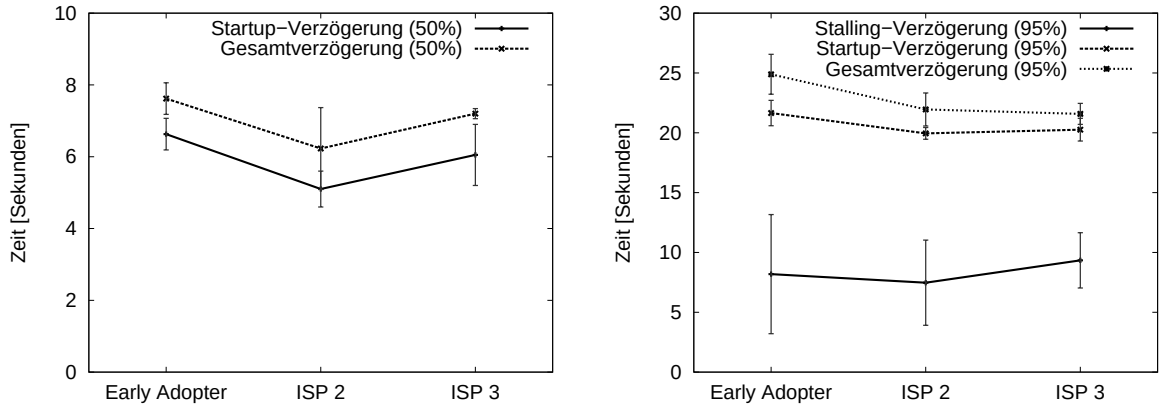
- Die bisherigen Ergebnisse legen Grund zur Vermutung nahe, dass sich die Abspielverzögerungen größtenteils ähneln sollten. Zum Einen, weil die Supporter-Strategie unterversorgte Peers mit Daten versorgen kann, zum Anderen weil der Einfluss von BM geringer ausfallen dürfte, wenn der Mechanismus nur von einem ISP umgesetzt wird.
- Die Server-Last sollte sich reduzieren, da Peers aus der ISP-Domäne des Early Adopters vermutlich weniger häufig in den Unterversorgt-Zustand gelangen als Peers anderer ISP-Domänen. Es ist jedoch zu erwarten, dass die Reduktion der Server-Last nicht so deutlich ausfällt, wie in den vorangegangenen Experimenten.
- Der Einfluss von BM im Falle eines Early Adopters sollte sich insbesondere auf das gesamte Inter-ISP-Volumen auswirken. Für den Early Adopter ist durch die Verwendung lokalitätsfördernder Mechanismen eine signifikante Reduktion zu erwarten. Die Dienstgüte auf Seiten der Peers sollte hierbei nicht in Mitleidenschaft gezogen werden. Für die ISPs, die BM nicht einsetzen, sollten sich keine signifikanten Änderungen hinsichtlich des Datenvolumens per Verkehrsklasse ergeben. Das gesamte Inter-ISP-Volumen dieser ISPs sollte deutlich höher ausfallen, als im Falle des Early Adopters.

8.3.1 Vergleich der Abspielverzögerungen

Abbildung 8.8(i) zeigt den Median bzgl. der Abspielverzögerungen. Die Betrachtung der Stalling-Verzögerung ist in dieser Grafik nicht enthalten, da sie für alle Peers der unterschiedlichen ISP-Domänen konstant bei 0 Sekunden gelegen hat. Die Abbildung zeigt, dass der Unterschied zwischen Startup- und Gesamtverzögerung für Peers jedes ISPs auf einem fast identischen Niveau liegt. Der Early Adopter zeigt sowohl bzgl. der Startup- als auch der Gesamtverzögerung und leicht schlechteres Verhalten. Der Unterschied liegt allerdings in einer Größenordnung von ca. 2 Sekunden und ist daher insbesondere bzgl. der Startup-Verzögerung nicht wirklich signifikant.

Abbildung 8.8(ii) zeigt die Abspielverzögerungen auf dem 95%-Perzentil. Es ist zu erkennen, dass sich die Peers in jeder ISP-Domäne ungefähr gleich verhalten. Der Early Adopter zeigt ein bzgl. der Startup- und Gesamtverzögerung etwas schlechteres Verhalten. Die absoluten Unterschiede zu anderen ISPs bewegen sich allerdings wieder in einem Bereich von wenigen Sekunden. Da die Gesamtverzögerung größtenteils durch die Startup-Verzögerung zustande kommt, sind diese Unterschiede nicht signifikant. Stalling-Verzögerungen der Peers zeigen zwischen den ISPs kaum Unterschiede. Speziell für den Early Adopter und ISP 2 fallen die Standardabweichungen bzgl. Stalling besonders hoch aus, so dass man von einer gewissen Messunschärfe ausgehen kann, die den Daten zugrunde liegt.

[1] Der Begriff Early Adopter bezeichnet in diesem Kontext einen ISP, der den neuen Mechanismus frühzeitig umsetzt, während andere ISPs darauf verzichten.



(i) Median

(ii) 95%-Perzentil

Abbildung 8.8: Abspielverzögerungen auf dem Median (i) und dem 95%-Perzentil (ii). Der Median der Stalling-Verzögerung liegt für alle ISPs 0,0 Sekunden und entfällt daher in Abbildung (i).

8.3.2 Einfluss auf die Server-Last

Tabelle 8.10 zeigt die absoluten Werte der Server-Last zusammen mit der relativen Einsparung. Die Server-Last reduziert sich auch im Falle des Early-Adopter Szenarios. Die relative Einsparung beträgt allerdings nur 9,32%. Dieser Wert misst nur noch 1/3 der Einsparung (28,98%), die der Einsatz von BM bei allen ISPs erreicht hat. Das stimmt relativ genau mit der Erwartung überein. Die akkumulierte Upstream-Kapazität aller Peers in der Domäne des Early Adopters ist höher, als in den anderen ISP-Domänen. Durch die zusätzliche Verwendung von BNS und BU verbessert sich dadurch hauptsächlich der Versorgungszustand von Peers in der Domäne des Early Adopters. Peers, die den anderen ISP-Domänen zugehörig sind, zeigen daher häufiger einen unterversorgten Zustand und benötigen Unterstützung eines Servers. Da die Supporter Server Peers aus allen ISP-Domänen mit Daten versorgen, verschlechtert sich daher global betrachtet das Potential für eine Reduktion der Server-Last. Der Unterschied in der relativen Einsprung ist daher auf das Verhältnis von ISPs, die BM einsetzen und der Gesamtzahl an ISPs zurückzuführen (1/3). Die Ergebnisse bestätigen dies, denn $9{,}32\%/28{,}98\% = 0{,}32 \approx 1/3$.

Konfiguration	**Server-Last**	
	Absolutwert	Einsparung
Basisfall A	5,47 GB ± 0,20 GB	-
BM mit Lokalität (Alle ISPs)	3,88 GB ± 0,16 GB	28,98%
BM mit Lokalität (Early Adopter)	4,96 GB ± 0,33 GB	9,32%

Tabelle 8.10: Absolute Server-Last (Mittelwert ± Standardabweichung) und prozentuale Angabe der Einsparung bzgl. der Server-Last im Vgl. zu Basisfall A und dem ausschließlichen Einsatz von BM (vgl. Abschnitt 8.2)

8.3.3 Vergleich des Transfervolumens nach Verkehrsklasse

Die Peers verteilen sich gleichmäßig auf die drei ISP-Domänen, so dass ein direkter Vergleich zwischen ISPs hinsichtlich des Datenverkehrsvolumens möglich ist. Tabelle 8.11 zeigt den Mittelwert des absoluten Datenvolumens für jede Verkehrsklasse. Durch den Einsatz des BM erreicht der Early Adopter im Vergleich zu den anderen ISPs eine mittlere Einsparung von 37,62% bzgl. Inter-ISP-Download und 32,22% bzgl. Inter-ISP-Upload.

Ein Vergleich des Inter-ISP-Volumens der ISPs, die kein BM einsetzen, ist gegen Basisfall A möglich. Die Peer-Verteilung in Basisfall A (vgl. Abschnitt 8.2.3) ist ebenfalls gleichmäßig, so dass sich das Datenvolumen in etwa entsprechend dieser Verteilung aufteilt. Das bedeutet, dass ein ISP in Basisfall A in etwa ein Inter-ISP-Download-

Konfiguration	Intra-ISP	Inter-ISP-Download	Inter-ISP-Upload
Early Adopter	6,85 GB	3,61 GB	2,64 GB
ISP 2 (kein BM)	5,40 GB	5,94 GB	3,85 GB
ISP 3 (kein BM)	5,45 GB	5,61 GB	3,94 GB

Tabelle 8.11: Absolutes Datenvolumen (Mittelwert) und prozentualer Vergleich über alle ISPs

Volumen von 6,15 GB und ein Inter-ISP-Upload-Volumen von 4,40 GB zeigt. Die Abweichung von diesen mittleren Werten ist relativ gering, so dass die ISPs, die kein BM einsetzen, nicht von dem Early Adopter profitieren.

8.3.4 Diskussion

Die Ergebnisse bestätigen die eingangs erwähnte Erwartung. In einem Early-Adopter Szenario profitiert in erster Linie nur der ISP, der BM einsetzt. Die Abspielverzögerungen auf Seiten der Benutzer zeigen sich weitestgehend stabil, was sicherlich auch der Funktionsweise der Supporter-Strategie zuzuschreiben ist. Die Erhöhung der Upstream-Kapazität in der Domäne des Early Adopters kompensiert zusätzliche Einbußen, die potentiell durch den Einsatz von lokalitätsfördernden Mechanismen hinsichtlich Abspielverzögerungen in Kauf zu nehmen wären. Eine Verbesserung hinsichtlich der Server-Last kann erzielt werden, jedoch fällt diese deutlich geringer aus. Ferner ist zu erwarten, dass in einem Early-Adopter Szenario mit einer größeren Anzahl von ISPs die Reduktion der Server-Last noch geringer ausfallen würde, als es die in Abschnitt 8.3.2 diskutierten Ergebnisse zeigen. Zusammengefasst lässt sich sagen, dass eine Gewinnsituation durchaus zu verzeichnen ist, diese jedoch im Vergleich zu dem Einsatz von BM bei allen ISPs (vgl. Abschnitt 8.2.1) deutlich geringer ausfällt.

9 Diskussion

Dieses Kapitel liefert einen zusammenfassenden Überblick über die Ergebnisse der vorliegenden Arbeit und diskutiert diese kritisch. Die Realisierung des adaptiven Bandbreitenmanagements und der Supporter-Strategie sind als Proof of Concept zu betrachten. Aus diesem Grund unterliegen die zugehörigen Implementierungen einigen Restriktionen, die einen praxistauglichen Einsatz erschweren. Die Diskussion zeigt diese Punkte auf und gibt Lösungshinweise, um insbesondere die Supporter-Strategie für den praktischen Einsatz effizienter zu realisieren.

9.1 Zusammenfassung

Bisherige Ansätze, die sich der Reduktion von Inter-ISP-Datenverkehr widmen, lösen dieses Problem nur einseitig. Ein ISP kann durch die Verwendung bereits existierender Mechanismen durchaus einen gewissen Nutzen erzielen. Jedoch berücksichtigen diese Mechanismen nicht die Interessen eines Content Providers oder Benutzers. Die eingangs gestellte Forschungsfrage nahm sich diesem Problem an und thematisierte die Reduktion des Inter-ISP-Datenverkehrs im Hinblick auf eine zusätzliche Reduktion der Last auf den Servern eines Content Providers und der Wahrung einer hinreichenden Dienstgüte auf Seiten des Benutzers. Der Ansatz wurde im Rahmen einer hybriden Systemarchitektur, die auf einer P2P-basierten Verteilstrategie neben einer Server-Komponente als Ausweichlösung basiert, motiviert. Die Arbeit griff für die Realisierung der Server-Komponente auf die Supporter-Strategie zurück, die im Rahmen von adaptiven Server-Allokationsstrategien entwickelt wurde. Die Supporter-Strategie wurde im Rahmen der vorliegenden Arbeit prototypisch implementiert und evaluiert. Die Ergebnisse dieser Evaluation zeigen, dass die Supporter-Strategie eine Kompromisslösung zwischen Server-Last und Leistung erzielt. Die Vorzüge dieser Strategie sind verschiedener Natur. Zum Einen erreicht ein Content Provider eine erhebliche Kosteneinsparung durch die Reduktion von Server-Last. Zum Anderen erlaubt es die Supporter-Strategie, adaptiv auf die Bedürfnisse des Schwarms zu reagieren und Ressourcen zur Verfügung zu stellen, sollten sich Verschlechterungen hinsichtlich der Dienstgüte auf Seiten der Benutzer ergeben. Die quantitative Studie in Kapitel 7 zeigt sogar, dass sich die Dienstgüte in allen betrachteten Szenarien gegenüber den Basisfällen (mehrere statische Server) verbessert. Insbesondere in Szenarien, in denen ein Content Provider den Peer-Bedarf nicht kennt, oder die Auslastung der Server über die Zeit hinweg starken Schwankungen unterzogen ist, bietet sich der Einsatz der Supporter-Strategie an.

Dieses durchweg positive Resultat nutzte die vorliegende Arbeit, um den Einsatz der Supporter-Strategie in der hybriden Systemarchitektur zu rechtfertigen. Um Inter-ISP-Datenverkehr zu reduzieren, wurde der Mechanismus des adaptiven Bandbreitenmanagements eingeführt. Die Evaluation dieses Ansatzes hat gezeigt, dass bei einer Anwendung des adaptiven Bandbreitenmanagements innerhalb jeder ISP-Domäne ein signifikanter Gewinn erzielt werden kann. Insbesondere die Kombination mit lokalitätsfördernden Mechanismen hat gezeigt, dass jede der beteiligten Interessengruppen durch den vorgestellten Ansatz profitiert:

- Zusätzliche Verbesserung der Dienstgüte (Benutzer)
- Zusätzliche Reduktion der Server-Last (Content Provider)
- Signifikante Reduktion von Inter-ISP-Datenverkehr, bis zu ca. 50% (ISP)

Die Ergebnisse haben jedoch auch gezeigt, dass durch den reinen Einsatz von lokalitätsfördernden Mechanismen eine Reduktion der Dienstgüte zu erwarten ist. Durch Einsatz des adaptiven Bandbreitenmanagements und insbesondere auch der Supporter-Strategie ist es möglich, diese negativen Effekte gänzlich zu kompensieren. Die Supporter-Strategie zeigt ihren positiven Einfluss demnach auch in diesem Szenario.

Die Arbeit griff nachfolgend die Frage auf, ob sich die positiven Effekte durch das adaptive Bandbreitenmanagement auch dann einstellen, wenn nur ein einzelner ISP den Mechanismus umsetzt. Die Ergebnisse bestätigen, dass ein gewisser Gewinn v. a. auf Seiten des ISPs zu verzeichnen ist. Jedoch zeigte sich auch, dass die Reduktion der Server-Last mit steigender Anzahl an ISPs, die den Mechanismus nicht einsetzen, sinkt. Die Performanz auf Seiten des Benutzers zeigte kaum Unterschiede. Vor dem Hintergrund, dass die Supporter-Strategie bereits einen guten Kompromiss zwischen Server-Last und Performanz erzielt, ist dies dennoch ein günstiges Ergebnis, da durch die Kombination dieser Mechanismen letzten Endes alle Interessengruppen einen gewissen Gewinn erzielen.

9.2 Zukünftige Arbeiten

In Anlehnung an diese Arbeit könnten sich künftige Studien mit den nachfolgend betrachteten Fragestellungen und Problemen beschäftigen.

9.2.1 Praxistauglichkeit der Implementierung der Supporter-Strategie

Die Evaluation der Supporter-Strategie hat gezeigt, dass der Ansatz für die Verwendung in einer hybriden Systemarchitektur durchaus geeignet ist. Die Implementierung der Supporter-Strategie ist dennoch als *Proof of Concept* zu betrachten, da sie noch einige Schwächen zeigt, die einen praktischen Einsatz erschweren:

- **Signalisierung der Zuweisung von Überwachungskomponente zu Peer**: Der zentrale Index-Server eines BT-ähnlichen Systems liefert jedem anfragenden Peer eine vollständige Liste der Supporter Server, die sich derzeit im Schwarm befinden. Dieser Schritt ist notwendig, da die Überwachungskomponente a priori nicht weiß, welchem Supporter ein unterversorgter Peer zugewiesen wird. Typischerweise ist die Nachbar-Liste des Index-Servers jedoch auf wenige Elemente beschränkt, um zum Einen die Bandbreitenauslastung des Index-Servers und zum Anderen den Aufwand auf Peer-Seite alle möglichen Tauschpartner zu kontaktieren, gering zu halten. Sofern sich nur wenige Supporter Server im Schwarm befinden, ist dies ein durchaus praktikabler Ansatz. Dieser Ansatz skaliert jedoch nicht, da mit wachsender Nachfrage zusätzliche Supporter Server benötigt werden, um die Peers des Schwarms hinreichend mit Daten zu versorgen. Dies wiederum erfordert mehr Listenplätze in der Antwort des Index-Servers, was gleichzeitig die Fülle an Kontaktinformationen regulärer Peers limitiert. Das kann die Performanz des Overlays reduzieren, da unnötig viel Unterstützung angefragt wird, schlicht und ergreifend, weil ein Peer nicht genug andere reguläre Peers kennt, mit denen er Daten austauschen kann. Eine Lösung dieses Problems kann bspw. über die Bereitstellung einer Schnittstelle auf Seiten des Peers erfolgen. Über diese Schnittstelle kann die Überwachungskomponente die Zuteilung zu einem Supporter Server an den betreffenden Peer kommunizieren. Die Notwendigkeit, alle Supporter Server in der Tracker-Antwort zu kodieren, wäre nicht mehr gegeben.
  Für einen Peer arbeitet der Mechanismus weitgehend transparent (abgesehen von der Signalisierung zwischen dem Peer und der Überwachungskomponente), sprich, der Peer kann nicht zuordnen, welcher Nachbar ein Supporter oder ein regulärer Peer ist. Der Peer versucht, Verbindungen zu potentiell guten Nachbarn aufrecht zu erhalten. Da der Supporter für alle Peers, die er nicht unterstützen soll, die Verbindungen auf den Blockiert-Zustand setzt, sieht ein anfragender Peer zunächst keinen Vorteil darin, eine solche Verbindung überhaupt aufrecht zu erhalten. Im Zweifelsfall muss der Peer zunächst alle Nachbarn erneut kontaktieren, um festzustellen, dass er nun Unterstützung durch einen Supporter Server erfährt. Abhilfe könnte hier ebenfalls eine direkte Signalisierung von Überwachungskomponente zu unterversorgtem Peer nach Zuweisung zu einem Supporter schaffen, um den unterversorgten Peer über den neuen Zustand zu informieren.
- **Mangelnder Horizont bzgl. des Versorgungszustands eines Peers**: Peers, die Unterstützung benötigen, verfügen über einen mangelnden Horizont, der nicht über den Versorgungszeitraum hinaus geht. Erhält ein Peer Unterstützung, so füllt sich sein Wiedergabepuffer und er erkennt, dass er mit einer konstanten und ausreichenden Datenrate die Wiedergabe aufrecht erhalten kann. Da der Supporter Mechanismus für den Peer transparent ist, geht der Peer nicht von einer kurzweiligen Erscheinung aus, sondern realisiert, dass die Unterstützung des Supporters zu gegebenem Zeitpunkt nicht mehr nötig ist. Der Peer signalisiert dies der Überwachungskomponente, worauf hin diese die Unterstützung aufhebt. Findet dieser Peer keine regulären Tauschpartner, um neue Chunks zu erhalten, so leert sich der Wiedergabepuffer und er benötigt erneut Unterstützung. In Streaming-Systemen ist dies durchaus ein Problem, da der Peer Chunks in konsekutiver Reihenfolge anfordert. Gibt es keine oder nur wenige andere reguläre Peers, die *vor* diesem Peer dem Schwarm beigetreten sind, so sinkt die Wahrscheinlichkeit, dass die Bedürfnisse des Peers durch den Schwarm aufgefangen werden können. Der Peer gerät erneut in den unterversorgten Zustand, erhält jetzt aber nicht notwendigerweise die Zuweisung zu einem Supporter. Dieses Problem besteht grundsätzlich, wenn sich zu einem gegebenen Zeitpunkt mehr Peers im Schwarm befinden, als Supporter Upload Slots (global) verfügbar sind. Insbesondere in Last-Situationen, in denen potentiell sehr viele Peers um die Zuweisung zu einem Supporter Server konkurrieren, kann sich die Performanz des Schwarms verschlechtern.
- **Limitierung der Transferrate zu unterstützender Peers**: Die prototypische Implementierung sieht nicht vor, die Transferrate von einem Supporter zu einzelnen Peers zu limitieren. Versorgt ein Supporter Server

10 Peers, dann teilen sich diese 10 Peers die Upstream-Kapazität des Supporters. Versorgt der Supporter Server dagegen nur einen Peer, dann erhält dieser die komplette Upstream-Kapazität. Dies ist jedoch nicht notwendig. Der Supporter kennt den Schwarm und die Anforderungen, die ein Peer an die Mindestdatenrate stellt und kann bspw. die Upload-Rate auf einer Verbindung auf die Bitrate begrenzen. Es ist zu erwarten, dass dies die Dienstgüte nicht einschränkt, dafür aber die Reduktion der Server-Last noch prägnanter ausfällt, als es die quantitativen Studien in Kapitel 7 zeigen.

- **Immunität gegenüber Peer-Attacken**: Der Supporter ist derzeit nicht immun gegen Attacken einzelner Peers. Eine Attacke könnte in Form fortwährender Unterversorgt-Nachrichten erfolgen, mit denen sich ein Peer die Unterstützung eines Supporter Servers zusichern kann.

Die Behebung dieser Schwächen und erneute Evaluierung der Supporter-Strategie in der Praxis könnte Gegenstand künftiger Arbeiten sein. Als Vergleichsbasis hierfür können die Ergebnisse der vorliegenden Arbeit dienen.

9.2.2 Adaptives Bandbreitenmanagement

Die Arbeit hat gezeigt, dass das adaptive Bandbreitenmanagement einen grundsätzlich positiven Effekt auf die Performanz eines hybriden P2P-Systems hat. Einige Fragen blieben jedoch aufgrund des limitierten Rahmens einer Masterarbeit unbeantwortet. Die nachfolgende Auflistung hebt besonders wichtige Fragestellungen hervor, erhebt aber keinen Anspruch auf Vollständigkeit:

- **Einfluss von Parametervariationen und -kombinationen**: Die vorliegende Arbeit konnte aufgrund der empirischen Erprobung des adaptiven Bandbreitenmanagements den Einfluss bei Variation der Parameter nicht analysieren. Hierfür eignet sich eine Studie, die durch Simulation erfolgt und die Vor- bzw. Nachteile variierender Parameter und Parameterkombinationen testet. Die Ergebnisse einer solchen Studie würden Anhaltspunkte für eine annähernd optimale Einstellung des Mechanismus in der Praxis liefern. Eine Parameterstudie ist aus diesem Grund insbesondere für ISPs interessant, die den Einsatz des adaptiven Bandbreitenmanagements in Erwägung ziehen, sich über die Auswirkungen unterschiedlicher Einstellungen jedoch nicht im Klaren sind.
- **Feldstudie in Kooperation mit ISP**: Insbesondere der Fall des Early Adopters hat gezeigt, dass die Auswirkungen bei einer großen Anzahl von ISPs speziell im Hinblick auf die Reduktion der Server-Last marginal sind. Ob es in diesem Szenario Abhängigkeiten bzgl. anderer Ausgabemetriken gibt, konnte ihm Rahmen der Evaluation dieser Arbeit nicht festgestellt werden. Das heißt jedoch nicht, dass etwaige Einflüsse in einem größer angelegten, realen System nicht existieren. Eine Feldstudie in Kooperation mit einem ISP könnte die Vorzüge für einen Early Adopter detaillierter herausstellen. Dass eine solche Kooperation möglich und sinnvoll ist, zeigt eine Studie von Comcast [49], in der das P4P-System [47] unter realen Bedingungen erprobt wurde.
- **Identifikationsgüte in stark heterogenen Systemen**: Der Mechanismus ist sehr stark von der Güte der Identifikationsmetriken abhängig. Die vorliegende Arbeit liefert erste Anhaltspunkte über die Performanz der vorgestellten Metriken. Insbesondere die Analyse der Kombinierbarkeit ist in dem limitierten Rahmen der vorliegenden Arbeit schwierig, zumal der Mechanismus thematisiert wird und nicht speziell die Identifikationsmetriken. Eine gründliche Erprobung dieser Metriken ist jedoch essentiell, um Aussagen über die Anwendbarkeit in praktischen Situationen treffen zu können. Hierzu könnte eine Studie dienen, die anhand von *Real-World-Traces* die Identifikationsgüte der Metriken in stark heterogenen Umgebungen untersucht.

Die Beantwortung dieser Fragestellungen ist als zentral zu betrachten und könnte den Ansatz weiter in Richtung eines Einsatzes in der Praxis bringen.

A Studie zur Anwendbarkeit von Linux Traffic Control

Die Umsetzung eines Zugangsprofils kann softwareseitig an einem Router erfolgen. Zu diesem Zweck existieren bereits verschiedene Lösungen, wobei insbesondere das Traffic Control Subsystem des Betriebssystems Linux herauszustellen ist. Gängige Distributionen bieten bereits eine einsatzfähige Version zum Zugriff auf dieses Subsystem an, so dass der Zugang zu der Software relativ einfach ist. Eine derartige Softwarelösung ist im Kontext der vorliegenden Arbeit interessant, da sie sich für die Umsetzung konkreter Zugangsprofile, die ein Mechanismus wie das adaptive Bandbreitenmanagement vorschlägt, anbietet. Exemplarisch für softwareseitige Lösungen, die das Verhalten von ein- und ausgehendem Datenverkehr manipulieren, greift dieses Kapitel Linux Traffic Control heraus und evaluiert es im Hinblick auf die Anwendbarkeit im Kontext des adaptiven Bandbreitenmanagements. Die Zielsetzung dieses Kapitels ist die Untersuchung der Möglichkeiten von Linux Traffic Control, insbesondere bzgl. der Ratenkontrolle auf ausgehenden Verbindungen.

Die Studie gliedert sich wie folgt. Abschnitt A.1 zeigt zunächst die Funktionsweise von Linux Traffic Control auf und erläutert, wie ein Benutzer des Linux Betriebssystems Zugriff auf dieses Subsystem erhält. Abschnitt A.2 geht kurz auf unabhängige Zusatzsoftware ein, die insbesondere im Kontext der Evaluationsszenarien dieser Studie vorteilhaft sind. Schließlich stellt Abschnitt A.3 die verschiedenen Szenarien vor und präsentiert die Ergebnisse. Der theoretische Hintergrund um Linux Traffic Control wird erweitert, sofern es für die Beschreibung eines Szenarios von Bedeutung ist. Abschnitt A.4 zeigt einige Problemfälle auf, die bei der Arbeit mit der Software im Kontext von Peer-to-Peer Systemen und speziell der vorliegenden Arbeit zu erwarten sind. Die Diskussion der Ergebnisse in Abschnitt A.5 rundet die Studie ab.

A.1 Linux Traffic Control

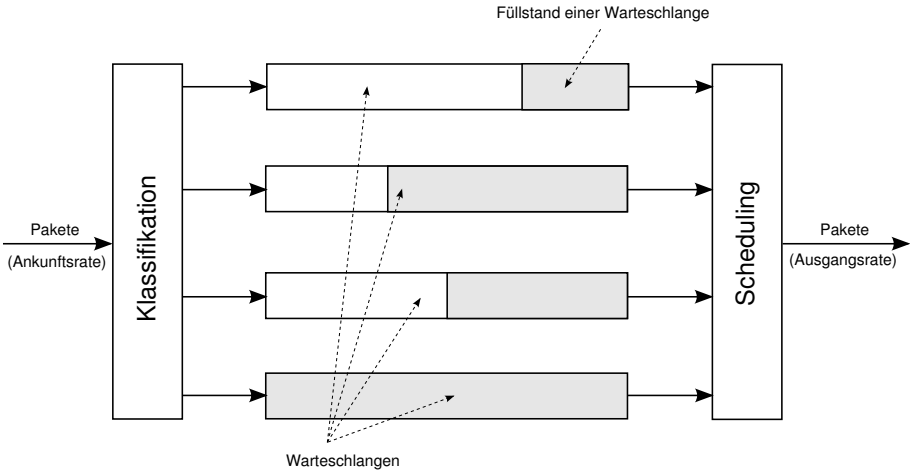


Abbildung A.1: Die Illustration stellt das Grundprinzip von Linux Traffic Control vereinfacht dar. Pakete erreichen das Traffic Control Subsystem mit einer Ankunftsrate, werden klassifiziert und in entsprechende Warteschlangen einsortiert. Das Scheduling sorgt dafür, dass Pakete zu gegebenem Zeitpunkt aus den Warteschlangen entfernt und versendet werden, so dass als Resultat ein Datenfluss mit einer gewissen Ausgangsrate entsteht (Abbildung frei nach [61]).

Das Betriebssystem Linux stellt eine Schnittstelle in Form des Kommandozeilenprogramms `tc` bereit, um auf das Traffic Control Subsystem zuzugreifen. Über `tc` hat der Benutzer die Möglichkeit, entsprechende Befehle an den Linux Kernel weiterzuleiten, um so aktiv das Verhalten des Netzwerk-Softwarestacks zu beeinflussen. Die Möglichkeiten, die Linux Traffic Control zur Verfügung stellt, umfassen die Klassifikation von Paketen, das Scheduling von Paketen sowie verschiedene Strategien zur Warteschlangenbehandlung [61, 62]. Die Klassifikation von Paketen arbeitet im Wesentlichen mit Filtern, die der Benutzer selbst anhand diverser Selektionskriterien definieren kann. Benutzerdefinierter Filter prüfen auf bestimmte Eigenschaften eines Pakets und ordnen sie dementsprechend unterschiedlichen Klassen zu. Die Klassen korrespondieren wiederum mit einer spezifischen Warteschlange, in die

Pakete eingelagert und zu gegebenem Zeitpunkt wieder entnommen werden (Scheduling). Abbildung A.1 zeigt die Integration der Kernbestandteile. Die Implementierung von Traffic Control innerhalb des Linux Kernels fasst das Verwalten von Warteschlangen und das Scheduling unter dem Begriff der Queueing Discipline (qdisc) zusammen. Für qdiscs können verschiedene Strategien zum Einsatz kommen, wie bspw. der sogenannte Token Bucket Filter, der mit einer konstanten Rate Pakete aus einer korrespondierenden Warteschlange zieht, oder aber der Hierarchical Token Bucket, der es ermöglicht, die Gesamtbandbreite anteilmäßig auf verschiedene Klassen aufzuteilen [61].

Vornehmlich werden Konfigurationen für die Priorisierung von bestimmten Datenflüssen angewendet. Derartige Szenarien sind im Wesentlichen statisch, da es genügt, eine Konfiguration einmalig anzuwenden, um den Datenverkehr zu formen. Dies lässt aber nicht den Schluss zu, dass dynamische Einsatzgebiete für Linux Traffic Control uninteressant sind. Gerade in Bezug auf die Regulierung von Peer-to-Peer Datenverkehr ergeben sich hier interessante Nutzungsmöglichkeiten, insbesondere auch bezüglich der Erprobung neuer Systeme. So könnte Linux Traffic Control bspw. dazu dienen, die Zugangsprofile im Rahmen eines adaptiven Bandbreitenmanagements über eine entsprechende Konfiguration umzusetzen. Ähnlich motivierte Einsatzgebiete sind denkbar.

A.2 Traffic Control Next Generation

Der Einsatz von `tc` gestaltet sich allerdings nicht trivial, da die Konfigurationssprache zum Einen nicht leicht zugänglich und zum Anderen unzureichend dokumentiert ist. Das Projekt Linux Advanced Routing & Traffic Control [62] schafft bezüglich der Dokumentation allerdings Abhilfe und stellt zusammen mit [61, 63, 64] einen guten Einstiegspunkt für die Arbeit mit `tc` dar. Für die vorliegende Evaluation wurde jedoch der Umweg über das Projekt Traffic Control Next Generation gewählt, da dieses mit dem Kommandozeilenprogramm `tcng` eine softwareseitige Lösung bereitstellt, um statische Konfigurationen in einer leicht zugänglichen Konfigurationssprache zu schreiben und in entsprechende `tc`-Befehle zu übersetzen. Das Softwarepaket `tcng` enthält zudem die Simulationssoftware `tcsim`. Die Simulationssoftware prüft benutzerdefinierte Konfigurationen zum Einen auf syntaktische Korrektheit und Konsistenz. Zum Anderen ermöglicht sie die exakte Simulation diverser Szenarien unter einer gegebenen Konfiguration. Dies hat mehrere Vorteile, da man ohne `tcng` und `tcsim` auf die Evaluation einer Konfiguration in einem laufenden System zurückgreifen müsste, wodurch der normale Netzwerkverkehr negativ beeinflusst werden könnte. Die Ausgabe von `tcsim` stellt die in der Simulation eingetretenen Ereignisse chronologisch dar. Man unterscheidet die folgenden Ereignisse voneinander:

- Enqueue Event: Ein Paket wurde in eine Warteschlange eingelagert.
- Dequeue Event: Ein Paket wurde aus einer Warteschlange entnommen.
- Packet Loss: Ein Paket wurde aufgrund einer vollen Warteschlange verworfen.

Die Analyse der Ausgabe eines Simulationsdurchlaufs liefert genaue Einblicke, ob das gewünschte Datenverkehrsmodell korrekt umgesetzt wird oder nicht. In Anbetracht der Evaluation von `tc` eignet sich die Simulation besonders, da unterschiedliche Konfigurationen in verschiedenen, für die Zielumgebung typischen Szenarien, einfach und schnell einem Durchführbarkeitstest unterzogen werden können. Dies wäre ohne die Simulation – aus den bereits angesprochenen Gründen – eine zeitintensive Beschäftigung, müsste man doch auf ein laufendes System nebst Applikationen zur Erzeugung spezifischen Datenverkehrs zurückgreifen.

Für den tatsächlichen Einsatz ist `tcng` nicht unbedingt geeignet, denn die Regulierung des Datenverkehrs in einem Peer-to-Peer System ist von hochgradig dynamischer Natur. Dies stellt ein Problem dar, denn `tcng` verarbeitet statische Konfigurationsdateien und setzt diese in Befehle für `tc` um. In einem dynamischen Umfeld ist es notwendig, bei jeder Änderung, den Umweg über `tcng` zu gehen (Erstellen der Konfiguration und Übersetzung in `tc`-konforme Befehle), bevor neue Änderungen an das Traffic Control Subsystem von Linux weitergeleitet werden können. Dies stellt eine zusätzliche Indirektionsebene dar und ist daher für den Live-Einsatz nicht geeignet.

A.3 Evaluationsszenarien

Die Einsatzfähigkeit von `tc` wird im Rahmen dieser Arbeit anhand diverser Szenarien erprobt. Selbstverständlich kann an dieser Stelle keine erschöpfende Überprüfung der Fähigkeiten von `tc` erfolgen. Vielmehr sollen nur solche Szenarien untersucht werden, die für den in dieser Arbeit vorgeschlagenen Ansatz eine praktische Relevanz besit-

zen. Es muss natürlich darauf hingewiesen werden, dass trotz eines realistischen Szenarios die Bedingungen, unter denen die Simulation durchgeführt wird, zu einem gewissen Grad von synthetischer Natur bleiben. Die Ergebnisse sind daher kritisch zu bewerten und eher als Tendenz zu verstehen, die auf praktische Einsatzgebiete, in denen ausgehender Datenverkehr reguliert werden soll, nur bedingt übertragbar sind. Die Behandlung von eingehendem Datenverkehr mittels `tc` ist aufgrund der Irrelevanz für die Anwendung nicht Gegenstand der nachfolgenden Betrachtung.

Die nachfolgenden Unterabschnitte stellen die verschiedenen Szenarien vor und gliedern sich nach Szenarienbeschreibung, der Diskussion der korrespondierenden Konfiguration für `tc` im Format der `tcng` DSL sowie der Auswertung der Ergebnisse der Simulationsläufe. Die individuellen Soll- und Ist-Raten orientieren sich grob an heutigen Internet-Anschlüssen.

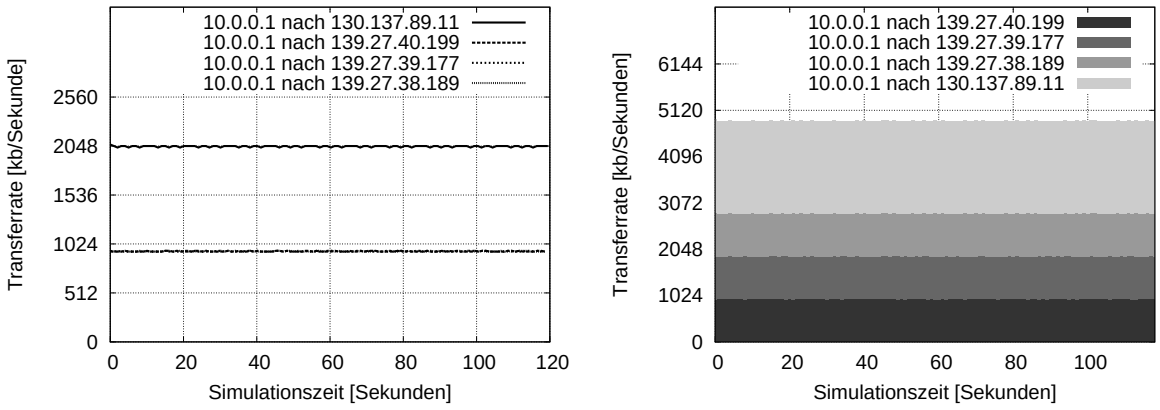
A.3.1 Ratenlimitierung per autonomen System

In diesem Szenario soll untersucht werden, inwiefern eine Ratenlimitierung für spezifische autonome Systeme mit `tc` durchgeführt werden kann. Von der Limitierung des ausgehenden Datenverkehrs pro AS sollen alle Verbindungen betroffen sein, die auf dem Ursprungshost 10.0.0.1 eingeleitet werden und in das selbe AS gerichtet sind. Diese Verbindungen müssen ferner in fairer Weise berücksichtigt werden, so dass Pakete, die einer bestimmten Verbindung zugehörig sind, das Scheduling von Paketen einer anderen Verbindung nicht dominieren. Das Szenario bedient sich zur Überprüfung dieser Vorgaben zwei autonomen Systeme, wobei alle Verbindungen zu Endgeräten mit der IP-Adresse 130.*.*.* AS 1 zugehörig sind und alle Verbindungen zu Endgeräten mit der IP-Adresse 139.*.*.* AS 2. Für Verbindungen in das AS 1 definieren wir eine Ist-Rate von 4096 kb/s, wohingegen über eine Verbindung in AS 2 versucht wird, einen Datenstrom mit einer konstanten Rate von 2048 kb/s zu versenden. Für vom Ursprungshost ausgehende Verbindungen in das AS 1 erfolgt eine Regulierung der Gesamtdatenrate auf 2048 kb/s, während über alle Verbindungen mit einem Zielhost in AS 2 eine maximale Datenrate von 3072 kb/s zulässig ist. Im Falle simultaner Verbindungen in das selbe AS muss eine Aufteilung der Gesamtdatenrate in gleiche Anteile erfolgen.

Die Kenngrößen des Szenarios sind:

- Paketgröße: 20 Byte IP-Header, 20 Byte TCP-Header ohne Optionen, 984 Byte Payload
- Ist-Rate:
  - 4096 kb/s über die Verbindung von (10.0.0.1:*,130.*.*.*:*)
  - 2048 kb/s über jede der drei Verbindungen von (10.0.0.1:*,130.*.*.*:*)
- Soll-Rate:
  - 2048 kb/s insgesamt für alle Verbindungen von (10.0.0.1:*,130.*.*.*:*)
  - 3072 kb/s insgesamt für alle Verbindungen von (10.0.0.1:*,139.*.*.*:*)
- Simulationsdauer: 2 Minuten

Listing A.1 zeigt die Konfiguration des Shapers. Verbindungen werden einfacherweise anhand eines IP-Präfix-Matchings über den ersten 8 Bit einem AS zugeordnet. Hier ist natürlich anzumerken, dass in der Praxis die Ausmaße eines sehr großen AS mit derart naiven Filterregeln nicht vollständig erfasst werden können. Komplexere Filterregeln sind in realistischen Umgebungen notwendig. Das Prinzip bleibt jedoch das gleiche, so dass diese einfache Zuordnung von Endgerät zu AS für die vorliegende Studie ausreichend ist. Problematisch ist jedoch nun, dass eine Warteschlange potentiell Pakete beinhaltet, die unterschiedlichen IP-Verbindungen angehören. Standardmäßig verwendet `tc` eine FIFO-Implementierung für die Warteschlange, welche drei Prioritätsstufen besitzt, auf die einzelne Pakete entsprechend des ToS-Feldes im IP-Header verteilt werden [62]. Jede Prioritätsstufe wird durch eine eigene Warteschlange realisiert. Bei der Verwendung dieser FIFO-Warteschlange werden Pakete allerdings unzureichend eingestuft, da das ToS-Feld in unserem Beispiel nicht genutzt wird. Sämtliche Pakete gelangen also in dieselbe Warteschlange, was dazu führen kann, dass bei voller Auslastung der verfügbaren Gesamtbandbreite Pakete, die einer bestimmten Verbindung zugesprochen werden, häufiger die Warteschlange verlassen, als Pakete, die einer anderen Verbindung angehören. Um diesem Problem gerecht zu werden, stellt `tc` eine spezielle Warteschlange bereit, die sogenannte Stochastic Fairness Queue (sfq) [62]. Unter der sfq versteht man eine Warteschlange, die bei voller Auslastung der zugewiesenen Gesamtbandbreite für Fairness unter den einzelnen Verbindungen sorgt.



(i) Visualisierung als Zeitreihe

(ii) Visualisierung als Histogramm

Abbildung A.2: Ratenlimitierung per autonomen System mit stochastischer Fairness

Die sfq implementiert somit keine Strategie zur Formung des Datenverkehrs (wie z. B. der htb), sondern betreibt lediglich Scheduling, so dass ihr Einsatz insbesondere bei den Blattknoten einer Klassenhierarchie sinnvoll erscheint. Die Fairness-Bedingung wird intern über insgesamt 127 FIFO-Warteschlangen umgesetzt, die abwechselnd senden dürfen. Eine simple Zuordnung mittels Hashwert über einer Verbindung entscheidet über die Zugehörigkeit eines Paketes zu einer dieser FIFO-Warteschlangen. Bei vielen Verbindungen sind Kollisionen durch die limitierte Anzahl von FIFO-Warteschlangen natürlich unvermeidlich. Die Implementierung von sfq mindert dieses Problem, indem die Hashfunktion gewechselt wird. Dieses Verhalten garantiert zwar keine absolute Fairness, aber zumindest eine stochastische Fairness über alle Verbindungen, die sich die selbe sfq teilen.

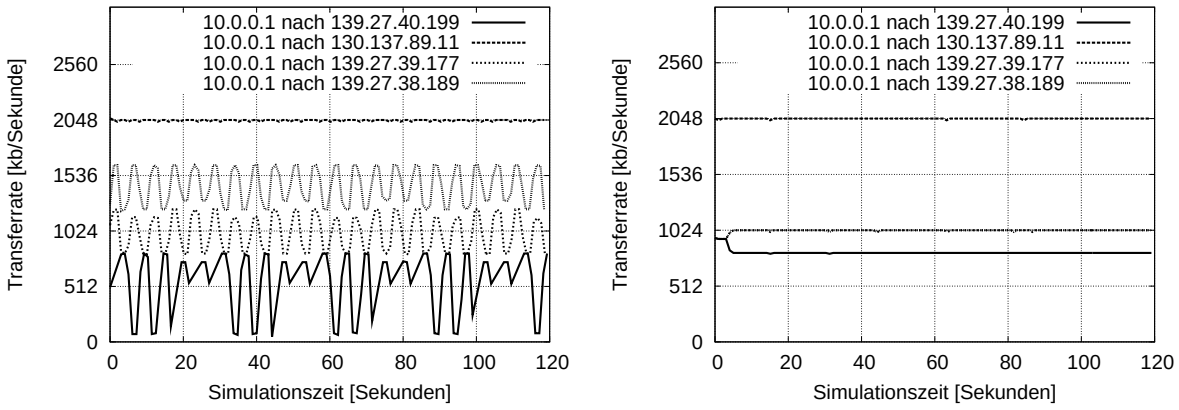
```
#include "fields.tc"

dev eth0 {
    egress {
        class ( <$AS_2> ) if ip_dst >> 24 == 139 ;
        class ( <$AS_1> ) if ip_dst >> 24 == 130 ;
        htb () {
            class ( rate 5120 kbps, ceil 5120 kbps ) {
                $AS_1 = class ( rate 2048 kbps, ceil 2048 kbps ) { sfq; } ;
                $AS_2 = class ( rate 3072 kbps, ceil 3072 kbps ) { sfq; } ;
            }
        }
    }
}
```

Listing A.1: Shaping-Konfiguration zur Ratenlimitierung per AS

Abbildung A.2(i) zeigt schließlich die Auswertung des Simulationslaufs unter Verwendung stochastischer Fairness. In der Zeitreihe ist deutlich zu erkennen, dass die spezifischen Transferraten über allen Verbindungen in AS 2 nahe an dem zu erwartenden Wert von 3072 kb/s/3 = 1024 kb/s liegen. Dass die Aufteilung des ausgehenden Datenverkehrs nach AS 2 in fairer Weise erfolgt, illustriert das Histogramm in Abbildung A.2(ii).

Hingegen zeigt Abbildung A.3 die Auswertung zweier Simulationsläufe ohne stochastische Fairness. Der Simulationslauf in Abbildung A.3(i) wurde mit unveränderten Ist- und Soll-Raten durchgeführt. Man erkennt eine sehr starke Schwankung um die Werte 1440 kb/s, 1024 kb/s und 480 kb/s. Die unfaire Aufteilung ist in dieser Grafik sehr schön an den einzelnen Verbindungen zu erkennen, denn die Summe der Spitzenwerte der Verbindungen 139.27.38.189 und 139.27.39.177 sind phasenverschoben zu den Spitzenwerten, die zu Verbindung 139.27.40.199 gehören. Dies zeigt, dass die beiden erstgenannten Verbindungen abschnittsweise bevorzugt werden, was eine Reduzierung der Datenrate im selben Maße für Verbindung 139.27.40.199 zur Folge hat.



(i) Simulation mit hohen Transferraten

(ii) Simulation mit niedrigen Transferraten

Abbildung A.3: Ratenlimitierung per autonomen System *ohne* stochastische Fairness: Der Fehler prägt sich bei hohen Transferraten (i) deutlich stärker aus, als bei niedrigen Transferraten (ii)

Der sehr starke Fehler über einzelnen Verbindungen lässt sich durch den hohen Gesamtdurchsatz erklären. Abbildung A.3(ii) zeigt einen Simulationslauf mit leicht angepasster Shaping- und Szenario-Konfiguration. Die Ist-Raten wurden um den Faktor 2 reduziert. Die Soll-Raten des htb sind für die Gesamtheit der Verbindungen nach AS 1 auf den Wert 1024 kb/s, und für alle Verbindungen nach AS 2 auf den Wert 2048 kb/s festgelegt. Die Grafik zeigt deutlich, dass der Fehler gegenüber dem Simulationslauf mit höheren Transferraten deutlich geringer ausfällt. Interessant ist hier die Tatsache, dass alle Verbindungen in das AS 2 zunächst bei $\approx$ 1024 kb/s beginnen, sich aber relativ schnell auf einen Endwert einstellen, der für einen Teil der Verbindungen besonders günstig ist ($\approx$ 1024 kb/s), aber andere Verbindungen dominiert ($\approx$ 800 kb/s).

Obwohl sich die individuellen Resultate beider Simulationsläufe stark voneinander unterscheiden, ist deutlich zu erkennen, dass eine faire Aufteilung der Gesamtdatenrate ohne stochastische Fairness nicht erfolgt. Für die Soll-Rate der einzelnen Verbindung in das AS 1 gilt dies nicht. Die Begrenzung der Datenrate wird in diesem Fall in beiden Szenarien ordnungsgemäß realisiert.

A.3.2 Ratenlimitierung per Verbindung zwischen zwei Endsystemen

Die Motivation dieses Szenarios unterscheidet sich grundsätzlich von derjenigen des vorangegangenen Abschnitts. Wir möchten nun nicht mehr nach autonomen Systemen klassifizieren, sondern pro Verbindung auf IP-Ebene. Die wesentlichen Unterschiede ergeben sich in diesem Szenario bezüglich der Shaping-Konfiguration. Die Beschreibung des Szenarios unterscheidet sich lediglich in den konkreten Werten für die Ist-Raten, mit denen über jede Verbindung Datenpakete gesendet werden.

Die Kenngrößen des Szenarios sind:

- Paketgröße: 20 Byte IP-Header, 20 Byte TCP-Header ohne Optionen, 984 Byte Payload
- Ist-Rate: 2048 kb/s (über jeweils eine Verbindung)
- Soll-Rate:
  - 512 kb/s über (10.0.0.1:*,130.137.89.11:*)
  - 1024 kb/s über (10.0.0.1:*,139.27.38.189:*)
  - 1536 kb/s über (10.0.0.1:*,139.27.39.177:*)
  - 2048 kb/s über (10.0.0.1:*,139.27.40.199:*)
- Simulationsdauer: 2 Minuten

Listing A.2 zeigt die Konfiguration des Shapers. Die Limitierung erfolgt pro Verbindung, so dass wir insgesamt vier Klassen benötigen, die anhand der IP-Adresse der Zielmaschine Pakete filtern und in die entsprechenden Warteschlangen einordnen. Zum Einsatz gelangt wieder ein Hierarchical Token Bucket. Der Zusatz der sfq ist in diesem

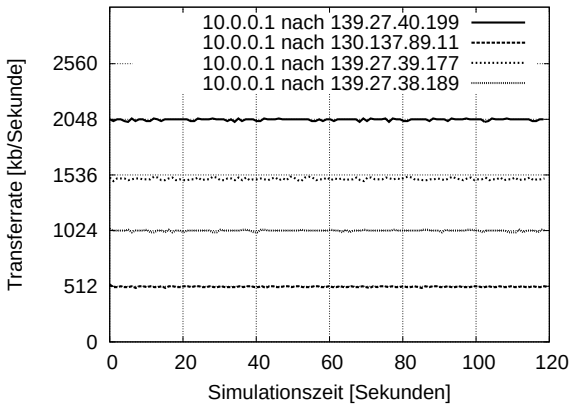


Abbildung A.4: Ratenlimitierung per Verbindung zwischen zwei Endsystemen

Szenario nicht notwendig, da wir zum Einen für jede Verbindung eine eigene Warteschlange verwalten und zum Anderen davon ausgehen, obere Schranken für einzelne Verbindungen konkret angeben zu können. Es ist daher nicht notwendig, die verfügbare Gesamtbandbreite explizit über mehrere Verbindungen fair aufzuteilen. Die Limitierung der Transferraten über den Klassen erfolgt sukzessive ansteigend, beginnend bei 512 kb/s in 512 kb/s Schritten.

```
#include "fields.tc"

dev eth0 {
    egress {
        class ( <$conn_1> ) if ip_dst == 130.137.89.11 ;
        class ( <$conn_2> ) if ip_dst == 139.27.38.189 ;
        class ( <$conn_3> ) if ip_dst == 139.27.39.177 ;
        class ( <$conn_4> ) if ip_dst == 139.27.40.199 ;
        htb () {
            class ( rate 5120 kbps, ceil 5120 kbps ) {
                $conn_1 = class ( rate  512 kbps, ceil  512 kbps ) ;
                $conn_2 = class ( rate 1024 kbps, ceil 1024 kbps ) ;
                $conn_3 = class ( rate 1536 kbps, ceil 1536 kbps ) ;
                $conn_4 = class ( rate 2048 kbps, ceil 2048 kbps ) ;
            }
        }
    }
}
```

Listing A.2: Shaping-Konfiguration zur Ratenlimitierung per IP-Verbindung

Abbildung A.4 zeigt die Ergebnisse des Simulationslaufs. Deutlich zu erkennen ist die zu erwartende Staffelung der individuellen Transferraten. Die spezifischen Raten oszillieren um die definierten Werte, was zeigt, dass die `tc` Konfiguration aus Listing A.2 korrekt vom Traffic Control Subsystem des Kernels umgesetzt wird.

A.3.3 Ratenlimitierung per Applikation

Die bisherigen Szenarien zeigen deutlich, dass eine Regelung des Datenverkehrs mit einer flachen Klassen-Hierarchie korrekt von `tc` umgesetzt wird. Wir sind nun daran interessiert, eine Ratenlimitierung für eine spezifische Applikation vorzunehmen. Dies ist bei einer Anwendung in einem verteilten System besonders wichtig, da zum Einen nur der Datenverkehr einer Regulierung unterzogen werden soll, der von der Applikation selbst erzeugt wird, zum Anderen ist dies aber auch der Natur von Testbeds geschuldet: Aus Mangel an Ressourcen ist es innerhalb eines Clusters von Maschinen notwendig, dass sich multiple Instanzen der zu evaluierenden Applikation eine physikalische Maschine teilen. Eine solche Umgebung erfordert eine strikte Klassifikation auf Ebene des 4-Tupels

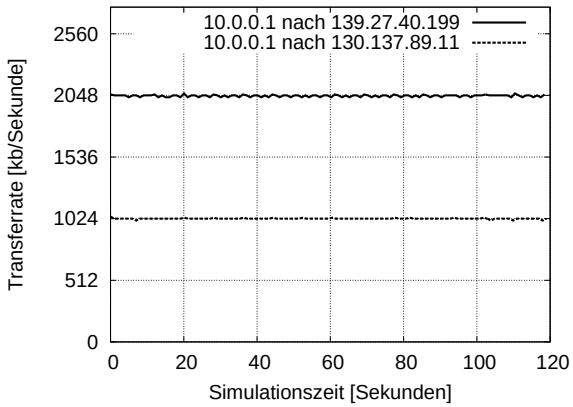


Abbildung A.5: Ratenlimitierung per Applikation

(Quell-IP, Quell-Port, Ziel-IP, Ziel-Port), um eine Verbindung zwischen zwei Applikationen eindeutig zu beschreiben. Die Ausgangsbasis für dieses Szenario ist wieder der Ursprungshost 10.0.0.1, der nun zwei Verbindungen an eine Applikation auf unterschiedlichen Endsystemen öffnet und darüber Daten sendet. Beide Datenströme werden durch einen Hierarchical Token Bucket reguliert. Die Aufteilung der verfügbaren Gesamtbandbreite von 3072 kb/s erfolgt im Verhältnis 1:2.

Die Kenngrößen des Szenarios sind:

- Paketgröße: 20 Byte IP-Header, 20 Byte TCP-Header ohne Optionen, 984 Byte Payload
- Ist-Rate: 4096 kb/s (über jeweils eine Verbindung)
- Soll-Rate:
  - 1024 kb/s über (10.0.0.1:37583,130.137.89.11:5000)
  - 2048 kb/s über (10.0.0.1:40302,139.27.40.199:5002)
- Simulationsdauer: 2 Minuten

Listing A.3 zeigt die zu verwendende Shaping-Konfiguration für dieses Szenario. Die Klassen filtern nun nach den Attributen des 4-Tupels (Quell-IP, Quell-Port, Ziel-IP, Ziel-Port). Entscheidend ist, dass hier eine Verbindung eindeutig identifiziert werden muss, was Wissen über anonyme Ports voraussetzt, die der Verbindungsinitiator bei der Anfrage nutzt. Dieses Wissen ist unter normalen Umständen nicht bekannt.

```
#include "fields.tc"

dev eth0 {
    egress {
        class ( <$conn_1> ) if ip_dst == 130.137.89.11 && ip_src == 10.0.0.1 &&
                               tcp_sport == 37583 && tcp_dport == 5000;
        class ( <$conn_2> ) if ip_dst == 139.27.40.199 && ip_src == 10.0.0.1 &&
                               tcp_sport == 40302 && tcp_dport == 5002;
        htb () {
            class ( rate 3072 kbps, ceil 3072 kbps ) {
                $conn_1 = class ( rate 1024 kbps, ceil 1024 kbps ) ;
                $conn_2 = class ( rate 2048 kbps, ceil 2048 kbps ) ;
            }
        }
    }
}
```

Listing A.3: Shaping-Konfiguration zur Ratenlimitierung per Applikation

Abbildung A.5 zeigt die Ergebnisse des Simulationslaufs. Die korrekte Umsetzung der individuellen Transferraten ist deutlich zu erkennen.

A.3.4 Aufteilung von Intra- und Inter-ISP-Datenverkehr mit htb

Der Hierarchical Token Bucket ist mit einem weiteren Merkmal ausgestattet, welches dafür sorgt, dass ungenutzte Tokens nicht verfallen. Liegt bspw. die Ist-Rate, mit der Pakete einer bestimmten Klasse eingehen, unterhalb ihrer reservierten Bandbreite (vgl. `rate`-Parameter), so können verbleibende Tokens an Nachbarklassen weitergegeben werden. Dieses Konzept ist unter dem Begriff *Token Borrowing* bekannt. Die Vater-Klasse kennt den Zustand aller Kind-Klassen, so dass die Information über frei verfügbare Tokens dort vorliegt und abgerufen werden kann. Dies funktioniert rekursiv bis zur Wurzel der Klassenhierarchie, so dass freie Tokens auch bei einer komplexen Hierarchie nicht ungenutzt bleiben müssen. Übersteigt die Ist-Rate einer Klasse die ihr zugesicherte Datenrate, so können von der Vater-Klasse weitere Tokens angefordert werden. Das Anfordern weiterer Tokens findet aber nur dann statt, wenn der Parameter `ceil` der Klasse mit einem Wert belegt wurde, der über dem Wert des Parameters `rate` liegt. `ceil` stellt somit gleichzeitig auch die maximale Datenrate dar, die einer Klasse zur Verfügung gestellt werden kann. Die Bestimmung von konkreten Werten für diesen Parameter ist allerdings nicht trivial und erfordert detaillierte Kenntnisse über die zu erwartenden Datenflüsse, insbesondere in Korrespondenz mit anderen Klassen auf derselben Ebene der Klassenhierarchie.

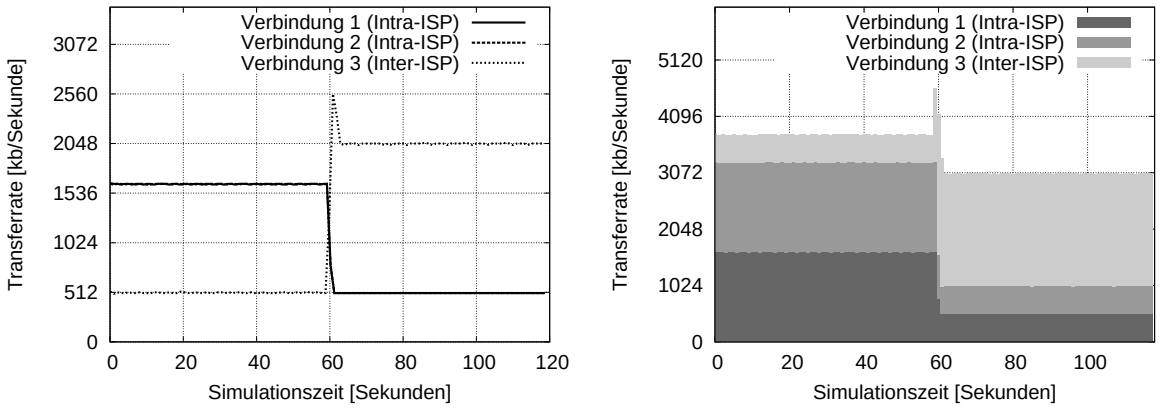
Durch das Konzept des Token Borrowing stellt der htb im Sinne einer optimalen Auslastung der verfügbaren Gesamtbandbreite eine wünschenswerte Strategie dar. Die Anwendungsmöglichkeiten sind insbesondere im Kontext der vorliegenden Arbeit interessant. In einem Peer-to-Peer System wie BitTorrent ist die maximale Anzahl von Verbindungen für gewöhnlich auf einen konstanten Wert begrenzt. Verwendet man nun lokalitätsfördernde Mechanismen, wie Biased Neighbour Selection oder Biased Unchoking (vgl. Abschnitt 3.3), so wird sich die Mehrzahl der Verbindungen an Peers richten, die aufgrund einer Lokalitätsmetrik nah an dem anfragenden Peer liegen, selbst wenn individuelle Transferraten anderer, weiter entfernter Peers eine bessere Auslastung versprechen. Dies kann zu einer verminderten Dienstgüte führen, was wiederum nicht im Interesse des Benutzers ist.

Dieses Problem kann vermindert werden, wenn man von einer Regulierung des Datenverkehrs innerhalb der Applikation absieht und das System selbst, bspw. durch den Einsatz eines htb, regelt. Es ist möglich, eingehende Pakete nach Zugehörigkeit zu einer Intra- oder Inter-ISP-Verbindung zu klassifizieren. Anhand dieser Unterscheidung kann unter Verwendung von htb ein bestimmter prozentualer Anteil k der verfügbaren Gesamtbandbreite für ISP-internen Datenverkehr reserviert werden. Im Gegenzug bedeutet dies, dass unter Garantie $1-k$ der Gesamtbandbreite für Verbindungen aufgewendet werden darf, die über die Grenzen des lokalen ISPs hinausgehen. Borrowing kommt in diesem Szenario dann zum Einsatz, wenn der tatsächliche Anteil k' des Datenverkehrs über lokale Verbindungen unterhalb von k liegt. In diesem Fall können ungenutzte Tokens entsprechend einer Rate von $|k-k'|$ für Inter-ISP-Verbindungen aufgewendet werden. Dadurch können Einbußen hinsichtlich der Dienstgüte vermindert werden, denn durch die zusätzliche Transferrate für Inter-ISP-Verbindungen kann sich ein Peer in einem BitTorrent-ähnlichen System für entfernte Tauschpartner interessanter machen. Wichtig ist zu erkennen, dass dies in Abhängigkeit des Anteils von lokalem Datenverkehr geschieht, sprich, wenn genügend gute lokale Verbindungen existieren, so bleibt der Datenverkehr über Inter-ISP-Verbindungen auf einen Anteil von $1-k$ der Gesamtbandbreite beschränkt.

Wir wollen dies anhand einer einfachen Beispielkonfiguration mit `tcsim` untersuchen. In diesem Szenario gehen wir von zwei Verbindungen aus, die an andere Peers im lokalen ISP-Netz gerichtet sind und einer Verbindung zu einem Peer, der sich in einem anderen ISP-Netz bewegt. Die Simulationszeit beträgt 120 Sekunden, wobei innerhalb der ersten 60 Sekunden versucht wird, über Intra-ISP-Verbindungen mit einer Gesamtrate von 5120 kb/s zu senden. Diese Datenrate verringert sich für die restlichen 60 Sekunden auf insgesamt 1024 kb/s.

Die Kenngrößen des Szenarios sind:

- Paketgröße: 20 Byte IP-Header, 20 Byte TCP-Header ohne Optionen, 984 Byte Payload
- Ist-Rate: 4096 kb/s (über jeweils eine Verbindung)
  - Für Zeit $t \in [0;59]$ mit 2560 kb/s über jede Intra-ISP-Verbindung (Ziel-IP-Präfix: 130.83.0.0/16)
  - Für Zeit $t \in [60;120)$ mit 512 kb/s über jede Intra-ISP-Verbindung (Ziel-IP-Präfix: 130.83.0.0/16)
  - Für Zeit $t \in [0;120)$ mit 2048 kb/s über die Inter-ISP-Verbindung (Ziel-IP-Präfix: 88.0.0.0/8)
- Soll-Rate:
  - 3584 kb/s über (130.83.139.83:*,130.83.*.*:*) mit Spitzenwert 4096 kb/s



(i) Visualisierung als Zeitreihe

(ii) Visualisierung als Histogramm

Abbildung A.6: Aufteilung von Intra- und Inter-ISP-Datenverkehr mit htb und Token Borrowing

  - 512 kb/s über (130.83.139.83:10004,88.195.10.37:10005) mit Spitzenwert 4096 kb/s
- Simulationsdauer: 2 Minuten

Listing A.4 zeigt die Konfiguration der Klassen und des htb. Pakete werden wieder entsprechend eines IP-Präfixes gefiltert, der nun zwischen Quell-IP-Adresse und Ziel-IP-Adresse auf Gleichheit prüft. Im Falle der Gleichheit wird die Verbindung als ISP-lokal eingestuft, andernfalls gehen wir von einer Inter-ISP-Verbindung aus. Der htb ist mit einer maximalen Soll-Rate von 4096 kb/s definiert, was einer realistischen Konfiguration heutiger Internet-Anschlüsse entspricht. Für Datenverkehr über Intra-ISP-Verbindungen werden $k = 0{,}875$ der verfügbaren Gesamtbandbreite von 4096 kb/s reserviert, was einer theoretischen maximalen Bandbreite von 3584 kb/s entspricht. Sollte der restliche Anteil von $1 - k = 0{,}125$ der Gesamtbandbreite nicht über Inter-ISP-Verbindungen genutzt werden, so darf die gesamte Bandbreite für lokale Verbindungen verwendet werden. Dies gilt analog für Inter-ISP-Verbindungen.

```
#include "fields.tc"

dev eth0 {
    egress {
        class (<$local>) if ip_src >> 24 == 130 && ip_dst >> 24 == 130 ;
        class (<$inter>) if ip_src >> 24 == 130 && ip_dst >> 24 != 130 ;
        htb () {
            class ( rate 4096 kbps, ceil 4096 kbps ) {
                $local = class ( rate 3584 kbps, ceil 4096 kbps ) { sfq; }
                $inter = class ( rate 512 kbps, ceil 4096 kbps ) { sfq ; }
            }
        }
    }
}
```

Listing A.4: Shaping-Konfiguration des Borrowing-Szenarios

Über beide Intra-ISP-Verbindungen wird versucht, einen Datenstrom mit insgesamt 5120 kb/s zu senden. Der htb reguliert den Gesamtdatenverkehr für diese Klasse auf 3584 kb/s herunter, wodurch sich nach einer fairen Aufteilung die konkreten Endwerte von $3584 \text{ kb/s}/2 = 1792 \text{ kb/s}$ ergeben. Abbildung A.6(i) zeigt, dass die Ist-Datenraten beider Verbindungen relativ nahe an diesem theoretischen Wert liegen. Da über die ersten 60 Sekunden der Intra-ISP-Datenverkehr ausgeschöpft wird, erfolgt korrekterweise die Regulierung des Inter-ISP-Datenverkehrs auf die in der Shaping-Konfiguration definierten 512 kb/s. Zum Zeitpunkt $t = 60$ verringert sich jedoch der Durchsatz über den Intra-ISP-Verbindungen auf jeweils 512 kb/s, wodurch ein Überschuss an Tokens im Gegenwert einer Rate von $3584 \text{ kb/s} - 2 \cdot 512 \text{ kb/s} = 2560 \text{ kb/s}$ entsteht. Die ungenutzten Tokens können nun von der Inter-ISP-Verbindung genutzt werden. Dies ist deutlich in Abbildung A.6(ii) zu erkennen, da über das Zeitintervall $[60; 120)$ diese Verbindung ihre Ist-Rate von 2048 kb/s komplett ausreizen kann.

Dieses Beispiel zeigt sehr gut, dass htb unter Verwendung von Token Borrowing für das beschriebene Einsatzgebiet eine gute Lösung darstellt. Inwiefern allerdings eine derartige QoS-Konfiguration von lokalitätsfördernden Maßnahmen auf Applikationsseite unterstützt werden kann, liegt außerhalb des Rahmens dieser Evaluation.

A.4 Probleme mit `tc` in einem G-Lab Testbed

Das Ziel der vorliegenden Evaluation ist es, nicht nur die Funktionsfähigkeit von `tc` in bestimmten Szenarien zu prüfen, sondern auch ihre Einsatzfähigkeit zu diskutieren. Die Evaluation des in dieser Arbeit vorgestellten Ansatzes erfolgt in einem G-Lab Live Testbed, welches auf der PlanetLab-Software basiert. Der Einsatz von `tc` ist in dieser Umgebung aufgrund der nachfolgend angeführten Probleme nicht möglich.

Problem 1: Die Netzwerkschicht eines physikalischen G-Lab Rechners wird bereits durch `tc` reguliert.

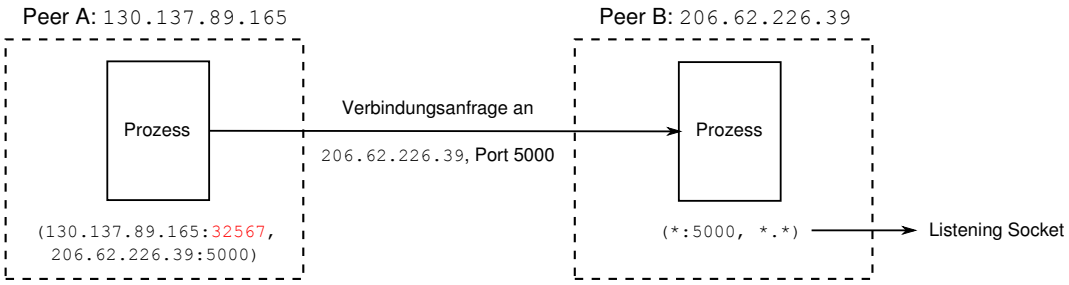
Sämtliche Knoten, die im G-Lab Verbund eingesetzt werden, nutzen die PlanetLab-Software, um virtualisierte Maschinen den Benutzern zur Verfügung zu stellen. Diese Virtualisierung erfolgt allerdings nicht auf den unteren Ebenen des ISO/OSI-Schichtenmodells, was für Netzwerksoftware, die auf diesen Schichten arbeitet, grundsätzlich ein Problem darstellt. Ferner wird `tc` durch die PlanetLab-Software bereits selbst genutzt, damit eine faire Aufteilung der Übertragungskapazitäten des Netzwerks zwischen allen virtualisierten Systemen, die auf einer physikalischen Maschine laufen, erfolgen kann. Für reguläre Benutzer ist `tc` damit nicht zugänglich. Dies wäre auch nicht im Sinne der Benutzer selbst, da durch fehlerhafte Konfigurationen eine starke Einschränkung der Funktionalität des G-Lab Systems die Folge wäre.

Problem 2: Anonyme Ports erschweren eine korrekte Ratenlimitierung.

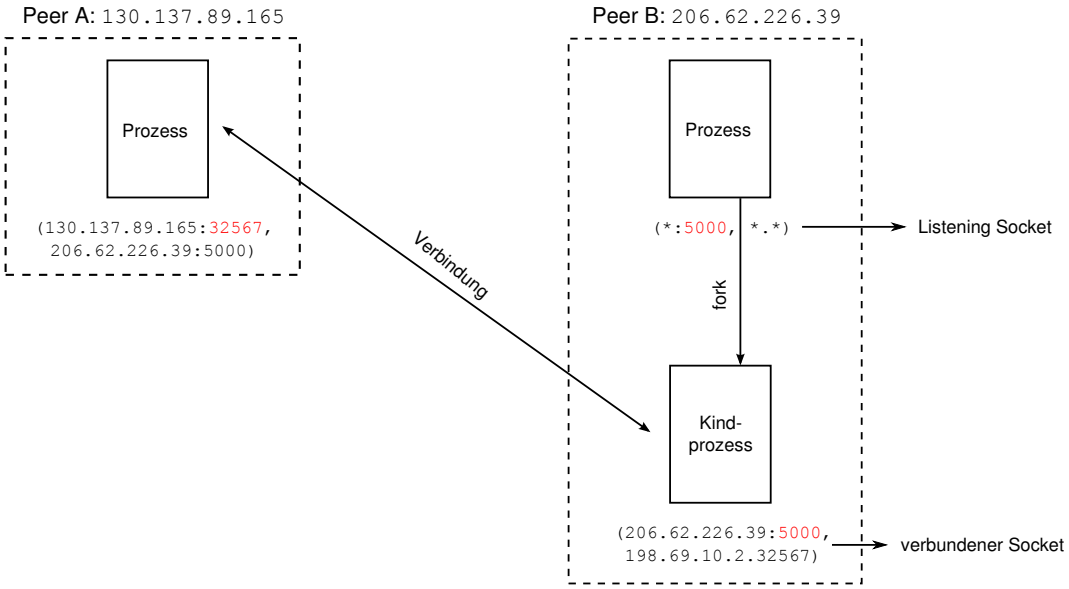
Wir betrachten ein einfaches Szenario, in dem zwei Peers untereinander Daten austauschen möchten. Abbildung A.7 veranschaulicht das Szenario und fasst zusammen, was auf TCP-Ebene während und nach der Verbindungsherstellung passiert. Die entsprechenden Schritte sollen in chronologischer Abfolge erläutert werden.

1. Zunächst stellt der Peer, der auf Maschine 130.137.89.165 läuft, eine Verbindungsanfrage an seinen Kommunikationspartner auf Maschine 206.62.226.39. Die Anfrage ist an den entsprechenden Listening Socket des Kommunikationspartners gerichtet. Dieser Kommunikationsendpunkt akzeptiert eingehende Verbindungen von sämtlichen Maschinen über alle dem physikalischen System angeschlossenen Netzwerkgeräte. Dies wird durch die Verwendung von Wildcards in Abbildung A.7(i) illustriert. Für die Verbindungsanfrage nutzt der Verbindungsinitiator Port 32567. Dies ist ein zufällig gewählter, anonymer Port, der in der Regel außerhalb des Prozesses, der ihn erzeugt hat, a priori *nicht bekannt* ist.
2. Der Kommunikationspartner erhält die Verbindungsanfrage und erstellt einen neuen Thread, um diese Verbindung zu verwalten. Dies ist auch in einigen P2P-Systemen üblich. Zu diesem Zeitpunkt müssen wir zwischen dem *Listening* Socket und dem *verbundenen* Socket unterscheiden. Letzterer wird von dem neu erstellten Thread verwaltet und ermöglicht es der Applikation auf weitere Verbindungsanfragen im Haupt-Thread zu reagieren.
3. Abbildung A.7(ii) stellt die Situation nach der Verbindungsherstellung dar. Zu erkennen ist nun, dass die Verbindung weiterhin über den anonymen Port und die IP-Adresse des Initiators sowie dem Listening Port und der IP-Adresse des Empfängers identifiziert wird. Auffallend ist die Ambiguität auf Seiten des Empfängers bezüglich der Ports beider Sockets. Um eindeutig aufzulösen, für welchen Socket erhaltene Datenpakete bestimmt sind, prüft die TCP-Implementierung auf Seiten des Empfängers zusätzlich die Sender-IP-Adresse und den Sender-Port. Dies ermöglicht eine korrekte Weiterleitung an den entsprechenden Socket.

Die Betrachtung macht deutlich, dass der Datenfluss zwischen zwei Applikationen entscheidend von den anonymen Ports, die bei der Verbindungsherstellung zufällig gewählt werden, abhängig ist. Diese Ports sind aber in der Regel nicht bekannt, so dass eine Ratenlimitierung, wie wir sie für die Evaluation des in dieser Arbeit vorgestellten Ansatzes benötigen würden, erschwert wird. Damit `tc` in einem solchen Szenario anwendbar ist, müsste der ausgehende Datenverkehr auf Verbindungsebene reguliert werden können. Hierzu ist es allerdings notwendig, eine Verbindung über das 4-Tupel (Quell-IP, Quell-Port, Ziel-IP, Ziel-Port) eindeutig zu identifizieren. Dieses Problem kann prinzipiell gelöst werden, wenn man Zugriff auf die Implementierung des Peer-to-Peer Systems hat. Der Empfänger einer



(i) Verbindungsanfrage von Peer A gerichtet an Peer B



(ii) Prozess auf Peer B erzeugt nebenläufigen Kind-Prozess für Verbindung

Abbildung A.7: Zuweisung eines anonymen Ports bei Verbindungsherstellung

Verbindungsanfrage kennt den anonymen Port, über den er vom Sender kontaktiert wurde und könnte diesen an eine weitere Komponente senden, die sämtliche anonymen Ports verwaltet und einer Peer-ID zuordnen kann. Dies ist allerdings mit einem erheblichen zusätzlichen Aufwand verbunden, da zum Einen der Quelltext des Peer-to-Peer Systems modifiziert, zum Anderen aber auch die Implementierung der beschriebenen Komponenten erfolgen muss.

A.5 Diskussion

Zunächst muss noch einmal ausdrücklich gesagt werden, dass das Traffic Control Subsystem von Linux auf sehr vielfältige Weise Möglichkeiten zur Regulierung des Datenverkehrs an den Benutzer gibt. Bspw. sind gerade in Echtzeit-Anwendungen, zu denen auch Live Streaming Applikationen zählen, Mechanismen wie Hierarchical Fair Service Curves von Interesse. `tc` stellt eine Vielzahl von Mechanismen bereit, deren Evaluation in Bezug auf die Anwendbarkeit in dynamischen Szenarien durchaus interessant erscheinen. Eine solche Ausarbeitung kann im Rahmen der vorliegenden Schrift aus Gründen des Umfangs allerdings nicht erfolgen. Die Thematik bietet aber eine Vertiefung in Form einer weiteren studentischen Arbeit an, die sich gemessen am geschätzten Umfang im Rahmen einer einsemestrigen Studienarbeit oder einer Bachelorarbeit bewegt.

Die in dieser kurzen Studie vorgestellten Szenarien sind für die Umsetzung des in dieser Arbeit vorgestellten Ansatzes einfach und ausreichend. Die in Abschnitt A.4 vorgestellten Probleme verbieten allerdings die Nutzbarkeit von `tc` vor dem Hintergrund der Evaluation in einem PlanetLab-basierten Testbed. An dieser Stelle muss ausdrücklich darauf hingewiesen werden, dass anonyme Ports die Benutzung von `tc` zwar erschweren, aber für die Anwendung selbst kein Hindernis darstellen. Das Problem besteht vielmehr dadurch, dass multiple Instanzen derselben Applikation auf einem virtualisierten System nebeneinander lauffähig sein müssen, damit die Anwendung in einem

Rechnerverbund getestet werden kann. Für ein reales Szenario, in dem in der Regel nur eine Instanz der Applikation auf einer Maschine läuft, ist dieses Problem nicht weiter nennenswert. Hier ist es möglich, mit eindeutigen Filtern über dem 3-Tupel (Quell-IP, Ziel-IP, Ziel-Port) zu arbeiten[1].

[1] Dies gilt unter der Voraussetzung, dass der Ziel-Port ein bekannter Port ist, der von allen Applikationen der gleichen Gattung genutzt wird.

B Limitierende Faktoren bei der Datenerhebung

Die Evaluationsplattform G-Lab zeigte sich im Verlauf der Testdurchführungen als deutlich instabil. Die Ursachen dafür sind nicht klar einzuordnen, da die konkrete Konfiguration und Arbeitsweise einzelner G-Lab Knoten für den Benutzer transparent ist. Durch umfangreiches Debugging und Testen war es möglich, einige Problemfelder zu identifizieren. Inwiefern diese Problemfelder von NextShare, dem System selbst, oder der Netzwerk-Architektur abhängig sind, konnte im Rahmen der vorliegenden Arbeit nicht erörtert werden.

- Bei der Mehrheit der Testläufe konnte man beobachten, dass ab einem gewissen Zeitpunkt die Transferraten zwischen einzelnen G-Lab Maschinen drastisch einbrechen. Dieses Problem erweckte zunächst den Anschein, zeitabhängig zu sein, da bereits zu Beginn einer Reihe von Testläufen ein vernünftiges Arbeiten mit dem G-Lab System nicht möglich gewesen ist. Spätere Tests zeigten jedoch auch, dass bei konsekutiv durchgeführten Testläufen sich nach einer gewissen Zeit dieselben Symptome zeigen. Eine verlässliche Möglichkeit, die aktuelle Performanz einer Verbindung zwischen zwei Endsystemen zu messen, bietet das Linux-Werkzeug `iperf`. Mehrmalige Tests mit `iperf` liefern schließlich auch eine gewisse Konfidenz, so dass das beschriebene Problem bestätigt werden konnte. Ausgewählte Ergebnisse der `iperf`-Tests finden sich im Anhang.
- Der modifizierte Tribler-Client liefert detaillierte Log-Ausgaben, die bei der Sichtung einige grundsätzliche Probleme mit dem Netzwerk aufzeigen. Auch diese Probleme sind reproduzierbar, allerdings nicht deterministisch.
    1. **Name or service not known**: Dieser Fehler trat auf, wenn ein Peer eine HTTP-Anfrage an den Tracker geschickt hat. Der Tracker lief zu den entsprechenden Zeitpunkten und zeigte normales Antwortverhalten.
    2. **Could not bind a UDP socket on port 6700 (Address already in use)**: Dieser Fehler zeigte sich in einem Szenario, in dem Clients Ports von 9000 aufwärts an benutzen und Supporter-Server auf den Ports 5000, 5002, 5004 und 5006 horchten. Bei der Verbindungsherstellung nutzt der Initiator anonyme Ports, allerdings liegen diese normalerweise in einem deutlich höheren Port-Bereich und werden zufällig gewählt, so dass Doppelbelegungen pro Maschine sehr unwahrscheinlich sind.
    3. **Temporary failure in name resolution**: Dieser Fehler zeigte sich beim Verbindungsmanagement der NextShare-Software. Hier konnte beobachtet werden, dass nach diesem Fehler einzelne Peers im System verweilen und sich nicht mehr korrekt beenden.

Die angeführten Probleme lassen den Schluss zu, dass es sich um grundlegende Probleme mit dem Netzwerk handelt. Auszuschließen sind andere Fehler, die bspw. bei der NextShare-Software, jedoch nicht. Die Probleme sind unabhängig von der Ankunftsrate der Peers aufgetreten, so dass Überlastsituationen durch die Konfiguration der Experimente auszuschließen sind. Bei der Datenerhebung musste auf die Testläufe zurückgegriffen werden, die stabil durchgelaufen sind. Dies erschwerte die Ausarbeitung einer aussagekräftigen Evaluation stark, da zum Einen deutlich mehr Zeit in die Datenerhebung investiert werden musste, zum Anderen eine genaue Überprüfung der Log-Dateien für einzelne Experimente notwendig war, um durch G-Lab induzierte Fehler aus den Statistiken zu isolieren.

C UML-Diagramme

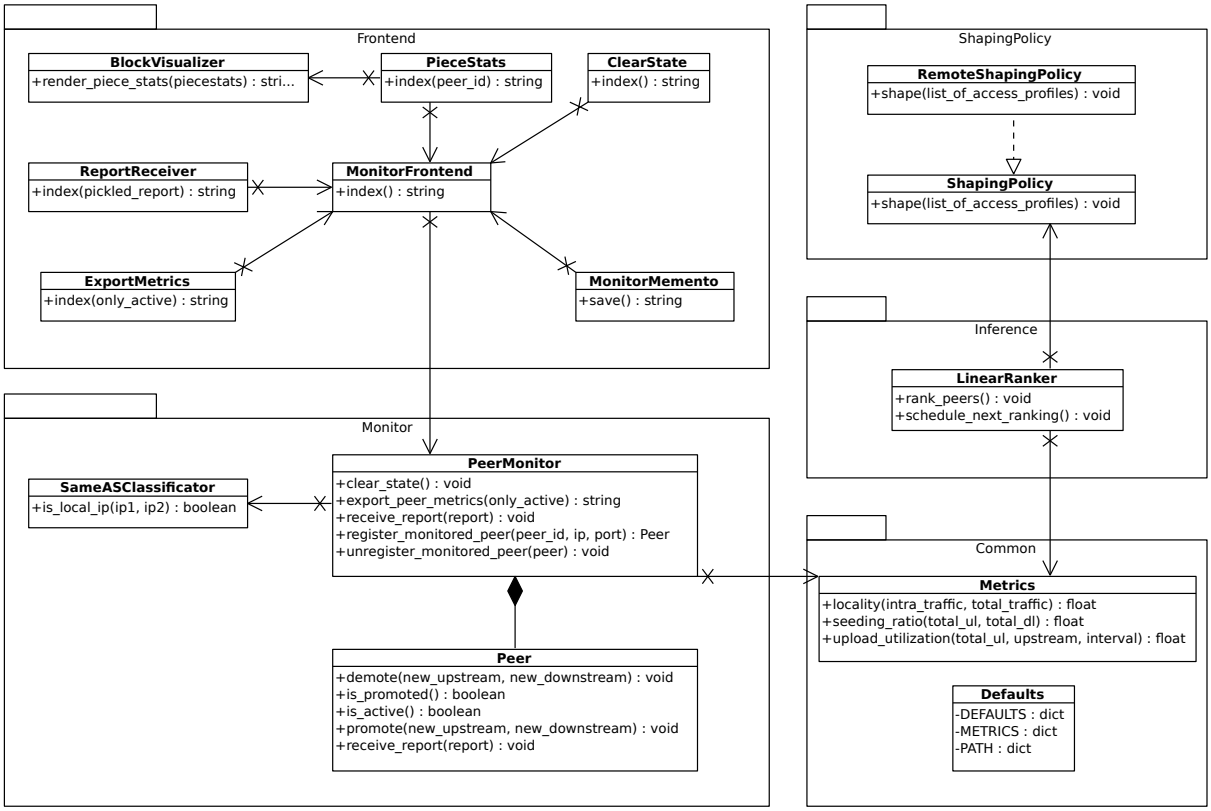


Abbildung C.1: Klassendiagramm des adaptiven Bandbreitenmanagements (Detailansicht)

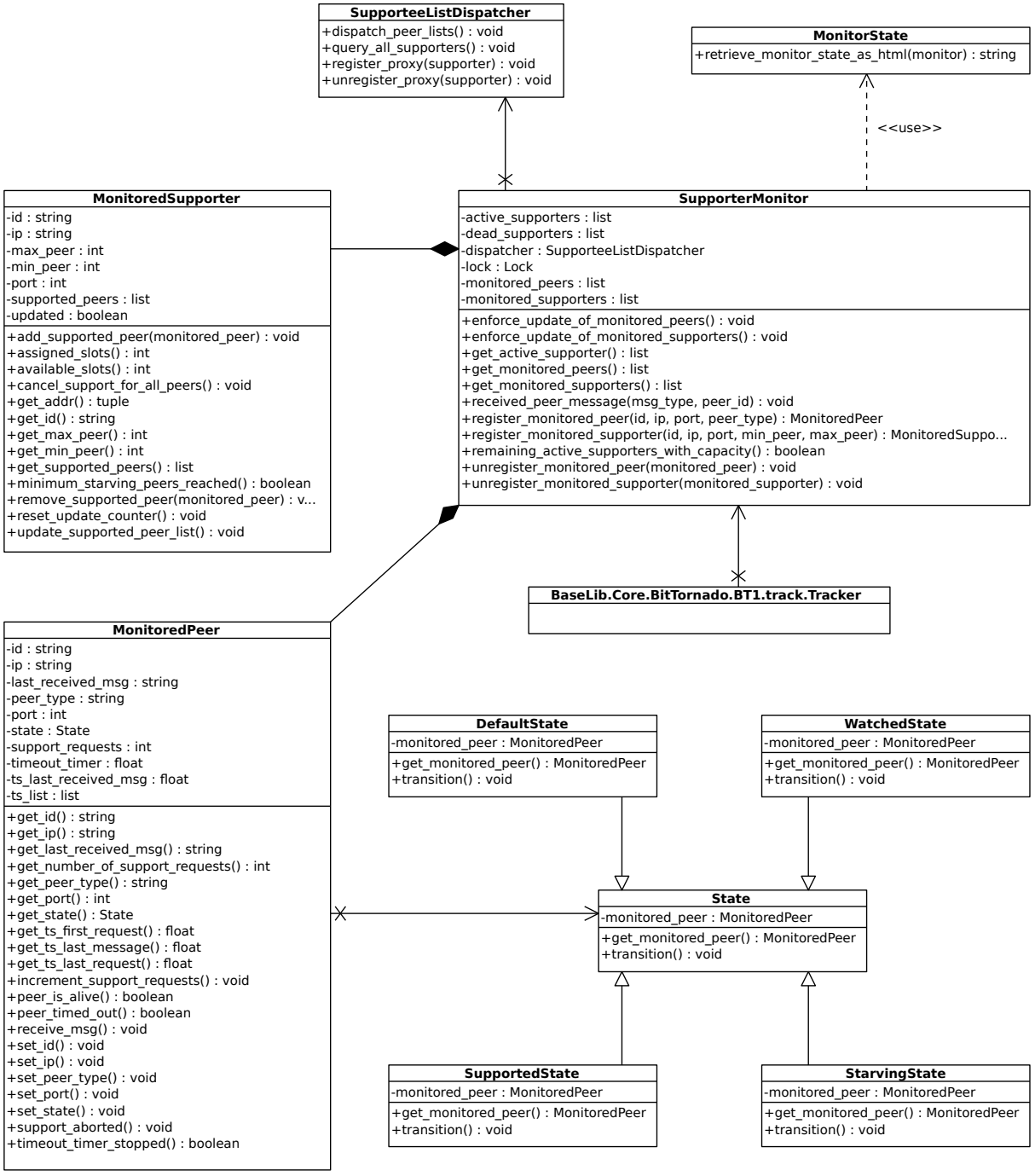


Abbildung C.2: Klassendiagramm des Moduls `SupporterMonitor` (Detailansicht)

D Client-seitige Statistiken

Allgemeine Statistiken

Schlüssel	Beschreibung
down_rate	Download-Rate für den zugehörigen Download
infohash	Infohash des zugehörigen Torrents in hexadezimaler Schreibweise
peer_id	Die Schwarm-bezogene ID des Peers, der die Statistiken sendet
peer_ip	Die externe IP-Adresse des Peers, sofern diese bekannt ist
peer_port	Der Listen Port des Peers
progress	Fortschritt des zugehörigen Downloads
status	Status des zugehörigen Downloads
timestamp	Zeitpunkt des Versendens auf Client-Seite
up_rate	Upload-Rate für den zugehörigen Download

Tabelle D.1: Übersicht über allgemeine Statistiken, die client-seitig aggregiert werden

Video-bezogene Statistiken

Schlüssel	Beschreibung
video_played	Anzahl Chunks, die bereits abgespielt wurden
video_stalled	Gesamtzeit an Unterbrechungen der Video-Wiedergabe
video_late	Anzahl Chunks, die der Peer zu spät erhalten hat
video_dropped	Anzahl Chunks, die vom Peer verworfen wurden
video_playback_started_delay	Verzögerung in Sekunden, von Starten des Downloads bis zum Beginn der Video-Wiedergabe

Tabelle D.2: Übersicht über Statistiken bzgl. der Wiedergabe des Videos

Datenblock-bezogene Statistiken

Schlüssel	Beschreibung
pieces_high_range	Aktuelle Grenzen des High-Priority Sets in Intervallangabe
pieces_missing_from_high_range	Anzahl fehlender Chunks über dem aktuellen High-Priority Set
pieces_playback_position	Gibt den Chunk an, der gerade abgespielt wird
pieces_stats	Statistiken über heruntergeladene Chunks (Zeitpunkt, Quelle)
pieces_total_amount	Gesamtzahl an Chunks

Tabelle D.3: Übersicht über Statistiken bzgl. bereits heruntergeladener oder fehlender Datenblöcke

Nachbarschafts-bezogene Statistiken

Schlüssel	Beschreibung
neighbours	Python Dictionary mit Statistiken der aktuellen Nachbarschaft

Tabelle D.4: Übersicht über client-seitige Statistiken bezüglich der aktuellen aktiven Nachbarschaft des Peers

Literaturverzeichnis

[1] *Cisco Inc. - Cisco Visual Networking Index: Forecast and Methodology, 2009–2014, White Paper, 2009 (Abruf: 30. November 2010)*. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[2] Chua, K. ; Cowan, C.: *Internet VoD: Meeting Consumer Demands, White Paper, 2010 (Abruf: 30. November 2010)*. http://www.valuepartners.com/VP_pubbl_pdf/PDF_Comunicati/Media%20e%20Eventi/2010/value-partners-Internet-VOD-Meeting-Consumer-Demands-Cowan-Chua.pdf

[3] Arrington, M.: *Google Relies On Akamai To Stream YouTube Live; 700,000 Concurrent Viewers (Abruf: 30. November 2010)*. http://techcrunch.com/2008/11/22/google-relies-on-akamai-to-stream-youtube-live-700000-concurrent-viewers

[4] Berners-Lee, T.: *Net Neutrality: This is serious (Abruf: 30. November 2010)*. http://dig.csail.mit.edu/breadcrumbs/node/144

[5] Steinmetz, R. (Hrsg.) ; Wehrle, K. (Hrsg.): *Lecture Notes in Computer Science*. Bd. 3485: *Peer-to-Peer Systems and Applications*. Springer Berlin / Heidelberg, 2005

[6] Karagiannis, T. ; Rodriguez, P. ; Papagiannaki, K.: Should Internet Service Providers Fear Peer-Assisted Content Distribution. In: *Proceedings of the 5th Conference on Internet Measurement*, USENIX Association, 2005, S. 63–76

[7] Bindal, R. ; Cao, P. ; Chan, W. ; Medved, J. ; Suwala, G. ; Bates, T. ; Zhang, A.: Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In: *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems* Bd. 06, 2006, S. 66–75

[8] Oechsner, S. ; Lehrieder, F. ; Hossfeld, T. ; Metzger, F. ; Pussep, K. ; Staehle, D.: Pushing the Performance of Biased Neighbor Selection through Biased Unchoking. In: *9th International Conference on Peer-to-Peer Computing*, 2009, S. 301–310

[9] Gerlach, F.: *Adaptive Ressourcenzuteilung für Dienstgütegarantien in Peer-unterstützten Video-on-Demand-Systemen*, TU Darmstadt, Fachgebiet Multimedia Kommunikation, Masterarbeit, 2010. – 85 S.

[10] Pussep, K. ; Abboud, O. ; Gerlach, F. ; Steinmetz, R. ; Strufe, T.: Adaptive Server Allocation for Peer-Assisted Video-on-Demand. In: *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, IEEE Computer Society, 2010, S. 1–8

[11] Pathan, A. M. K. ; Buyya, R.: A Taxonomy and Survey of Content Delivery Networks. In: *Grid Computing and Distributed Systems (GRIDS) Laboratory* (2006), S. 1–44

[12] Su, A.-J. ; Choffnes, D. R. ; Kuzmanovic, A. ; Bustamante, F. E.: Drafting Behind Akamai (Travelocity-Based Detouring). In: *Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ACM Press, 2006, S. 435–446

[13] Cohen, B.: Incentives Build Robustness in BitTorrent. In: *Proceedings of the Workshop on Economics of Peer-to-Peer Systems* (2003)

[14] *BitTorrent Protocol Specification v1.0 (Abruf: 30. November 2010)*. http://wiki.theory.org/BitTorrentSpecification

[15] Legout, A. ; Urvoy-Keller, G. ; Michiardi, P.: Rarest First and Choke Algorithms Are Enough. In: *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, ACM Press, 2006, S. 203–216

[16] Cohen, B.: *The BitTorrent Protocol Specification (Abruf: 30. November 2010)*. http://bittorrent.org/beps/bep_0003.html

[17] Liu, Y. ; Guo, Y. ; Liang, C.: A Survey on Peer-to-Peer Video Streaming Systems. In: *Peer-to-Peer Networking and Applications* (2008), S. 18–28

[18] Castro, M. ; Druschel, P. ; Kermarrec, A.-M. ; Nandi, A. ; Rowstron, A. ; Singh, A.: SplitStream: High-Bandwidth Content Distribution in Cooperative Environments. In: *Peer-to-Peer Systems II, Second International Workshop* Bd. 37, Springer Berlin / Heidelberg, 2003, S. 292–303

[19] Kostić, D. ; Rodriguez, A. ; Albrecht, J. ; Vahdat, A.: Bullet: High Bandwidth Data Dissemination Using An Overlay Mesh. In: *Proceedings of the 19th ACM Symposium on Operating Systems Principles* Bd. 37, ACM Press, 2003, S. 282

[20] Guo, Y. ; Suh, K. ; Kurose, J. ; Towsley, D.: P2Cast: Peer-to-Peer Patching Scheme For VoD Service. In: *Proceedings of the 12th International Conference on World Wide Web*, ACM Press, 2003, S. 301–309

[21] Hua, K. A. ; Cai, Y. ; Sheu, S.: Patching: A Multicast Technique For True Video-on-Demand Services. In: *Proceedings of the 6th ACM International Conference on Multimedia*, ACM Press, 1998, S. 191–200

[22] Gao, L. ; Towsley, D.: Threshold-based Multicast for Continuous Media Delivery. In: *IEEE Transactions on Multimedia* (2001), S. 405–414

[23] Dana, C. ; Li, D. ; Harrison, D. ; Chuah, C.-N.: BASS: BitTorrent Assisted Streaming System for Video-on-Demand. In: *IEEE 7th Workshop on Multimedia Signal Processing*, 2005, S. 1–4

[24] Guo, Y. ; Mathur, S. ; Ramaswamy, K. ; Yu, S. ; Patel, B.: *Ponder: Providing Commercial-Quality Video-on-Demand Service Using Peer-to-Peer Networks*

[25] Annapureddy, S. ; Guha, S. ; Gkantsidis, C. ; Gunawardena, D. ; Rodriguez, P.: Is High-Quality VoD Feasible Using P2P Swarming? In: *Proceedings of the 16th International Conference on World Wide Web*, ACM Press, 2007, S. 903–912

[26] Abboud, O. ; Pussep, K. ; Müller, M. ; Kovacevic, A. ; Steinmetz, R.: Advanced Prefetching and Upload Strategies for P2P Video-on-Demand. In: *Proceedings of the ACM Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking*, ACM Press, 2010, S. 31–36

[27] Magharei, N. ; Reza, R.: Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. In: *Proceedings of the 26th IEEE International Conference on Computer Communications*, 2007, S. 1424–1432

[28] Mol, J. J. D. ; Pouwelse, J. A. ; Meulpolder, M. ; Epema, D. H. J. ; Sips, H. J.: Give-to-Get: Free-Riding-Resilient Video-on-Demand in P2P Systems. In: *Proceedings of the 15th SPIE/ACM Multimedia Computing and Networking*, 2008

[29] Gross, G.: *Comcast Sets Monthly Bandwidth Limit for Customers (Abruf: 30. November 2010)*. http://www.cio.com/article/446674/Comcast_Sets_Monthly_Bandwidth_Limit_for_Customers

[30] Eckersley, P.: *Comcast is also Jamming Gnutella (and Lotus Notes?) (Abruf: 30. November 2010)*. http://www.eff.org/deeplinks/2007/10/comcast-also-jamming-gnutella-and-lotus-notes

[31] *Bad ISPs (Abruf: 30. November 2010)*. http://wiki.vuze.com/w/Bad_ISPs

[32] *Sandvine Inc. - Intelligent Broadband Networks (Abruf: 30. November 2010)*. http://www.sandvine.com

[33] *Packeteer Inc. - Packeteer Product Overview - Innovative Bandwidth Farming, White Paper, 2001 (Abruf: 30. November 2010)*. https://bto.bluecoat.com/packetguide/5.2.1/documents/PSISPOverview.pdf

[34] Packeteer Inc. - TCP Rate Control and Alternatives, White Paper, 2002 (Abruf: 30. November 2010). http://www.icann.org/en/tlds/org/applications/unity/appendices/pdfs/packeteer/TcpRateControl.pdf

[35] Eckersley, P. ; Lohmann, F. von ; Schoen, S.: *Packet Forgery By ISPs: A Report on the Comcast Affair (Abruf: 30. November 2010)*. http://www.eff.org/wp/packet-forgery-isps-report-comcast-affair

[36] Dischinger, M. ; Mislove, A. ; Haeberlen, A. ; Gummadi, K. P.: Detecting BitTorrent Blocking. In: *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, ACM Press, 2008, S. 3–8

[37] Wang, J. ; Huang, C. ; Li, J.: On ISP-Friendly Rate Allocation for Peer-Assisted VoD. In: *Proceedings of the 16th ACM International Conference on Multimedia*, ACM Press, 2008, S. 279–288

[38] Lehrieder, F. ; Hossfeld, T. ; Oechsner, S. ; Singeorzan, V.: The Impact of Caching on BitTorrent-Like Peer-to-Peer Systems. In: *IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, IEEE Computer Society, 2010

[39] Cai, Y. ; Chen, Z. ; Tavanapong, W.: Caching Collaboration and Cache Allocation in Peer-to-Peer Video Systems. In: *Multimedia Tools and Applications* (2007), S. 117–134

[40] Dán, G.: Cooperative Caching and Relaying Strategies for Peer-to-Peer Content Delivery. In: *Proceedings of the 7th International Conference on Peer-to-Peer Systems*, USENIX Association, 2008, S. 16–21

[41] Papafili, I. ; Soursos, S. ; Stamoulis, G.: Improvement of BitTorrent Performance and Inter-Domain Traffic by Inserting ISP-Owned Peers. In: *Network Economics for Next Generation Networks, 6th International Workshop on Internet Charging and Qos Technologies*, Springer Berlin / Heidelberg, 2009, S. 97–108

[42] Pussep, K. ; Kuleshov, S. ; Gross, C. ; Soursos, S.: An Incentive-Based Approach to Traffic Management for Peer-to-Peer Overlays. In: Stiller, B. (Hrsg.) ; Hossfeld, T. (Hrsg.) ; Stamoulis, G. (Hrsg.): *Incentives, Overlays, and Economic Traffic Control* Bd. 6236. Springer Berlin / Heidelberg, 2010, S. 2–13

[43] Keralapura, R. ; Taft, N. ; Chuah, C.N. ; Iannaccone, G.: Can ISPs Take the Heat From Overlay Networks. (2004)

[44] Seedorf, J. ; Kiesel, S. ; Stiemerling, M.: Traffic Localization for P2P-Applications: The ALTO Approach. In: *Proceedings P2P 2009, Ninth International Conference on Peer-to-Peer Computing*, IEEE Computer Society, 2009, S. 171–177

[45] *MaxMind - GeoIP: IP Address Location Technology (Abruf: 30. November 2010)*. http://www.maxmind.com/app/ip-location

[46] Aggarwal, V. ; Feldmann, A. ; Scheideler, C.: Can ISPS and P2P Users Cooperate for Improved Performance? In: *SIGCOMM Computer Communication Review* 37 (2007), S. 29–40

[47] Xie, H. ; Yang, Y. R. ; Krishnamurthy, A. ; Liu, Y. ; Silberschatz, A.: P4P: Provider Portal for Applications. In: *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* Bd. 38, ACM Press, 2008, S. 351–362

[48] Soursos, S. ; Pussep, K. ; Racz, P. ; Spirou, S. ; Stamoulis, G. D. ; Stiller, B.: ETMS: A System for Economic Management of Overlay Traffic. (2010), S. 1–10

[49] Griffiths, C. ; Livingood, J. ; Popkin, L. ; Woundy, R. ; Yang, Y.: *RFC 5632: Comcast's ISP Experiences in a Proactive Network Provider Participation for P2P (P4P) Technical Trial (Abruf: 30. November 2010)*. http://www.ietf.org/rfc/rfc5632.txt

[50] *The SmoothIT Project (Abruf: 30. November 2010)*. http://www.smoothit.org

[51] Racz, P. ; Oechsner, S. ; Lehrieder, F.: BGP-Based Locality Promotion for P2P Applications. In: *Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN)*, IEEE Computer Society, 2010, S. 1–8

[52] Choffnes, D. R. ; Bustamante, F. E.: Taming the Torrent. In: *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ACM Press, 2008, S. 363–374

[53] Piatek, M. ; Madhyastha, H. V. ; John, J. P. ; Krishnamurthy, A. ; Anderson, T.: Pitfalls for ISP-Friendly P2P Design. (2009)

[54] Hossfeld, T. ; Hock, D. ; Oechsner, S. ; Lehrieder, F. ; Despotovic, Z. ; Kellerer, W. ; Michel, M.: Measurement of BitTorrent Swarms and their AS Topologies / University of Würzburg. 2009. – Forschungsbericht

[55] Blond, S. L. ; Legout, A. ; Dabbous, W.: Pushing BitTorrent Locality to the Limit. (2008)

[56] *NGN Working Definition (Abruf: 30. November 2010)*. http://www.itu.int/ITU-T/studygroups/com13/ngn2004/working_definition.html

[57] Pussep, K. ; Gross, C. ; Oechsner, S. ; Makidis, M. ; Racz, P. ; Rodriguez, M. A. C. ; Michel, M. ; Despotovic, Z. ; Soursos, S. ; E., Agiatzidou ; Papafili, I. ; Stamoulis, G. ; Cholda, P. ; Dulinski, Z. ; Kantor, M. ; Stankiewicz, R.: *ETM Models and Components and Theoretical Foundations (Abruf: 30. November 2010)*. http://www.smoothit.org/index.php?eID=tx_nawsecuredl&u=0&file=fileadmin/public_deliverables/D2.3-v1.0-final.pdf&t=1291114454&hash=026678eb57e37b66f038c118eff3d2f8

[58] Tran-gia, P. ; Berlin, T. U. ; Feldmann, A. ; Steinmetz, R. ; Zitterbart, M. ; Schotten, H.: *G-Lab Whitepaper Phase 1 (Abruf: 30. November 2010)*. http://www.german-lab.de/fileadmin/Press/G-Lab_White_Paper_Phase1.pdf

[59] Yu, H. ; Zheng, D. ; Zhao, B. Y. ; Zheng, W.: Understanding User Behavior in Large-Scale Video-on-Demand Systems. In: *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems*, ACM Press, 2006, S. 333–344

[60] Li, B. ; Xie, S. ; Qu, Y. ; Keung, G. Y. ; Lin, C. ; Liu, J. ; Zhang, X.: Inside the New Coolstreaming: Principles, Measurements and Performance Implications. In: *Proceedings of the 27th Conference on Computer Communications*, IEEE Computer Society, 2008, S. 1031–1039

[61] Almesberger, W.: Linux Traffic Control - Next Generation. In: *Proceedings of the 9th International Linux System Technology Conference*, 2002, S. 95–103

[62] Hubert, B. ; Graf, T. ; Maxwell, G. ; Mook, R. van ; Oosterhout, M. van ; Schroeder, P. ; Spaans, J. ; Larroy, P.: *Linux Advanced Routing & Traffic Control (Abruf: 30. November 2010)*. http://lartc.org

[63] Boxman, J.: *Installing and Using tcng under Debian GNU/Linux (Abruf: 30. November 2010)*. http://blog.edseek.com/~jasonb/articles/tcng_shaping.html

[64] Gasmi, T.: Traffic Shaping unter Berücksichtigung von TCP Eigenschaften. 2005. – Forschungsbericht. – 67 S.

[65] Cuevas, R. ; Laoutaris, N. ; Yang, X. ; Siganos, G. ; Rodriguez, P.: Deep Diving into BitTorrent Locality. In: *Proceedings of the 2010 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2010, S. 349–350

[66] Al-Hamra, A. ; Legout, A. ; Barakat, C. ; Antipolis, S.: Understanding the Properties of the BitTorrent Overlay. 2007. – Forschungsbericht. – 18 S.

[67] Hasslinger, G.: ISP Platforms Under a Heavy Peer-to-Peer Workload. In: *Peer-to-Peer Systems and Applications* Bd. 3485. Springer Berlin / Heidelberg, 2005, Kapitel 22, S. 369–381

[68] Fujita, Y. ; Yoshida, J. ; Yoshida, K. ; Tsuda, K.: Peer-Assisted Media Streaming: A Holistic Review. In: *Knowledge-Based Intelligent Information and Engineering Systems*, Springer Berlin / Heidelberg, 2010, S. 317–340

[69] Saleh, O. ; Hefeeda, M.: Modeling and Caching of Peer-to-Peer Traffic. In: *Proceedings of the 14th IEEE International Conference on Network Protocols*, IEEE Computer Society, 2006, S. 249–258

[70] Li, J.: On Peer-to-Peer (P2P) Content Delivery. In: *Peer-to-Peer Networking and Applications* 1 (2008), Nr. 1, S. 45–63

[71] Picconi, F. ; Massoulie, L.: ISP - Friend or Foe? Making P2P Live Streaming ISP-Aware. In: *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, IEEE Computer Society, 2009, S. 413–422

[72] Souza, L. ; Cores, F. ; Yang, X. ; Ripoll, A.: DynaPeer: A Dynamic Peer-to-Peer Based Delivery Scheme for VoD Systems. In: Kermarrec, A.-M. (Hrsg.) ; Bougé, L. (Hrsg.) ; Priol, T. (Hrsg.): *Euro-Par 2007 Parallel Processing* Bd. 4641. Springer Berlin / Heidelberg, 2007, S. 769–781

[73] Ching, Y. C. ; Hsu, C.-H. ; Li, K. C.: On Improving Network Locality in BitTorrent-Like Systems. In: *Scalable Information Systems*. Springer Berlin / Heidelberg, 2009 (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), S. 58–75

[74] Kaya, A. O. ; Chiang, M. ; Trappe, W.: P2P-ISP Cooperation: Risks and Mitigation in Multiple-ISP Networks. In: *Proceedings of the Global Communications Conference*, IEEE Computer Society, 2009, S. 1–8

[75] Sam, C. M. ; John, L. ; Lui, C. S.: On the Interaction and Competition among Internet Service Providers. In: *IEEE Journal on Selected Areas in Communications* 26 (2009), Nr. 7, S. 1277–1283

[76] Liao, W. ; Papadopoulos, F. ; Psounis, K.: Performance Analysis of BitTorrent-Like Systems with Heterogeneous Users. In: *Performance Evaluation* 64 (2007), S. 876–891

[77] Norton, W. B.: Interconnection Strategies for ISPs. In: *NANOG - The North American Network Operators Group* (1999)

[78] Mills, E.: *Google Entering Video-on-Demand Business (Abruf: 30. November 2010)*. http://news.cnet.com/2100-1025_3-6021998.html

[79] *Cisco Inc. - Approaching the Zettabyte Era, White Paper, 2008 (Abruf: 30. November 2010)*. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481374_ns827_Networking_Solutions_White_Paper.html

[80] Kaune, S. ; Lauinger, T. ; Kovacevic, A. ; Pussep, K.: Embracing The Peer Next Door: Proximity in Kademlia. In: *Proceedings of the Eighth International Conference on Peer-to-Peer Computing*, IEEE Computer Society, 2008, S. 343–350

[81] Guo, L. ; Chen, S. ; Xiao, Z. ; Tan, E. ; Ding, X. ; Zhang, X.: Measurements, Analysis, and Modeling of BitTorrent-Like Systems. In: *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, ACM Press, 2005, S. 35–48

[82] Legout, A. ; Liogkas, N. ; Kohler, E. ; Zhang, L.: Clustering and Sharing Incentives in BitTorrent Systems. In: *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ACM Press, 2007, S. 301–312