Ausarbeitung AW 2 - WsSe 2011/12

Sebastian Zagaria

Pull-Multicast für HAMcast

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einleitung

Heutzutage spielen internetbasierte Anwendungen eine immer wichtigere Rolle im Alltag der Menschen. Zu den beliebtesten Internetdiensten zählen Video- und Audio-Konferenzen, IPTV, Chatprogramme und Filesharing. Alle diese Anwendungen basieren auf einer Art Gruppenkommunikation, bei dem die Daten von einer Quelle an eine Vielzahl von Empfängern übertragen werden muss. Kommunikationskonzepte wie die Client-Server Architektur bieten eine Punkt-zu-Punkt Verbindung zwischen einem Sender und einem Empfänger. Dieses Architekturkonzept ist für eine Gruppenkommunikation nicht geeignet. Vielmehr wird ein Kommunikationsdienst benötigt, der eine Verteilung von Daten an viele Benutzer gleichzeitig erlaubt. Das Architekturkonzept von Publish/Subscribe beschreibt genau solch eine Art der Gruppenkommunikation. Bei dem Publish/Subscribe unterscheidet man zwischen zwei Arten von Benutzern, den Publisher und den Subscriber. Der Sender einer Nachricht (Publisher) schickt Daten an eine ihm unbekannte Anzahl an Empfänger. Die Empfänger (Subscriber) können ihr Interessen an Daten bekunden. Bei diesem Entwurfsmuster müssen sich weder Sender noch Empfänger untereinander kennen. Die Daten werden beim Verteilen an eine Gruppe adressiert und an alle Empfänger können ihr Interesse über den beitritt in diese Gruppe bekunden. Die konkrete Verteilung der Daten an die einzelnen Empfänger wird über einen Mechanismus im Netzwerk durchgeführt. Ein Kommunikationsprotokoll was das Architekturkonzept von Publish/Subscribe verwendet ist Multicast.

Multicast existiert generell in zwei Formen. Als natives Kommunikationsprotokoll unterstützt durch IPv4, IPv6 und Application-Layer-Multicast (ALM), Multicast Algorithmen auf Anwendungsebene. Die Entwicklung von Application-Layer-Multicast Algorithmen [6] ist darauf zurückzuführen, dass die Verbreitung von Routern im Internet, die über Multicastrouting verfügen, nur eingeschränkt vorhanden ist. Ein Nachteil von ALM ist, dass es nicht so effizient wie das native Multicast arbeiten kann. Neuste Entwicklungen versuchen die Vorteile von ALM und nativen Multicast miteinander zu kombinieren. Bei diesem Ansatz wird nativ Multicast überall dort eingesetzt, wo er verfügbar ist und über ALM miteinander verbunden. ALM wird dabei meist als Brücke zwischen nativen Multicast Netzwerken genutzt. Diese Ansätze werden als hybrider Multicast bezeichnet.

Im Bereich des Filesharing und einiger Kommerzieller IPTV Anwendungen haben sich unstrukturierte Chunk-Trading Netzwerke durchgesetzt. Als bekanntestes Beispiel für ein Chunk-Trading Netzwerk gilt BitTorrent. Der unstrukturierte Aufbau dieser Netzwerke verringert den Managementaufwand und die Komplexität, was diese Netzwerke sehr attraktiv gestaltet. Durch das Chunk-Trading wird zusätzlich die Bandbreitenausnutzung gegenüber den Push Ansätzen erhöht.

1.1 Motivation

Ziel der Masterarbeit ist die Entwicklung und Implementation eines Pull-Multicast, der Daten über ein Chunk-Trading austauscht. Pull-Netzwerke fallen heutzutage nicht unter den Begriff der Gruppenkommunikation. Es soll gezeigt werden das Pull-Netzwerke dem Entwurfsmuster des Publish/Subscribe entsprechen können. Des Weiterem soll die Performance und Skalierbarkeit mit anderen Multicast-Technologien verglichen werden. Die Implementation des Pull-Multicast wird als HAMcast Technologie Modul umgesetzt.

Das HAMcast Projekt [7] stellt eine universelle und technologieunabhängige hybride Multicast-API zur Verfügung [**?**]. Ziel des Projektes ist es dem Anwendungsentwickler eine API zur Verfügung zu stellen, die es ermöglicht Gruppenkommunikations-Software unabhängig von der verwendeten Technologie zu entwickeln.

1.2 Inhalt der Ausarbeitung

Inhalt dieser Ausarbeitung ist die Vorstellung verwandter Arbeiten, sowie deren Relevanz für das eigene Projekt. Als Erstes wird das HAMcast Projekt vorgestellt, auf dem diese Arbeit basiert. Danach wird das Chunk-Trading beschrieben und den Push-Verfahren gegenübergestellt. Anschließend folgt die Vorstellung von GRAPLE eine generische API zur Erstellung von Mesh-Netzwerken. Darauf folgend werden Verfahren vorgestellt, die es ermöglichen zeitsensitive Daten über ein Chunk-Trading Netzwerk zu verteilen. Insbesondere wird der BiToS Algorithmus für das Verteilen von Video-Streams über ein Mesh-Netzwerk vorgestellt. Abschließend wird ein Fazit über die Verwandten Arbeiten gegeben und deren Relevanz für die Arbeit eingeschätzt. Zum Schluss folgt ein Ausblick auf die zweite Projektarbeit.

2 Verwandte Arbeiten

2.1 Hybride-Multicast Netzwerke

Die Idee einer hybriden Multicast-Architektur ist es, isolierte Multicast-Netzwerke sogenannte Multicast-Inseln, über deren Grenzen hinaus miteinander zu verbinden. Das Ziel dieses Ansatzes ist es einen Multicast Dienst zur Verfügung zu stellen, der die Vorteile von IP-Multicast mit denen von Application-Layer-Multicast kombiniert.

Der Begriff Multicast-Inseln bezeichnet eine Netzwerk Domäne mit einer Multicast-Technologie und mit beliebiger Größe. Die Grenzen dieser Inseln werden durch den Bereich gekennzeichnet, indem eine Multicast-Nachricht verschickt werden kann.

Hybride Multicast-Architekturen verfügen meistens über zwei Ebenen. In der ersten Ebene befinden sich die Multicast-Inseln. In der zweiten Ebene werden diese Multicast-Inseln über Edge-Nodes miteinander verbunden. Edge-Nodes bezeichnen knoten die den Datenverkehr zwischen den Multicast-Inseln weiterleiten. Diese Edge-Nodes sind meistens über einen ALM miteinander verbunden.

Architekturen, die einen solchen oder ähnlichen Ansatz vertreten, sind u.a. „Scalable and Backbone Topology-Aware Hybrid Multicast“ (SHM) [5], „Universal IP multicast delivery“ (UM) [11], „Island Multicast: Combining IP Multicast With Overlay Data Distribution“ (IM) [4] und "Hybrid Adaptive Mobile Multicast"(HAMcast) [7].

2.1.1 HAMcast

Ziele von HAMcast

Zum jetzigen Zeitpunkt gibt es keine einheitliche Programmierschnittstelle, die einen Gruppenkommunikations-Dienst von der Technologie abstrahiert. Somit ist es den Softwareentwicklern überlassen festzulegen, welche Technologie die Anwendung verwendet. Daraus ergibt sich, dass heutige Gruppenkommunikations-Anwendungen mit zwei Problemen zu kämpfen haben. Entweder sind sie abhängig von der Technologie und funktionieren nur unter bestimmten Netzwerk-Voraussetzungen oder es werden Leistungseinbußen in Kauf genommen, um die Anwendung für eine breitere Benutzergruppe zur Verfügung stellen zu können.

Das HAMcast Projekt [7] stellt einen universellen Multicast bereit, ohne dabei Veränderungen an vorhandenen Technologien und Protokollen vornehmen zu müssen. Vielmehr werden die bereits vorhandenen Technologien und Protokolle in ihrer Funktionalität durch HAMcast erweitert. Die HAMcast-Architektur bietet Funktionalitäten, die es dem Softwareentwickler ermöglichen Technologie unabhängige Software zu entwickeln.

Hybrider Multicast mit HAMcast

Um einen technologieunabhängigen Dienst zur Verfügung stellen zu können, muss die Adressierung und Namensgebung von der verwendeten Technologie abstrahiert werden. HAMcast basiert auf dem Identifier-Locator-Split, d.h., Gruppennamen werden von Technologie abhängigen Adressen separiert. HAMcast-Anwendungen verwenden Gruppennamen zur Identifikation der Gruppe. Durch diese Namen kann die verwendete Technologie für das Verteilen der Daten vor der Anwendung verborgen werden.

Ein Gruppenname wird durch eine HAMcast uri repräsentiert.

scheme :// group @ instantiation : port / sec-credentials

Das „scheme“ bezeichnet den Namensraum, z.B. ip, sip oder scribe. „group“ gibt die Gruppennamen an. „instantiation“ identifiziert die Entität, die eine Instanz der Gruppe generiert, z.B. bei SSM. „port“ identifiziert eine bestimmte Applikation und sec-credentials ist optional für die Implementierung von Sicherheitsmechanismen. Zulässige Gruppen IDs sind z.B. ip://239.0.0.1:1234, scribe://239.0.0.1:1234 oder sip://snoopy@peanuts.com. Durch ein Late-Binding der HAMcast uri zur Laufzeit wird es dem Entwickler ermöglicht, Technologie unabhängige Anwendungen zu entwickeln.

Um verschiedene Technologien miteinander verbinden zu können, verwendet HAMcast Inter Domain Gateways (IMG). Ein IMG leitet hierfür den Datenstrom aus einem Teilnetzwerk in ein anderes weiter, er dient als Brücke zwischen zwei Netzwerken. In der Abbildung 1 wird gezeigt wie verschiedene Netzwerke durch IMGs miteinander verbunden werden können. Verschiedene Multicast Netzwerke, dargestellt durch Wolken, sind logisch oder auch Technologisch voneinander getrennt.

Durch Platzierung von IMGs können die verschiedenen Netzwerke zu einem einzigen HAMcast Netzwerk zusammengefasst werden. Dabei können sich die IMGs in einem eigenen Application-Layer-Multicast Netzwerk organisieren. Nach Empfang von Daten leitet ein IMG diese dann über einen ALM an die anderen teilnehmenden IMGs weiter. Diese wiederum können die Daten dann an das lokale Netzwerk weiterleiten.

Um auf den Multicast-Dienst von HAMcast zugreifen zu können, wird als Teil des HAMcast Projektes eine API zur Verfügung gestellt, die „common multicast API“ [9]. Diese API wird durch eine Middleware und Library realisiert. Die unterstützten Multicast-Technologien sind in Module aufgeteilt, die von der Middleware verwaltet werden. Diese Aufteilung ermöglicht eine hohe Erweiterbarkeit, Anpassung und Flexibilität der Middleware. So können neue Technologien hinzugefügt werden, ohne eine Veränderung der Anwendung. Die Anwendungen kommunizieren über einen Socket-Stub mit der Middleware, der Socket-Stub ist mittels Inter-Prozess-Kommunikation (IPC) mit der Middleware verbunden. Zum Ausführen des Multicast-Dienstes muss eine Instanz der Middleware auf jedem Host, der den Dient nutzen möchte, initiiert werden. Die Abbildung 2 zeigt den Aufbau der Middleware.

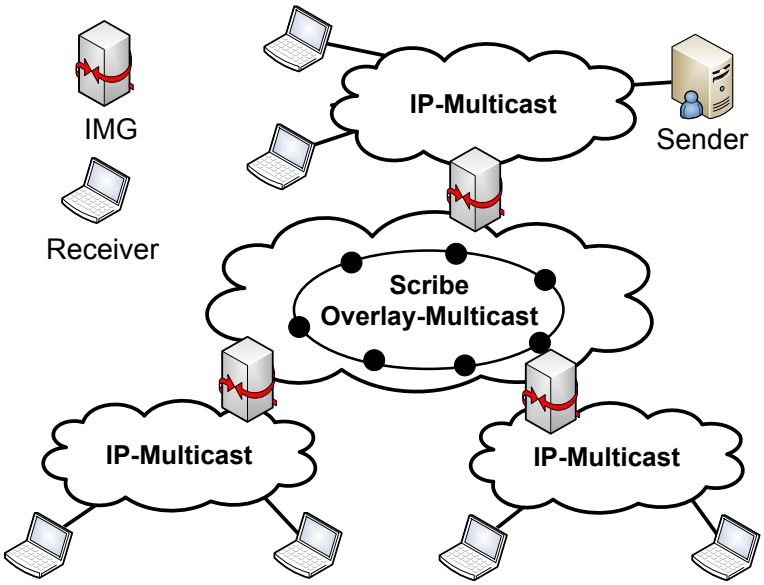

Abbildung 1: Einsatz von IMG´s

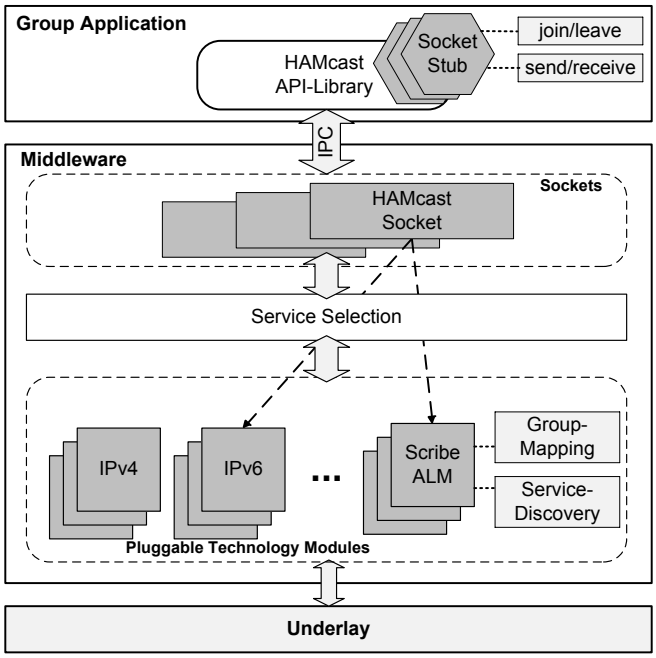

Abbildung 2: Aufbau der HAMcast Middleware

2.2 Chunk Trading

In P2P Netzwerken kann man bei der Verteilung von Daten generell zwischen zwei Ansätzen unterscheiden, dem Tree-Based Multicast und dem Mesh-Based.

Beim Tree-Based oder auch Push-Multicast genannten Ansatz organisieren sich die Peers in einer Baumstruktur. Jeder Knoten in dieser Baumstruktur besitzt dabei ausgehende Kanten zu seinen Kinderknoten. Die Daten werden entlang der Wurzel bis zu den Blattknoten weitergeleitet. Diese Art der Datenverteilung hat zwei Nachteile, die Bandbreite von Blattknoten wird nicht ausgenutzt, da diese die Daten nicht weiterleiten können. Beim Verteilen von Dateien ist es für Knoten die nachträglich dem Netzwerk beitreten nicht möglich diese noch komplett zu empfangen. Ein Lösungsansatz wäre es die Dateien periodisch zu versenden, was aber nicht sehr effizient ist. Um die Bandbreitenausnutzung der Blattknoten zu erhöhen, werden die Daten aufgeteilt und über mehrere voneinander unabhängige Bäume verteilt. Eine Bedingung dabei ist, dass jeder Knoten in einem Baum als Blatt- und Innerer-Knoten vorhanden ist. Dies hat einen hohen Managementaufwand zur Folge.

Beim Mesh-Based sind die Knoten unstrukturiert miteinander verbunden. Über einen Indexer oder auch Tracker-Server bezeichneten Rendevouz-Punkt werden Informationen über das Netzwerk ausgetauscht. Die Daten werden in kleinere Pakete aufgeteilt, den Chunks. Durch die Aufteilung in Chunks können die Daten in beliebiger Reihenfolge heruntergeladen werden. Mehrere Chunks können gleichzeitig von verschiedenen Quellen heruntergeladen werden was, die Bandbreitenausnutzung der Peers erhöht. Beim Beitreten eines Netzwerkes ruft ein Peer Listen über mögliche Kommunikationspartner ab und teilt dem Server seinen Status mit. Abhängig von der Peer-Selection Strategie verbindet sich der Peer dann mit einer Menge von Peers. Eine mögliche Peer-Selection Strategie wäre es z. B. die Peers zu bevorzugen, die möglichst viele Teile einer Datei besitzen. Eine weitere Strategie ist es das ein Peer sich zu einer zufälligen Menge von Peers verbindet. Ein weiterer wichtiger Teil ist die Chunk-Selection-Strategie, diese entscheidet darüber in welcher Reihenfolge ein Peer Chunks herunterlädt. Die Chunk-Selection-Strategie ist abhängig von der Art der Daten, die verteilt werden. So ist es nötig für das Tauschen von Dateien eine andere Strategie zu benutzen als für das Übertragen von Video-Streams.

2.2.1 GRAPES

GRAPES "Generic Library for P2P Streaming"[1] ist eine Library zur Erstellung von P2P-Mesh-Netzwerken. Ziel von GRAPES ist es dem Entwickler eine Library zur Verfügung zu stellen, die es ermöglicht nach einer Art Baukasten Prinzip neue P2P-Streaming Anwendungen zu entwickeln. Die Entwicklungszeit und die Kosten beim Erstellen neuer Anwendungen sollen dadurch reduziert werden. GRAPES steht als Open-Source Library zur Verfügung und erlaubt es somit eigene Anpassungen an der Library vorzunehmen. Ein weiterer Vorteil ist das die Algorithmen zur Datenverteilung angepasst werden können, ohne Auswirkungen auf den Rest der Anwendung zu haben.

GRAPES ist als C Library implementiert und es werden keine Abhängigkeiten zu anderes Librarys benötigt. Dies macht GRAPES auf vielen verschiedenen Systemen einsetzbar. Um sowohl Event-Based als auch Threading Anwendungen zu ermöglichen verzichtet GRAPES auf API-Calls zum Empfangen und Senden von Nachrichten diese müssen von der Anwendung unterstützt werden. Empfangene Daten können über einen API-Call an die Module weiter geleitet werden. Die komplette Library ist modular aufgebaut, es bleibt den Benutzer überlassen, welche Module er verwenden möchte. Alle Module haben eine eigene API und können durch den Benutzer erweitert oder neu implementiert werden.

Die Module

Net-Helper Modul

Das Net-Helper Modul dient den anderen Modulen als Netzwerkschnittstelle. Die Verantwortung für den Gebrauch des Net-Helpers liegt bei der Anwendung. Der Net-Helper verfügt über ein eigenes Messaging System, was zur Kommunikation zwischen den einzelnen Peers verwendet werden kann. Alternative ist es auch möglich sich die Nachrichten direkt von den Modulen generieren zu lassen, die Verantwortung der Weiterleitung dieser Nachrichten liegt dann allerdings bei der Anwendung.

Peer-Sampling Module

Das Modul dient dazu zufällig gewählten Peer-Listen, von aktiven Peers im Netzwerk, zur Verfügung zu stellen. Diese dienen dem Peer als Einstiegspunkt in das Netzwerk. Der Sampling-Algorithmus zum Auffinden der Peers kann durch eine eigene Implementation ersetzt werden.

Chunk-Trading-Modul

Die Nachrichten zur Signalisierung und den Austausch von Chunk Informationen können über das Chunk-Trading Modul verwaltet werden. GRAPES besitzt ein generisches Nachrichtenprotokoll welches es dem Benutzer erlaubt eine Vielzahl von Chunk-Trading Protokollen zu implementieren. Das Modul verfügt über Methoden zum Erstellen und lesen von Nachrichten, sowie einen eigenen Datentyp für die Verwaltung von Chunk-IDs.

Chunk-Buffer

Der Chunk-Buffer ist ein Datentyp zum Verwalten von Chunks. Der Buffer ist ausgelegt für das Verwalten von zeitkritischen Daten wie Video-on-Demand oder Video-Streaming. Zu den Eigenschaften gehört das verhindern doppelter Chunks und das Verwalten von Chunks mit Hilfe von Timestamps. So werden Chunks deren zeitliche Gültigkeit abgelaufen ist entfernt. Zur Verwaltung der Daten bietet der Buffer verschiedene Strategien, die konfiguriert werden können. Zusätzlich ist der Buffer so ausgelegt, dass der Benutzer eigene Strategien hinzufügen kann.

Scheduling-Modul

Das Scheduling dient dazu, grundlegende Scheduling Algorithmen zur Verfügung zu stellen. Die Implementation eigener Algorithmen ist für komplexere Chunk-Trading Verfahren nötig, da der Scheduling Algorithmus stark von der Art des Chunk-Tradings abhängt. So muss darüber entschieden werden welche Chunks angefordert oder verteilt werden, wie diese priorisiert werden und an welche Peers Chunks weitergeleitet werde.

2.2.2 Video-Streaming in Mesh-Netzwerken

Ein wichtiger Aspekt bei der Verteilung der Daten über ein Mesh-Netzwerk ist die "Quality of experience"(QoE) von zeitkritischen Daten. Anders als bei Push-Netzweken, wie z.B IPv4 Multicast, werden die Daten in beliebiger Reihenfolge im Netzwerk verteilt, um so die Bandbreitenausnutzung und Verfügbarkeit zu erhöhen. Dadurch ist z.B. beim Video-Streaming oder Video-on-Demand, die Zeit, die ein Video zum Buffern braucht, je nach Bandbreite sehr lang. Das ununterbrochene Abspielen eines Videos könnte somit erst nach einer kompletten Übertragung der Datei möglich sein. Zu diesem Zweck gibt es eine Vielzahl von Scheduling Algorithmen, die es ermöglichen, sollen Video-Streaming und Video-on-Demand über Chunk-Trading Netzwerke durchzuführen. Bei den überwiegenden Ansätzen werden die Chunks mit einem Zeitstempel versehen der die zeitliche Gültigkeit der Chunks darstellt. Anhand dieser Zeitstempel wird dann eine Priorisierung der Chunks vorgenommen. Ansätze für solche Scheduling Algorithmen sind “IPTV over P2P Streaming Networks: The Mesh-Pull Approach“ [3], “Improving VoD Server Efficiency with BitTorrent“ [2] und BitToS [8].

BiToS: Enhancing BitTorrent for Supporting Streaming Applications

BiToS [8] bietet einen Algorithmus der die QoE für das Video-Streaming in Mesh-Netzwerken deutlich verbessert, indem zeitkritische Chunks eingesetzt werden. Der BiToS Algorithmus benötigt dafür drei verschiedene Chunk-Buffer. Das High Priority Set ein Buffer indem alle Chunks liegen, die zeitlich am nächsten an der Abspielzeit liegen. Das Remaining-Pieces-Set indem alle übrigen Chunks aufgelistet sind, die zu einer Datei gehören. Das Received-Pieces-Set indem alle fertig heruntergeladenen Chunks liegen.

Received-Pieces-Set

Das Received-Pieces-Set unterscheidet zwischen zwei Arten von Chunks, Donwloaded und Missed. Downloaded Chunks, sind Chunks die heruntergeladen wurden, diese Daten können an den Video-Player weitergeleitet werden. Missed Chunks, sind Chunks die während ihrer zeitlichen Gültigkeit nicht bezogen werden konnten.

Remaining-Pices-Set

Enthält alle Chunks die nicht heruntergeladen wurden, nicht als Missed markiert sind und nicht im Hight-Priority-Set sind. Chunks in diesem Set haben zwei Status Möglichkeiten. Not-Requested, alle Chunks die noch nicht heruntergeladen wurden. Currenty-Downloading, alle Chunks die zur Zeit heruntergeladen werden und nicht im High-Priority-Set sind.

High-Priority-Set

Enthält Chunks die bisher nicht heruntergeladen wurden, die nicht als Missed markiert sind und die zeitlich die höchste Priorität haben. Chunks mit zeitlich hoher Priorität sind Chunks die vom Video-Player als nächstes abgespielt werden müssen. Diese Chunks haben die höchste Priorität. Das Set besitzt eine feste Größe, diese wird vom System festgelegt. Üblicherweise entspricht die Größe einige Sekunden an Video Material. Chunks in diesem Set können als Not-Requested oder Currently-Downloading markiert sein. Wenn ein Chunk fertig heruntergeladen wurde wird es in das Received-Pieces-Set gelegt und durch das zeitlich nächste Chunk in das Set eingefügt.

Piece-Selection

Ein wichtiger Aspekt bei Mesh-Netzwerken ist die Auswahl der Chunks zur Verteilung im Netzwerk. Dieses wird über einen Scheduling Algorithmus bestimmt. Bei der Verteilung von nicht zeitkritischen Daten wird dabei häufig der Rarest-First Algorithmus angewandt.

Beim Rarest-First werden die Chunks, die innerhalb der Nachbarschaft Peers am seltensten sind mit höchster Priorität von den Peers angefordert. Somit kann eine optimale Verteilung der Daten innerhalb des Netzwerkes sichergestellt werden. Auch wenn die ursprüngliche Quelle der Daten nicht mehr vorhanden ist, besteht so die Wahrscheinlichkeit, dass die Daten noch komplett im Netzwerk zur Verfügung stehen.

Bei BiToS entscheidet sich ein Peer mit einer Wahrscheinlichkeit von P ein Chunk aus dem High-Priority-Set herunterzuladen. Mit einer Wahrscheinlichkeit von P-1 entscheidet sich ein Peer ein Chunk aus dem Remaining-Pices-Set herunterzuladen.

Die Wahrscheinlichkeit soll eine Balance schaffen zwischen Chunks die unmittelbar benötigt werden (High-Priority-Set) und solche die in der Zukunft benötigt werden (Remaining-Pices-Set). Dieser Wechsel zwischen den beiden Sets sorgt dafür das sich die Daten besser im Netzwerk verteilen, besonders Benutzern mit einer hohen Bandbreite können so für eine bessere Verteilung der kompletten Daten sorgen. Innerhalb des High-Priority-Sets und Remaining-Pices-Set werden die Chunks nach dem Rarest-First Verfahren ausgewählt, wobei es eine Anpassung gibt. Wenn Chunks die gleiche Seltenheit aufweisen wird das Chunk mit der höheren zeitlichen Priorität gewählt.

Piece-Deadline

Zur Verwaltung der Gültigkeit von Chunks exsistiert eine Deadline Funktion. Diese Funktion bestimmt ob ein Chunk seine Gültigkeit behält. Sollte ein Chunk die zeitliche Gültigkeit überschritten haben, so wird es als Missed markiert. Wenn sich ein Chunk bereits im Download befindet, so wird der Download abgebrochen und das Chunk als Missed markiert. Dabei wird berechnet wie lange die kürzeste Download Zeit für dieses Chunk ist.

3 Fazit

3.1 Zusammenfassung

Das Ziel der Masterarbeit ist die Entwicklung eines Pull-Multicast Technologie Module für HAMcast. Während die Verteilung von Dateien in Mesh-Netzwerken wie z.B bei BitTorrent ausgiebig erforscht und weiterentwickelt wurde liegt die Herausforderung der Arbeit in der Entwicklung der Multicast-Gruppenverwaltung und der Erweiterung der API. Um die HAMcast API möglichst generisch zu belassen wird versucht möglichst viel über die Codierung der HAMcast URI durchzuführen. Ein weiterer Punkt ist das Verteilen von Echtzeit und zeitkritischen Daten wie Live- und Video-Streaming. Video-Streaming über Mesh-Netzwerke ist aktuell ein in der Forschung und Wirtschaft populäres Thema.

Die Generic Library for P2P Streaming (GRAPES) kann dazu verwendet werden erste Prototypen des Technologie Module zu entwickeln und verschiedene Scheduling Algorthimen für Video-Streaming und Datenverteilung zu testen. Durch den modularen Aufbau könnte es sogar möglich sein, das die Benutzer selber entscheiden, welcher Verteilungsalgorithmus verwendet werden soll.

BiToS zeigt das durch leichte Anpassung des Chunk-Trading Algorithmus es möglich ist, Video-Streaming über ein Mesh-Netzwerk durchzuführen. Eigene Messungen des Algorithmus sollten zeigen wie der Algorithmus sich bezogen auf die QoE verhält und welche Verbesserungen möglicherweise durchzuführen sind. Nachteilig ist, dass dieses Verfahren nicht für Live-Streams geeignet ist, dass im Voraus die kompletten Daten zur Verfügung stehen müssen.

3.2 Ausblick

Im zweiten Projekt geht es um die Entwicklung eines Push-Multicast Technologie Moduls für HAMcast. Der zu implementierende Push-Multicast ist BIDIR-SAM [10]. BIDIR-SAM ist ein bidirektionaler Multicast der Daten anhand von Präfix-Bäumen verteilt. Dieses Modul soll als Vergleichstechnologie für das bereits existierende Scribe Modul als auch für den Pull-Multicast dienen. Anhand dieser Vergleichstechnologien können dann Aussagen über die Effizienz des Pull-Moduls getroffen werden. Des Weiteren werden die Vorteile der verschiedenen Technologien deutlich.

Literatur

[1] ABENI, Luca ; KIRALY, Csaba ; RUSSO, Alessandro ; BIAZZINI, Marco ; LO CIGNO, Renato: Design and implementation of a generic library for P2P streaming. In: *Proceedings of the 2010 ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking*. New York, NY, USA : ACM, 2010 (AVSTP2P '10), S. 43–48. – URL http://doi.acm.org/10.1145/1877891.1877902. – ISBN 978-1-4503-0169-5

[2] CHOE, Yung R. ; SCHUFF, Derek L. ; DYABERI, Jagadeesh M. ; PAI, Vijay S.: *Improving VoD Server Efficiency with BitTorrent ?*

[3] HEI, Xiaojun ; LIU, Yong ; ROSS, K.W.: IPTV over P2P streaming networks: the mesh-pull approach. In: *Communications Magazine, IEEE* 46 (2008), february, Nr. 2, S. 86 –92. – ISSN 0163-6804

[4] JIN, Xing ; CHENG, Kan-Leung ; CHAN, S.-H. G.: Island multicast: combining IP multicast with overlay data distribution. In: *IEEE Transactions on Multimedia* 11 (2009), Nr. 5, S. 1024–1036. – ISSN 1520-9210

[5] LU, Shaofei ; WANG, Jianxin ; YANG, Guanzhong ; GUO, Chao: SHM: Scalable and Backbone Topology-Aware Hybrid Multicast. In: *16th Intern. Conf. on Computer Communications and Networks (ICCCN'07)*, August 2007, S. 699–703. – ISSN 1095-2055

[6] LUA, Eng K. ; CROWCROFT, Jon ; PIAS, Marcelo ; SHARMA, Ravi ; LIM, Steven: A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. In: *IEEE Communications Surveys & Tutorials* 7 (2005), Nr. 2, S. 72–93

[7] MEILING, Sebastian ; CHAROUSSET, Dominik ; SCHMIDT, Thomas C. ; WÄHLISCH, Matthias: System-assisted Service Evolution for a Future Internet – The HAMcast Approach to Pervasive Multicast. In: *Proc. of IEEE GLOBECOM 2010, Workshop MCS 2010*. Piscataway, NJ, USA : IEEE Press, Dec. 2010, S. 913–917

[8] VLAVIANOS, Aggelos ; ILIOFOTOU, Marios ; FALOUTSOS, Michalis: BiToS: enhancing BitTorrent for supporting streaming applications. In: *In IEEE Global Internet*, 2006, S. 1–6

[9] WÄHLISCH, Matthias ; SCHMIDT, Thomas C. ; VENAAS, Stig: A Common API for Transparent Hybrid Multicast / IRTF. URL http://tools.ietf.org/html/draft-irtf-samrg-common-api, January 2012 (04). – IRTF Internet Draft – work in progress

[10] WÄHLISCH, Matthias ; SCHMIDT, Thomas C. ; WITTENBURG, Georg: BIDIR-SAM: Large-Scale Content Distribution in Structured Overlay Networks. In: YOUNIS, Mohamed

(Hrsg.) ; CHOU, Chun T. (Hrsg.): *Proc. of the 34th IEEE Conference on Local Computer Networks (LCN)*. Piscataway, NJ, USA : IEEE Press, October 2009, S. 372–375

[11] ZHANG, Beichuan ; WANG, Wenjie ; JAMIN, Sugih ; MASSEY, Daniel ; ZHANG, Lixia: Universal IP multicast delivery. In: *Computer Networks* 50 (2006), Nr. 6, S. 781–806. – ISSN 1389-1286

Abbildungsverzeichnis

Listings