LocalCluster – Bildung von lokalen Clustern in P2P-Overlay-Netzwerken

Kristian Herpel, 26634

Diese Arbeit stellt einen Entwurf zur Bildung von lokalen Clustern in P2P-Overlay-Netzwerken vor. Die Hosts werden anhand ihrer Distanzen aus Netzwerksicht in mehrere lokale Cluster gruppiert, wobei die Hosts einer Gruppe nahe beieinander liegen. Für die Distanzermittlung werden die Hosts durch Punkte in einem mehrdimensionalen Koordinatensystem abgebildet. Bei diesem Ansatz wird die Distanz zwischen zwei Hosts durch die lineare Entfernung zwischen der Abbildung dieser beiden Hosts im Koordinatensystem bestimmt.

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren sind die Anforderungen an Bandbreite und Antwortzeiten bei vielen Anwendungen im Internet sprunghaft gestiegen. Laut einer Studie von CacheLogic[1] machen insbesondere Peer-to-Peer-Systeme (P2P) einen Großteil des gesamten Datenverkehrs aus. Die damit verbundene Übertragung großer Datenmengen über viele Teilnetze erzeugt hohe Kosten für die beteiligten Unternehmen und Nutzer.

Zur Reduzierung der Übertragungskosten sowie zur optimalen Nutzung des unterliegenden Netzwerkes für Anwendungen im Internet spielt die Aufsplittung der beteiligten Hosts in lokale Cluster eine immer größere Rolle. Lokale Cluster werden von den in einem Netzwerk vorhandenen Hosts aufgrund ihrer Nähe aus Netzwerksicht gebildet. Der Datenaustausch in diesen Clusternetzen erfolgt nicht mehr beliebig zwischen einzelnen Hosts sondern zwischen den Hosts innerhalb eines Cluster und über vorgegebene Verbindungen zwischen den Clustern.

1.1 Lokale Cluster

Durch lokale Cluster wird eine Menge von Hosts in unterschiedliche Gruppen unterteilt, so dass die Abstände zwischen den Hosts einer Gruppe aus Netzwerksicht gering sind.

Das Problem, wie lokale Cluster aus einer vorgegebenen Menge von Hosts gebildet werden, lässt sich formal wie folgt beschreiben:

Host Clustering Problem - Gegeben sei ein gerichteter Graph $G(V, E)$ wobei V die Menge der Hosts repräsentiert und E die Distanzen zwischen den Hosts auf Basis einer Distanzkarte angibt. Mit der festen Größe D wird der maximale Durchmesser für einen Cluster definiert. Das Ziel des Host Clustering Problems ist die Partitionierung des Gesamtgraphen aller Hosts und deren Verbindungen in K Cluster, so dass die Anzahl der Cluster K minimal ist und die Cluster den gesamten Graph überdecken. Die Distanz zwischen zwei Hosts in einem Cluster ist immer kleiner als D. Existiert eine Partitionierung, so dass die Bedingungen erfüllt werden können, dann ist K die optimale Clusteranzahl. Die dabei gebildeten Gruppen von Hosts sind die lokalen Cluster [AGR03].

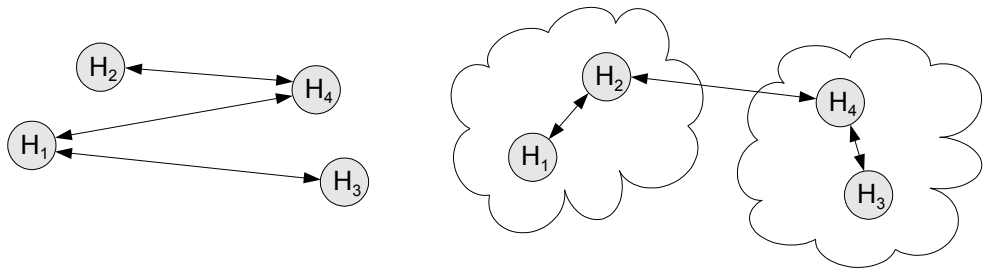

Abbildung 1: Kommunikation in Netzwerken ohne und mit lokalen Clustern

Die Abbildung 1 zeigt die Funktionsweise eines lokalen Clusters im Vergleich zu einem einfachen Netzwerk. Bei der Kommunikation mehrerer Hosts ohne lokales Clustering werden die Ver-

1 http://www.cachelogic.com/research/slide1.php

bindungen zufällig zwischen den einzelnen Hosts im Netzwerk aufgebaut. Dadurch können direkte Verbindungen zwischen Hosts über weite Entfernungen hergestellt werden, was neben zusätzlichem Zeitaufwand auch eine höhere Belastung des darunterliegenden Netzes bedeutet. Die Hosts H_1 und H_2 kommunizieren beide mit Host H_4 über eine größere Entfernung. Bei Nutzung von lokalen Clustern bilden Hosts, die sich aus Netzwerksicht nahe beieinander befinden, einen Cluster und kommunizieren innerhalb dieses Clusters. Nur Daten, die nicht im Cluster verfügbar sind, werden über wenige Außenverbindungen in den Cluster übertragen. Zwischen den Hosts H_1 und H_2 sowie H_4 auf der anderen Seite ist somit nur noch eine Datenverbindung notwendig. Die einmal übertragenen Daten werden dann innerhalb des Clusters ausgetauscht. Dieses Verfahren reduziert die übertragene Datenmenge und entlastet das unterliegende Netzwerk.

Innerhalb von Overlay-Netzen[2], die als Basis von vielen Anwendungen im Internet dienen, bietet lokales Clustering eine hochgradig effiziente Übertragung von Daten und Steuerinformationen, was zu einer deutlich höheren allgemeinen Systemperformance in diesen Netzen führt. Eine der Hauptanwendungen von Overlay-Netzen stellen P2P-Netzwerke dar. In den Netzwerken werden große Datenmengen zwischen zufällig ausgewählten Hosts übertragen. Diese zufällige Auswahl der kommunizierenden Hosts führt aber insbesondere bei großen P2P-Netzwerken zu langen Übertragungswegen und nutzt dabei nicht die Charakteristik von P2P-Netzwerken. Die in diesen Netzwerken ausgetauschten Daten sind meist mehrfach vorhanden und der Datenverkehr lässt sich deutlich reduzieren, indem die Daten zwischen nahe beieinander liegenden Hosts ausgetauscht werden. Eine Spezialisierung von P2P-Netzwerken stellt das P2P-Multimediastreaming dar, bei dem alle Hosts den gleichen Datenstrom verwenden und diesen untereinander austauschen. Aufgrund dieser Eigenschaft kann bei P2P-Multimediastreaming eine deutlich höhere Performance bei Verwendung von lokalen Clustern erreicht werden.

1.2 Aufbau und Ziel dieser Arbeit

Das Ziel dieser Arbeit ist der Entwurf und die Implementierung eines Algorithmus zur Bildung von lokalen Clustern. Neben der Clusterbildung basierend auf den Entfernungen zwischen den Hosts soll dieser Algorithmus zusätzliche Kriterien unterstützen, um problemlos im P2P-Umfeld eingesetzt werden zu können. Dafür spielen auch Faktoren wie die Bandbreite und die Verfügbarkeit der Hosts innerhalb eines Clusters eine wichtige Rolle.

Der geplante Einsatzbereich für den Algorithmus ist das P2P-Multimediastreaming, bei dem alle Hosts die gleichen Daten austauschen. Daher wird im Rahmen dieser Arbeit die Verfügbarkeit von Daten in den einzelnen Clustern nicht als Kriterium betrachtet, kann aber aufgrund der ge-

2 „Ein Overlay-Netzwerk ist ein virtuelles Netzwerk. Ein virtuelles Netzwerk besteht aus einer kleinen Untermenge von Nodes des unterliegenden Netzwerks. Diese Nodes kommunizieren über virtuelle Links oder Tunnel, die 2 Nodes des Overlay-Netzwerks verbinden. Die Daten werden zwischen den beiden Nodes über einen Tunnel unter Nutzung des unterliegenden Netzwerks ausgetauscht. Die Daten werden dabei gekapselt."
aus S.Jain, R.Mahajan „Self-Organizing Overlays" [JAI00]

forderten Erweiterbarkeit mit geringem Aufwand implementiert werden.

Die Arbeit gliedert sich in sieben Kapitel: Nach der Einleitung werden im zweiten Kapitel die allgemeinen Anforderungen für den Algorithmus aufgestellt. Das dritte Kapitel enthält die Grundlagen zum Thema sowie eine Vorstellung von allgemeinen Verfahren und weitere Arbeiten. Schwerpunkte sind dabei die Ermittlung der Position eines Hosts für die Clusterbildung und Operationen zur Verwaltung der Cluster. Auf den Entwurf des Algorithmus wird im vierten Kapitel eingegangen. Die bei der Implementierung des Ansatzes gewählten Verfahren und Lösungen werden im fünften Kapitel erläutert. Im Kapitel 6 werden der Entwurf und die Implementierung bewertet, woran sich im siebenten Kapitel das Fazit anschließt.

2 Anforderungen

In diesem Kapitel werden der Einsatzbereich des zu entwickelnden Cluster-Systems klar umrissen und die sich daraus ergebenden einzelnen Anforderungen erläutert.

2.1 Der Einsatzbereich

Das zu entwickelnde Cluster-System soll im Bereich des P2P-Multimediastreamings zum Einsatz kommen. Für dieses Anwendungsgebiet existieren verschiedene Entwürfe, im praktischen Einsatz befindet sich bisher keiner dieser Ansätze (u.a. [PAD03], [TRA03]). Der im Rahmen dieser Arbeit erstellte Entwurf dient als Erweiterung des clusterbasierten Ansatzes aus der Arbeit *„Effizientes kooperatives Multimedia-Streaming in Overlay-Netzen“* von Kai Weber [WEB04] um lokales Clustering.

P2P-Multimediastreaming unterscheidet sich vom klassischen Streaming durch die Charakteristik der beteiligten Systeme. Beim klassischen Streaming gibt es eine Trennung zwischen dem Server als Sender des Multimediastreams und den Hosts als Empfänger. Beim P2P-Multimediastreaming wird diese Unterscheidung aufgehoben. Es existiert zwar immer noch eine eindeutige Quelle mit dem Original des Multimediastreams, die beteiligten Hosts fungieren aber zusätzlich als weitere Server und versorgen andere Hosts mit dem Multimediastream. Diese Vorgehensweise ermöglicht gegenüber dem klassischen Streamingmechanismus prinzipiell eine weitaus größere Skalierbarkeit des Systems in Bezug auf die Teilnehmerzahl.

Der Einsatzbereich bedingt aber zusätzliche Faktoren, die beim Aufstellen der Anforderungen beachtet werden müssen. So spielen die verfügbare Bandbreite und die Antwortzeiten innerhalb des Systems eine wichtige Rolle. Hervorzuheben ist insbesondere der Realtime-Charakter der Daten, die per Multimediastreaming ausgetauscht werden. Es muss nicht nur beachtet werden, dass alle Hosts die Daten erhalten, zusätzlich muss sichergestellt werden, dass die Daten innerhalb einer vorgegebenen Zeit und in der benötigten Reihenfolge bei den Hosts zur Verfügung stehen.

2.2 Die Anforderungen

Der im Rahmen der Arbeit entwickelte Algorithmus muss verschiedene grundlegende Anforderungen erfüllen, um die gestellte Aufgabe erfüllen zu können. Die im Anschluss näher erläuterten Anforderungen an den Algorithmus und dessen Implementierung sind:

- Schnelles und einfaches Verfahren
- Hohe Genauigkeit bei Clusterzuordnungen
- Datenaustausch innerhalb der Cluster
- Clusterzuordnung anhand verschiedener Kriterien
- Funktionen zur Clusterpflege
- Verwendung des P2P-Paradigmas.

1 Schnelles und einfaches Verfahren

Das Clustering-Verfahren soll auf einem simplen Algorithmus mit wenigen Einzelschritten basieren. Zur Entlastung des Netzwerkes dürfen dabei nur geringe Steuerinformationen zur Verwaltung des Cluster-Systems notwendig sein. Die Übermittlung benötigter Informationen muss schnell durchgeführt werden können. Neue Hosts dürfen nur einer kurzen Wartezeit ausgesetzt sein, bis sie einem Cluster beitreten können.

2 Hohe Genauigkeit

Der Algorithmus soll eine hohe Genauigkeit hinsichtlich der Zuordnung von Hosts zu den einzelnen lokalen Clustern bieten. Dafür muss die Ermittlung der Position eines neuen Hosts zu den Clustern präzise sein.

3 Datenaustausch innerhalb eines Clusters

Der Datenaustausch zwischen den Hosts erfolgt hauptsächlich innerhalb eines Clusters. Mit den Hosts in anderen Clustern werden benötigte Daten nur ausgetauscht, wenn diese in dem Cluster nicht verfügbar sind. Im vorgegebenen Einsatzbereich des Multimediastreamings bedeutet dies im Idealfall, dass insgesamt nur eine Kopie der gesamten Daten von der Quelle oder aus anderen Clustern in einen Cluster übertragen wird. Alle Hosts innerhalb dieses Clusters versorgen sich dann untereinander.

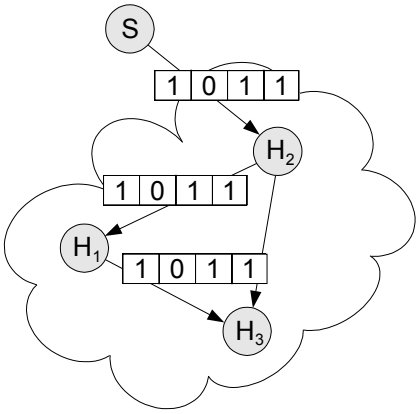

Abbildung 2: Datenaustausch bei P2P-Multimediastreaming

Die Abbildung 2 stellt den Datenaustausch bei P2P-Multimediastreaming schematisch dar. Die Quelle S versorgt den Cluster mit einer Kopie des Streams. Der Host H_2 versorgt den Host H_1 vollständig mit den empfangenen Streamingdaten und gemeinsam versorgen sie dann den Knoten H_3.

4 Clusterzuordnung anhand verschiedener Kriterien

Als Auswahlkriterium zur Zuordnung von Hosts zu den einzelnen Clustern dient die Entfernung zu den Hosts in diesen Clustern. Diese wird anhand der Latenzzeiten zwischen einzelnen Hosts bestimmt, geografische Entfernungen spielen keine Rolle. Der Algorithmus muss aber auch um zusätzliche Kriterien erweiterbar sein, die beim Beitreten eines Hosts zu einem Cluster ebenfalls

beachtet werden. Diese Erweiterbarkeit setzt voraus, dass der Algorithmus die Möglichkeit bietet, statt dem nächstgelegenen auch andere Cluster zu verwenden. Dabei muss diese Funktion ohne erneutes Ermitteln der Distanzen zwischen den verbliebenen Clustern und dem Host arbeiten.

5 Funktionen zur Pflege der Cluster

Neben den Funktionen zur Bildung und Verwaltung der Cluster benötigt der Algorithmus auch Ansätze zur Absicherung des Systems gegenüber Ausfällen sowie vordefinierte Aktionen bei Störungen in einem Cluster.

6 Arbeit nach dem P2P-Paradigma

Die vom Algorithmus gebildeten Cluster sollen auf der Basis des P2P-Paradigmas arbeiten, eine Steuerung durch externe Systeme darf nicht notwendig sein. Der P2P-Ansatz bietet eine bessere Skalierbarkeit und höhere Ausfallsicherheit gegenüber Ansätzen, bei denen die Cluster mit Unterstützung einer zentralen Kontroll- und Verwaltungsinstanz gebildet werden.

Im folgenden Abschnitt wird das P2P-Paradigma kurz erläutert. Daran anschließend werden die zusätzlichen Anforderungen für diese Systemumgebung aufgestellt.

2.3 Das P2P-Paradigma

Das Peer-to-Peer (P2P) Paradigma steht für ein Computernetzwerk ohne speziellen Server. Im Unterschied zum Client/Server-Modell mit strikter Trennung sind die Hosts (Peers) in einem P2P-Netzwerk gleichsam als Client und Server aktiv. Kommunikationsverbindungen werden in diesem Netzwerk zwischen den Peers direkt hergestellt. Durch das Fehlen von zentralen Servern ermöglichen P2P-Systeme eine weitaus höhere Skalierbarkeit bei geringeren Kosten und höherer Ausfallsicherheit als Systeme nach dem Client/Server-Modell.

Für weitergehende Informationen, eine umfangreiche Beschreibung zum P2P-Modell sowie eine Übersicht existierender Peer-to-Peer Netzwerke sei an dieser Stelle auf die entsprechende Literatur verwiesen (u.a. [SCH02], [ORE01]).

2.4 Erweiterte Anforderungen in P2P-Umgebungen

Aus der Anforderung, das lokale Clustering nach dem P2P-Paradigma umzusetzen, ergeben sich auch für den Systementwurf weitere Vorgaben. So darf zur Bildung der Cluster kein zentrales System notwendig sein, sondern jeder der beteiligten oder neu hinzukommenden Hosts muss sich selbst um die Zuordnung zu einem Cluster kümmern.

Die Bestimmung der Distanzen zwischen den verschiedenen Hosts soll nur innerhalb des Netzes erfolgen, externe Systeme werden dabei nicht verwendet. Dieses Kriterium stellt eine Verschärfung der Anforderung für P2P-Umgebungen dar, bei denen die Verwendung externer Systeme problemlos möglich ist. So kann aus P2P-Netzwerken im Internet ohne weiteres auf DNS-Server zugegriffen werden. Dafür ermöglicht diese Erweiterung auch den Einsatz in Ad-hoc-Netzwerken, in denen ausschließlich temporäre Teilnehmer vorhanden sind.

Aus Gründen der Skalierbarkeit wird für den zu entwickelnden Algorithmus vorausgesetzt, dass er ohne ein globales Wissen auskommt. Es ist bei großen Systemen unmöglich, die exakte Position aller beteiligten Hosts zu ermitteln und für weitere Abfragen vorzuhalten. Daher muss eine Positionierung und Zuordnung von Hosts zu Clustern möglich sein, ohne dass neu hinzukommende Hosts einen Gesamtüberblick vom Netzwerk haben.

Da die Daten hauptsächlich innerhalb eines Clusters ausgetauscht werden, muss die Bandbreite innerhalb des Clusters ausreichen, um alle Hosts mit den Daten zu versorgen.

2.5 Resultierende Designkriterien für die Umsetzung

Für die spätere Umsetzung des Algorithmus ergeben sich aus den erläuterten Anforderungen grundlegende Designkriterien, ebenso muss eine Evaluierung der bestehenden Ansätze hinsichtlich der aufgestellten Anforderungen erfolgen.

Als mögliches weiteres Kriterium für die Aufnahme von Hosts in die Cluster kann neben der Distanz die in den Clustern verfügbare Bandbreite dienen. Dabei sollten neue Hosts nur aufgenommen werden, wenn innerhalb des Clusters freie Bandbreite zur Versorgung des neuen Hosts verfügbar ist. Dieses Kriterium muss insbesondere im Hinblick auf praktische Anwendungen beachtet werden, da viele der potentiellen Hosts im Internet mit asymmetrischen Internetanbindungen ausgestattet sind und nur geringe Uploadbandbreiten zum weiteren Verteilen zur Verfügung stellen können. Diese Uploadbandbreite kann noch weiter verringert werden, wenn sich einzelne Hosts nicht oder nur eingeschränkt an der Verteilung der Daten beteiligen[SAR01]. Zur Sicherstellung dieses Designkriteriums muss der zu entwickelnde Algorithmus die verfügbare freie Bandbreite in einem Cluster kennen oder abfragen können oder um diese Funktionalität erweiterbar sein.

Im Hinblick auf den Einsatzbereich für Multimediastreaming ergeben sich weitere Anforderungen wie die Ausnutzung der einen verfügbaren Quelle sowie die Übertragung von unterschiedlichen Qualitätsstufen des Streams, bei der dafür gesorgt werden muss, dass innerhalb eines Clusters nur Empfänger der gleichen Qualitätsstufe des Streams vorhanden sind. Diese zusätzlichen Anforderungen sind nicht Bestandteil des im Rahmen dieser Arbeit entworfenen Algorithmus, können aber als zusätzliches Kriterium bei der Clusterzuordnung implementiert werden.

Für die Implementierung des Algorithmus wird die Programmiersprache Java[3] verwendet, um die Integration in den bestehenden P2P-Multimediastreaming-Ansatz von Kai Weber [WEB04] zu ermöglichen.

3 http://java.sun.com

3 Grundlagen

Die Unterteilung von Hosts innerhalb eines Netzwerks in lokale Cluster benötigt verschiedene Teilschritte. So müssen die Positionen der einzelnen Hosts zueinander bestimmt werden, um die Lokalisierung überhaupt durchführen zu können. Die Positionsbestimmung von einzelnen Hosts zu bekannten Hosts soll dabei aus Gründen der Skalierbarkeit unbedingt ohne Ermittlung aller Distanzen im Netzwerk bzw. ohne eine zentrale Datenbank aller ermittelten Distanzen auskommen. In einem zweiten Schritt erfolgt die Zuordnung von Hosts zu den einzelnen lokalen Clustern auf Basis der ermittelten Werte. Die gebildeten Cluster müssen im weiteren Betrieb ständig überwacht und gepflegt werden.

In diesem Kapitel werden die einzelnen Schritte und die zur Lösung der jeweiligen Aufgabe möglichen Ansätze vorgestellt. Der erste Abschnitt beschreibt verschiedene Methoden zur Ermittlung der Position eines Hosts und evaluiert diese hinsichtlich der im zweiten Kapitel aufgestellten Anforderungen. Weiterhin werden ausgewählte Verfahren zur Clusterbildung vorgestellt und ebenfalls anhand der Kriterien aus dem vorhergehenden Kapitel untersucht.

3.1 Bestimmung der Distanzen zwischen Hosts im Internet

Die im folgenden Abschnitt vorgestellten Verfahren werden hinsichtlich ihrer Genauigkeit, der erzeugten Netzwerklast und der Laufzeit beurteilt.

Es werden folgende Verfahren vorgestellt: *Ping*, *Traceroute* sowie die Verwendung von geographischen Distanzen. Im Abschnitt 3.3 werden zudem weitere Verfahren auf Grundlage von ermittelten Netzwerkinformationen erörtert.

3.1.1 Ping

Das *Ping*-Kommando[4] wird im Rahmen dieser Arbeit als Synonym für das grundsätzliche Prinzip des Versands von Echo-Paketen vorgestellt. Es ist auf nahezu allen IP-Endgeräten verfügbar und verwendet ICMP-Pakete für die Kommunikation zwischen den Hosts. Bei der Kommunikation werden ICMP-Echo-Pakete[5] an den Zielrechner gesendet und von diesem sofort beantwortet. Der Sender berechnet aus dem Absende- und dem Empfangszeitpunkt die *Round Trip Time* (Umlaufzeit), die eine sehr gute Angabe für die Distanz zwischen beiden Hosts aus Netzwerksicht ist. Um kurzfristige Netzwerkengpässe infolge der Netzwerkdynamik auszugleichen sind mehrere *Ping*-Abfragen notwendig.

Die Verwendung von Echo-Requests zu Distanzmessung bietet den großen Vorteil, dass das Verfahren auch in einer Applikation nachgebildet werden kann, womit die Nutzung in Netzwerken mit Firewalls erleichtert wird. Die Übertragung der Echo-Pakete erfolgt in diesem Fall als Teil der Pakete des jeweiligen Protokolls („*piggybacking*“ [6]).

---

4 http://ftp.arl.mil/~mike/ping.html

5 ftp://ftp.rfc-editor.org/in-notes/rfc792.txt

6 auch für „Huckepack“

Ping ist ein sehr simples und schnelles Verfahren mit hoher Genauigkeit und benötigt nur wenige Steuerinformationen [HUF02]. Es eignet sich für die Verwendung bei der Distanzbestimmung im Rahmen dieser Arbeit.

3.1.2 Traceroute

Traceroute ist eine Anwendung, die in erster Linie zur Diagnose in TCP/IP-Netzwerken entwickelt wurde und auf allen Systemen mit TCP/IP-Unterstützung verfügbar ist. Dabei wird durch Senden von UDP-Paketen mit ansteigenden TTL-Werten[7] die Route zwischen zwei Hosts durch das unterliegende Netzwerk ermittelt. Die Router zwischen den beiden Hosts geben aufgrund der ablaufenden TTL-Werte Timeout-Fehler an den Sender zurück. Eine umfassende Beschreibung der Funktionsweise von *Traceroute* bietet [HEI98, Seite 375].

Neben der Gesamtzeit für die Verbindung zwischen den Hosts identifiziert *Traceroute* auch alle Router auf dem Weg sowie die Distanzen in Millisekunden zwischen dem Sender und den einzelnen Routern. Die gesammelten Informationen zu den Routern können bei der Bildung lokaler Cluster verwendet werden, z.B. für die Zuordnung von Hosts zu lokalen Clustern anhand ihrer Zeitabstände zu bestimmten Routern oder die Bildung von lokalen Clustern über die Zugehörigkeit zu *Autonomous Systems*[8] [HEF02]. Zudem lassen sich Messungen einsparen, wenn bei unterschiedlichen Messungen die Verbindungen über die gleichen Router hergestellt werden.

Die Vielzahl der ermittelten Informationen ist gleichzeitig auch der größte Nachteil von *Traceroute* für den im Rahmen dieser Arbeit betrachteten Einsatzbereich. Neben der relativ langen Dauer zur Ermittlung der Übertragungszeit von Daten zwischen den beiden Endsystemen wirkt sich auch die bei großen verteilten Systemen erzeugte immense Datenmenge negativ auf die Performance des Gesamtsystems aus. Gegen den Einsatz von *Traceroute* steht zudem die fehlende Unterstützung in der Programmiersprache Java, mit der die Implementierung erfolgt.

Aufgrund dieser im Vergleich zu *Ping* relativ langen Zeit bis zur Ermittlung der Umlaufzeit und der großen Datenmenge bei Verwendung von *Traceroute* wird der Einsatz dieser Anwendung für die Distanzbestimmung in dieser Arbeit ausgeschlossen. Eine Bildung von lokalen Clustern auf der Basis von Routerinformationen widerspricht auch der Anforderung, das Verfahren in Ad-hoc-Netzwerken mit direkter Kommunikation zwischen den Hosts einsetzen zu können.

3.1.3 Geographische Distanzen

Zusätzlich zur Ermittlung der Umlaufzeit der Daten bei der Kommunikation zwischen zwei Hosts mittels *Ping* und *Traceroute* existieren verschiedene Ansätze, die die Distanzbestimmung anhand des geographischen Standortes der einzelnen Hosts durchführen. Neben der Verwendung von zentralen Datenbanken mit Zuordnungen von IP-Adressen zu geographischen Positionen (z.B.

7 time-to-live

8 Das Internet besteht aus mehreren *Autonomous Systems (AS)*, die untereinander verbunden sind. Jedes AS besitzt eine eindeutige 16-bit ID und wird von einer *Administrative Authority* verwaltet. Ein AS besteht aus einer Anzahl von physischen Netzen.

NetGeo[9]) wird in [PAD01] IP2Geo für statische Hosts vorgestellt. Die Bestimmung der Position erfolgt dabei nicht direkt für einen Host, sondern über den nächstgelegenen Router. Das vorgestellte *GeoTrack* arbeitet auf Basis von *Traceroute,* untersucht die Router zwischen den beiden Hosts anhand der DNS-Namen und versucht damit eine Geopositionierung. Da als Grundlage das bereits beschriebene *Traceroute* verwendet wird, ergeben sich hierbei ebenfalls die genannten Nachteile.

Leider bieten Verfahren, die auf geographischen Distanzen arbeiten, für die geplante Anwendung keine ausreichende Genauigkeit. Es gibt deutliche Unterschiede zwischen der Lage der Hosts in den Strukturen des Internets und ihrer geographischen Lage. Befinden sich verschiedenen Hosts im gleichen *Autonomous System*, so kommunizieren sie im Normalfall erheblich schneller als Hosts, die geographisch dichter beieinander liegen, aber aus Netzwerken unterschiedlicher Anbieter stammen. Ein Einsatz dieser Verfahren in Ad-hoc-Netzwerken, bei denen die Hosts geographisch sehr nahe beieinander liegen und keinen festen Standort haben, lässt sich zudem nur schwierig umsetzen.

Zentrale Datenbanken wie NetGeo widersprechen der Anforderung, dass das System unabhängig von äußeren Informationen agieren soll. Zudem steht diese Herangehensweise dem P2P-Charakter entgegen, bei dem sich das Netzwerk selbständig verwalten soll.

3.1.4 Gewähltes Verfahren

In den vorangegangenen Abschnitten wurden verschiedene Verfahren zur Bestimmung der Distanz zwischen Hosts vorgestellt und hinsichtlich der aufgestellten Anforderung untersucht. Aufgrund der problemlosen Implementierbarkeit auf allen Systemen, der schnellen Ermittlung des Ergebnisses und den geringen Steuerinformationen bietet sich ein Echo-Verfahren wie *Ping* für die Verwendung im Rahmen dieser Arbeit an. Die weiteren vorgestellten Methoden sind dagegen auf Grund ihrer geschilderten Nachteile für den Entwurf des Systems nicht geeignet: *Traceroute* benötigt eine Fülle an Steuerinformationen und geographische Distanzen sind im Internet nur zweitrangig, da sie nicht die reale Netzwerkstruktur abbilden.

3.2 Ausgewählte Verfahren zur Positionierung von Hosts

Die mit dem Echo-Verfahren ermittelten Distanzen zwischen den Hosts reichen für ein Clustering nur aus, wenn die Distanzen zwischen allen Hosts bestimmt werden. Dieser Ansatz kann aufgrund der fehlenden Skalierbarkeit in der Praxis nicht eingesetzt werden. Daher wurden verschiedene Methoden entwickelt, die mit wenigen Distanzmessungen die Bestimmung der relativen Position eines Hosts innerhalb eines Netzwerks ermöglichen. In diesem Abschnitt werden verschiedene Entwürfe sowie existierende Systeme vorgestellt und anhand der Anforderungen evaluiert. Besonders wichtig sind dabei die Selbstverwaltung des Netzwerks, die präzise und schnelle Ermittlung der Position eines Hosts sowie die Voraussetzung, dass keine zentrale Datenbank aller Verbindungen und aller Distanzen zwischen den Hosts nötig sein darf.

9 http://www.netgeo.com und http://www.caida.org/tools/utilities/netgeo/

3.2.1 Internet Distance Map Service (IDMaps)

Das IDMaps-System ([FRA00]) ist einer der ersten Ansätze, im Internet die Position eines Hosts zu ermitteln, ohne alle Distanzen einzeln zu messen. Dafür werden durch im Internet verteilte feste Hosts (*Tracer*) die Distanzen zwischen Bereichen mit unterschiedlichen IP-Adresspräfixen sowie den Tracern untereinander bestimmt. Die ermittelten Daten werden zentral gesammelt und bilden in ihrer Gesamtheit eine *Distance Map*. Für die Messungen und die Generierung dieser *Distance Map* werden in der Arbeit verschiedene Möglichkeiten vorgestellt. Die Anzahl und die Verteilung der Tracer haben deutliche Auswirkungen auf die Ergebnisse der Berechnungen.

Auf der Grundlage der *Distance Map* können nun Hosts die Entfernung zu beliebigen anderen Hosts feststellen, ohne dass dazu weitere Messungen notwendig sind. Anhand von den IP-Adresspräfixen der beteiligten Hosts wird die jeweilige Distanz näherungsweise aus den gesammelten Daten ermittelt. Zudem können Hosts mit Hilfe des *Spanning Tree* Algorithmus (u.a. [SED92]) und der *Distance Map* die Distanzen beliebiger Hosts bestimmen.

Die Berechnung der Distanzen basiert dabei auf der Annahme, dass die Distanz zwischen zwei Punkten durch die Summe der Distanzen von dazwischen liegenden Punkten abgeschätzt werden kann, was analytisch der Dreiecksungleichung entspricht. Dieses Verfahren wird als Triangulation bezeichnet.

> *Triangulation* – In einem Graphen $G(V,E)$ erfüllt eine Kostenfunktion C die Dreiecksungleichung, wenn für alle Knoten $a,b,c \in V$ gilt $C(a,c) \leq C(a,b)+C(b,c)$. Damit ist die Distanz (a,c) nach oben durch $(a,b)+(b,c)$ und nach unten durch $|(a,b)-(b,c)|$ beschränkt und lässt sich bei bekannten Distanzen (a,b) und (b,c) abschätzen. [FRA00]

In der Arbeit von Francis [FRA00] wird auf der Basis von gemessenen Werten überprüft, inwieweit mit der Annahme der Triangulation akkurate Werte berechnet werden. Die Auswertung dieser Untersuchung lässt keine allgemeine Aussage zur Genauigkeit zu, trotzdem vertreten die Autoren die Auffassung, dass die Verwendung der Triangulation einen praktikablen Ansatz darstellt.

Der Vorteil bei Verwendung von IDMaps zur Positionierung von Hosts ist die Bestimmung des Ergebnisses ohne weitere Messungen. Dabei ist aber eine zentrale Instanz zur Speicherung der *Distance Map* notwendig. Vor der Anwendung des Verfahrens sind zudem Messungen durch feste Hosts (Tracer) notwendig, um die *Distance Map* zu erstellen. Somit ist IDMaps ein sehr statisches System und kann nicht in Ad-hoc-Netzwerken und dynamischen P2P-Netzwerken verwendet werden.

3.2.2 Global Network Positioning (GNP)

Das Global Network Positioning ist ein Verfahren zur Positionierung von Hosts in einem geometrischen Raum [ZHA01]. Jeder Host in einem Netzwerk wird durch Koordinaten in diesem geometrischen Raum abgebildet. Die Distanzbestimmung zwischen Hosts im Netzwerk erfolgt dann über die Berechnung der Abstände zwischen den Koordinaten der Hosts.

Die Verwendung eines Koordinatensystems bei der Distanzbestimmung erlaubt bei Kenntnis der eigenen Koordinaten eine schnelle und einfache Auswahl der nahe liegenden Systeme aus einer großen Menge von Systemen, ohne dabei Messungen durchführen zu müssen. Für die Berechnungen kann auf die Koordinaten der einzelnen Hosts zurückgegriffen werden, es müssen keine Distanzinformationen mehr zwischen den Hosts übertragen werden.

Durch die dabei auftretende Reduzierung der Datenmenge kann die Netzwerklast bei der Distanzbestimmung gesenkt werden: Die Koordinaten zur Bestimmung der Distanzen zwischen K Hosts werden bei GNP mit einer Datenmenge von $O(K \cdot D)$ übertragen, wobei D die konstante Datenmenge für die Koordinaten eines Hosts ist. Bei Übermittlung aller berechneten Distanzen ist die Komplexität der Datenmenge dagegen $O(K \cdot K)$. GNP bietet damit eine höhere Skalierbarkeit und gute Einsatzmöglichkeit in P2P-Netzwerken.

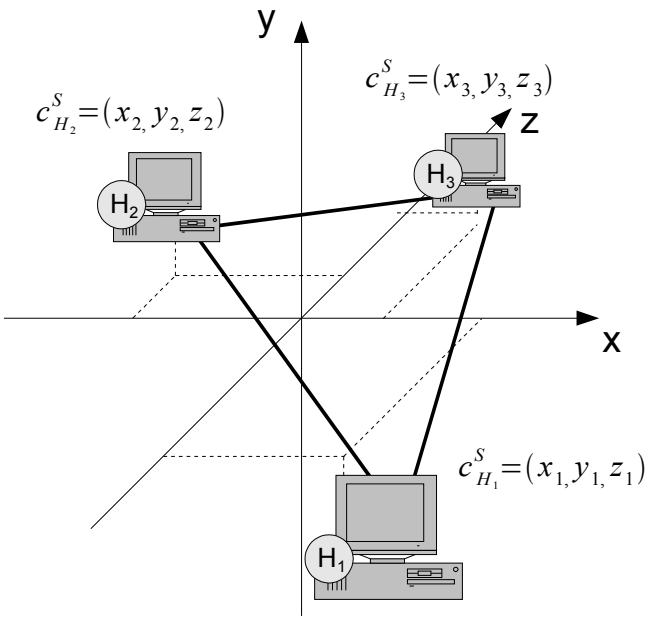

Abbildung 3: Dreidimensionaler geometrischer Raum bei GNP

GNP und IDMaps sind beides Verfahren, die mit Landmarks arbeiten. Landmarks sind feste, allen Hosts bekannte Referenzhosts im Netzwerk. Die Positionsbestimmung der einzelnen Hosts erfolgt relativ zu diesen Referenzhosts. Bei GNP bestimmen die Landmarks ihre Koordinaten in einem gewählten geometrischen Raum und speichern sie in einer Liste (*Frame of Reference*). Alle Hosts können nun mit Hilfe der Liste und den gemessenen Distanzen zu den Landmarks ihre Position im geometrischen Raum bestimmen. Die Ermittlung der Distanzen zu den Landmarks kann z.B. per Echo-Nachrichten erfolgen (siehe Abschnitt 3.1.1).

Der allgemeine Ablauf des GNP teilt sich in die Operationen der Landmarks und der einzelnen Hosts auf.

Grundlagen

Global Network Positioning verwendet einen geometrischen Raum S. Ein Host H wird in S durch die Koordinaten c_H^S positioniert. Die Berechnung der Distanz zwischen zwei Hosts H_1 und H_2 erfolgt durch eine Distanzfunktion $f^S(\cdot)$ mit:

$$f^s(c_{H_1}^S, c_{H_2}^S) = \hat{d}_{H_1 H_2}^S$$

Bei einem geometrischen Raum S der Dimension D werden mindestens $D+1$ Landmarks benötigt, da sonst keine eindeutigen Koordinaten für die Hosts bestimmt werden können.

Landmark Operationen

Bevor Hosts ihre Koordinaten bestimmen und damit Distanzen berechnen können, erfolgt in einem ersten Schritt die Ermittlung der Positionen der Landmarks im geometrischen Raum S. Dazu bestimmen die n Landmarks L_1 bis L_n per ICMP Ping-Nachrichten die Round Trip Time durch wiederholte Messungen untereinander. Aus den gemessenen Distanzen $d_{H_i H_j}^S$ zwischen den Landmarks H_i und H_j mit $i > j$ wird eine $n \times n$ Distanzmatrix gebildet. Mit dieser Matrix müssen nun durch einen Host die Koordinaten $c_{L_1}^S, \ldots, c_{L_n}^S$ für die n Landmarks so bestimmt werden, dass der Gesamtfehler zwischen den gemessenen Distanzen und den Distanzen zwischen den Koordinaten minimiert wird. Dazu ist die Minimierung der Funktion

$$f_{obj1}(c_{L_1}^S, \ldots, c_{L_N}^S) = \sum_{L_i, L_j \in \{L_1, \ldots, L_N\} | i > j} \varepsilon(d_{L_i, L_j}, \hat{d}_{L_i L_j}^S)$$

notwendig, wobei $\varepsilon(\cdot)$ eine beliebige Fehlerfunktion wie z.B. der quadratische Fehler ist. Die Berechnung der Koordinaten anhand der Gleichung ist ein multidimensionales globales Minimierungsproblem, deren Berechnungsmöglichkeiten in der Arbeit [ZHA01] erläutert werden. Prinzipiell existieren unendlich viele Lösungen, da aber nur die relative Lage der Landmarks zueinander wichtig ist, nicht deren absolute Lage im Raum, reicht eine Lösung für das Minimierungsproblem aus.

Nachdem die Koordinaten der Landmarks berechnet wurden, erfolgt eine Verteilung dieser Koordinaten an alle Hosts, die am GNP teilnehmen möchten. Diese können auf der Basis der vorberechneten Koordinaten der Landmarks die eigene Position innerhalb von S ermitteln.

Host Operationen

Die Hosts können beim Global Network Positioning mit wenigen Messungen und einer anschließenden Berechnung ihre Position innerhalb des geometrischen Raums S selbständig ermitteln. Dazu bestimmen sie zuerst die Round Trip Times zu den n Landmarks per ICMP Ping-Kommando. Mit den gemessenen Werten können nun die Koordinaten des Hosts in S so bestimmt werden, dass der Gesamtfehler zwischen den gemessenen und berechneten Distanzen zwischen dem Host und den Landmarks in der Funktion

$$f_{obj2}(c_H^S) = \sum_{L_i \in L_{1,\dots}, L_n} \varepsilon(d_{HL_i}, d_{HL_i}^{\hat{S}})$$

minimal wird.

Einschätzung zu GNP

Global Network Positioning ist ein Verfahren zur Positionierung von Hosts, bei dem nur wenige Messungen notwendig sind. Es erzeugt nur geringe Netzwerklast durch wenige ICMP Ping-Nachrichten, denn die eigentliche Distanzberechnung führen die Hosts auf der Basis der ermittelten Koordinaten durch. Die in der Arbeit [ZHA01] durchgeführten Simulationen zeigen eine hohe Genauigkeit bei schneller Ermittlung der Werte bei Verwendung von 15 Landmarks in einem 7-dimensionalem euklidischen Raum. Bei Einsatz von GNP zur Positionierung von Hosts werden keine externen Systeme benötigt, weshalb sich dieses Verfahren prinzipiell zur Verwendung in Ad-hoc-Netzwerken eignet.

Einen großen Nachteil von GNP stellt die benötigte Unterstützung durch eine Infrastruktur (Landmarks) dar. Ohne Vorberechnung der Landmark-Koordinaten ist keine Positionierung von Hosts möglich. Zudem findet keine Überprüfung der Koordinaten statt, die durch einen Host geliefert werden.

3.2.3 Vivaldi

Das Vivaldi-Verfahren wurden von Russ Cox u.a. [COX03] auf den Prinzipien von GNP entwickelt, setzt jedoch keine festen Hosts in dem System voraus. Der Algorithmus arbeitet verteilt und ordnet den Hosts synthetische Koordinaten zu. Die Distanzbestimmung zwischen den Hosts erfolgt dann als euklidische Distanz auf Basis der berechneten Koordinaten.

Wie bei GNP werden aus den gemessenen Latenzzeiten die Koordinaten der Hosts im Koordinatensystem ermittelt, so dass der Fehler E zwischen den gemessenen realen Distanzen und den Distanzen zwischen den Koordinaten der Hosts minimal ist. Als Fehlerfunktion wird auch in dieser Arbeit der quadratische Fehler verwendet.

Für die Minimierung der Fehlerfunktion und damit der genauen Koordinatenberechnung werden die Verbindungen zwischen Hosts als physikalische Federn modelliert. Die Länge der Feder im Ruhezustand l_R ist die reale Latenzzeit zwischen den beiden Hosts, die aktuelle Länge der Feder im Modell l_S ist dagegen der Abstand zwischen den Koordinaten der Hosts im euklidischen Raum. Die potentielle Energie einer Feder ist dabei proportional zum Quadrat der Differenz von aktueller Federlänge und der Länge im Ruhezustand $F \propto (l_R - l_S)^2$. Dieser Unterschied ist identisch zum Fehler bei der Vorhersage von Koordinaten für die Hosts. Die Minimierung der potentiellen Energie des Gesamtsystems entspricht somit der Minimierung des Fehlers E .

Bei der erstmaligen Kommunikation zweier Hosts im Netzwerk erfolgt eine Neuberechnung der eigenen Koordinaten anhand der bestimmten Latenzzeiten bei der Übertragung. Die Hosts tauschen ihre aktuellen Koordinaten aus und reduzieren den Unterschied zwischen den ge-

messenen und den berechneten Werten. Dazu werden die beiden betroffenen Hosts im Modell entlang der Geraden durch diese Hosts in eine durch die Latenzzeit vorgegebene Richtung gezogen:

> „Ein Node bewegt seine Koordinaten zu einem Punkt p auf der Geraden zwischen sich und dem anderen Knoten. Der Punkt p wird dabei so gewählt, dass die Differenz der vorhergesagten und der gemessenen Latenzzeit zwischen den beiden Knoten 0 ist. Um Schwankungen zu vermeiden, bewegt sich die Node nur um einen bestimmten Bruchteil ϕ in Richtung des Punktes p." [COX03, Seite 2]

Der Wert von ϕ wird bei jedem Durchlauf der Berechnungsfunktion bis zu einer vorgegebenen unteren Grenze reduziert. Bei den durchgeführten Simulationen haben sich der Startwert $\phi = 1$, ein Reduktionswert pro Durchlauf von 0,025 und die untere Grenze bei 0,05 als gute Berechnungsgrundlage erwiesen.

Neue Knoten können ihre Startkoordinaten selbst bestimmen, indem sie die Latenzzeit zu mehreren Hosts im Netzwerk bestimmen. Alle dabei verwendeten Verbindungen werden als zusätzliche physikalische Federn in das Gesamtsystem eingefügt. Die Tests in Rahmen der Arbeit [COX03] ergeben, dass eine genaue Positionierung des neuen Hosts mit Distanzmessungen zu 10 im Netzwerk vorhandenen Hosts möglich ist.

Vivaldi ist ein verteiltes System, bei dem zur Ermittlung der Positionen von Hosts nur wenige Steuerinformationen notwendig sind und das eine hohe Genauigkeit besitzt. Die im Rahmen der Arbeit [COX03] durchgeführten Simulationen ergeben bei 750 Hosts einen durchschnittlichen Fehler von 20ms bei den berechneten Distanzen. Der Algorithmus arbeitet im Vergleich zu GNP ohne fest vorgegebene und allen Hosts bekannte Landmarks.

Als Nachteil erweisen sich die Abhängigkeiten zwischen allen Hosts durch die Modellierung der Verbindungen als Federn. Die Modifikation der Position eines Hosts bewirkt auch Positionsänderungen für alle anderen Hosts, die mit dem Host kommunizieren. Das Verfahren eignet sich aus diesem Grund nur wenig für skalierbares Clustering, da zwischen den Hosts unterschiedlicher Cluster geänderte Koordinaten für Neuberechnungen übertragen werden müssen.

3.2.4 Practical Internet Coordinates for Distance Estimation (PIC)

Die bisher vorgestellten koordinatenbasierten Verfahren GNP und Vivaldi weisen Nachteile bei der Verwendung in realen Umgebungen auf. GNP verwendet zur Berechnung der Koordinaten fixe Landmarks und kann somit nur schlecht in P2P-Umgebungen eingesetzt werden, die durch eine stark schwankende Verfügbarkeit der Hosts gekennzeichnet sind. Der Ansatz von Vivaldi löst zwar das Problem von fixen Landmarks, ist aber anfällig gegenüber falschen Koordinaten- und Distanzangaben durch einzelne Hosts. Diese verfälschten Werte führen zu fehlerhaften Berechnungen bei der Bestimmung von Koordinaten für neue Hosts im Overlay-Netzwerk.

Das von M.Costa, M.Castro u.a. in [COS03] vorgestellte PIC-Verfahren löst das Problem der fixen Landmarks, indem alle Hosts mit bereits berechneten Koordinaten als potentielle Landmarks

genutzt werden können. Zudem kann PIC genaue Koordinaten auch dann berechnen, wenn einzelne Landmarks fehlerhafte Koordinaten an die Hosts übermitteln.

Die grundsätzliche Funktionsweise von PIC entspricht dem Ansatz von GNP. Ein neuer Host berechnet die eigenen Koordinaten in einem d - dimensionalen Raum auf der Basis von gemessenen Distanzen zu mindestens $d+1$ Landmarks und deren bereits berechneten Koordinaten. Dazu wird wie bei GNP ein multidimensionaler globaler Optimierungsalgorithmus verwendet, der den Fehler zwischen den gemessenen Distanzen im Overlay-Netzwerk und den berechneten Distanzen im Koordinatensystem minimiert.

Für die Genauigkeit der bestimmten Koordinaten bei PIC spielt die Auswahl der Landmarks eine große Rolle. Die Autoren evaluieren in [COS03] verschiedene Strategien zur Wahl der Landmarks:

1. Zufällige Auswahl von Landmarks
2. Nächstgelegene Hosts als Landmarks
3. Hybride Strategie aus den beiden anderen Ansätzen.

Bei der zufälligen Wahl der Landmarks werden aus allen Hosts mit bereits berechneten Koordinaten eine bestimmte Anzahl beliebiger Hosts ausgewählt. Ein neuer Host ermittelt die Distanzen zu diesen Hosts und berechnet dann mit deren Koordinaten die eigene Position.

In den beiden anderen Strategien müssen neue Hosts für die Berechnung der Koordinaten die nächstgelegenen Hosts finden. Dazu kann ein „Durchwandern“ des Overlay-Netzwerks durchgeführt werden, bis diese nächstgelegenen Hosts gefunden werden. Dieser Ansatz entspricht aber dem im Rahmen dieser Arbeit ebenfalls vorgestellten mOverlay (siehe Abschnitt 3.3.2) und verursacht durch die wiederholten Distanzmessungen erhebliche Netzwerklast. Zudem könnte der nächstgelegene Cluster bei dieser Herangehensweise auch direkt bestimmt werden.

Angesichts der Nachteile des „Durchwanderns“ wird für PIC eine andere Lösung genutzt. Ein neuer Host berechnet zunächst näherungsweise seine Koordinaten durch zufällige Auswahl von Landmarks. Anschließend erfolgt das „Durchwandern“ des Overlay-Netzwerks, wobei als Startpunkt nun die Landmarks verwendet werden, die im ersten Schritt als nächstgelegene Hosts ermittelt wurden. Mit diesen Hosts berechnet der neue Host bei der Strategie „Nächstgelegene Hosts als Landmarks“ dann die eigenen Koordinaten erneut. Der hybride Ansatz unterscheidet sich in diesem Schritt durch die verwendeten Hosts, da nicht nur die neu gefundenen nächstgelegenen Hosts sondern auch die zufälligen Hosts und deren bereits bestimmte Distanz für die Neuberechnung verwendet werden.

Der Bootstrap-Vorgang bei PIC stellt im Vergleich zu GNP aufgrund von fehlenden fixen Landmarks ein größeres Problem dar. Die Hosts müssen ihre Koordinaten von Beginn an ohne externe Systeme selbständig bestimmen. Zu diesem Zeitpunkt ist das Overlay-Netzwerk noch sehr klein und es stehen nicht genügend Landmarks zur Verfügung, um eine genaue Abbildung der Hosts in einem Koordinatensystem der Dimension d durchführen zu können (siehe Abschnitt 3.2.2). Tritt ein neuer Host n einem Overlay-Netzwerk mit Hosts N und Landmarks L bei, in dem

$|N|<|L|$ und $|L|>d$ gilt, so misst der Host die Distanz zu allen Hosts in N und fragt zusätzlich die Distanzen zwischen den Hosts aus N untereinander ab. Mit den gemessenen Werten wird eine $N \times N$ Matrix gebildet, die dann als Basis für die Einbettung der Hosts in ein Koordinatensystem dient. Dieser Vorgang verläuft analog zu den Landmark-Operationen bei GNP.

Für die genaue Positionierung von Hosts im Koordinatensystem muss sichergestellt sein, dass die ermittelten Werte mit den realen Gegebenheiten übereinstimmen. Im Gegensatz zu Vivaldi und GNP bietet PIC die Möglichkeit, fehlerhafte Werte einzelner Landmarks zu erkennen und darauf zu reagieren. Dabei berechnet ein neuer Host seine Koordinaten und überprüft dann mit diesen Werten, ob die Dreiecksungleichung für alle Paare von Landmarks und dem Host erfüllt ist:

L ist eine Menge von Landmarks, die zur Berechnung der Koordinaten für einen neuen Host n verwendet werden, wobei i und j zwei Landmarks in L mit $i \neq j$ sind. Die gemessenen Distanzen zwischen dem Host n und den Landmarks betragen $dist_i^m$ für Host n und Landmark i sowie $dist_j^m$ für Host n und Landmark j. Die mit den Koordinaten berechnete Distanz zwischen den beiden Landmarks i und j beträgt $dist_{i,j}^p$.

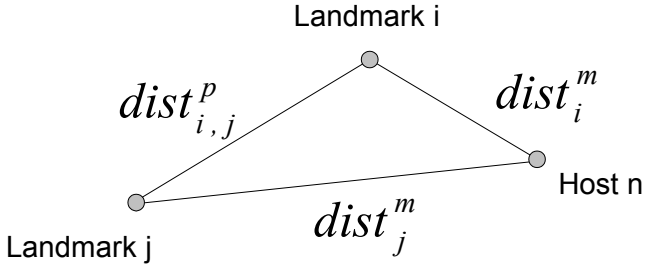

Abbildung 4: Dreiecksungleichung für Host und Landmarks

Bei korrekten Werten für Landmark i gelten nach der Dreiecksungleichung die folgenden Bedingungen:

$$dist_i^m \leq dist_j^m + dist_{i,j}^p \; (obere\ Grenze\ für\ dist_i^m)$$
$$dist_i^m \geq dist_j^m - dist_{i,j}^p \; (untere\ Grenze\ für\ |dist_j^m - dist_{i,j}^p|)$$
$$dist_i^m \geq dist_{i,j}^p - dist_j^m \; (untere\ Grenze\ für\ |dist_j^m - dist_{i,j}^p|)$$

Der neue Host n überprüft nun diese Bedingungen für alle Paare i und j aus L und berechnet dabei die folgenden Metriken:

$$upper_i = \sum_{j=1}^{|L|} \begin{cases} dist_i^m - (dist_j^m + dist_{i,j}^p) & wenn\,(dist_j^m + dist_{i,j}^p) < dist_i^m \\ 0 & sonst \end{cases}$$
$$lower_i = \sum_{j=1}^{|L|} \begin{cases} |dist_j^m - dist_{i,j}^p| - dist_i^m & wenn\,|dist_j^m - dist_{i,j}^p| > dist_i^m \\ 0 & sonst \end{cases}$$

Die Metrik $upper_i$ ist die Summe aller Abweichungen von der oberen Grenze für $dist_i^m$; die Summe der Abweichung von der unteren Grenze für $|dist_i^m - dist_{i,i}^p|$ ist $lower_i$.

Der PIC-Algorithmus entfernt nun den Landmark aus der Menge L, der die größten Werte für $upper_i$ und $lower_i$ aufweist und führt die Berechnung der Koordinaten mit den restlichen Landmarks erneut durch. Diese Schritte können auch mehrfach wiederholt werden, um die Sicherheit gegenüber fehlerhaften Angaben von Landmarks weiter zu erhöhen.

Die im Rahmen der Arbeit [COS03] durchgeführten Tests und Vergleiche der drei vorgestellten Strategien ermitteln als optimale Werte $d=12$ für die Dimensionen und $|L|=16$ für die Landmarks. Eine Erhöhung der Anzahl der Dimensionen über diesen Wert führte bei keinem der Ansätze zu einer erhöhten Genauigkeit in den berechneten Koordinaten.

Ebenfalls untersucht wurde die Auswirkung der Zahl von verwendeten Landmarks auf das Resultat. Nur bei der zufälligen Auswahl der Landmarks führte die Verwendung von mehr als 16 Landmarks zu genaueren Ergebnissen, bei der hybriden Strategie und der Verwendung von nächstgelegenen Hosts als Landmarks blieben die Ergebnisse konstant.

Im Vergleich der drei Ansätze mit GNP konnte nur die hybride Strategie ähnlich genaue Koordinaten für die Hosts liefern, in den beiden anderen Fällen war der relative Fehler deutlich größer.

3.2.5 Weitere Verfahren

Neben dem Global Network Positioning und Vivaldi existieren weitere Verfahren, die ebenfalls mit Landmarks oder mit der Positionierung von Hosts in geometrischen Räumen arbeiten. GNP wurde in dieser Arbeit als prinzipieller Ansatz vorgestellt, da sich viele der anderen Verfahren explizit auf GNP beziehen und die grundlegende Idee um zusätzliche Faktoren erweitern.

Bei mServer [THE00] werden eine Menge von Servern zur Messung verwendet (die mServer), die ihre Distanzen zueinander bestimmen. Die Hosts werden dann dem nächstgelegenen mServer zugeordnet und bestimmen ihre Distanzen untereinander durch die Distanzen zwischen den zugeordneten mServern. Dieses Verfahren ist ungenauer als GNP, da die Abstände zwischen Hosts und mServern nicht betrachtet werden. Dafür entfällt bei diesem Verfahren die Berechnung von Positionen durch die einzelnen Hosts.

Mit Big-Bang Simulation [SHA02] und Lighthouses [PIA03] existieren neben Vivaldi und PIC weitere Verfahren, die die Koordinaten von Hosts ohne Unterstützung von festen Hosts bestimmen. Bei Big-Bang Simulation (BBS) erfolgt die Positionierung der Hosts durch Simulation einer Partikelexplosion in einem Kraftfeld. Dazu werden die Hosts als bewegliche Partikel in einem euklidischen Raum unter Beeinflussung durch ein Kraftfeld modelliert. Das Kraftfeld reduziert die potentielle Energie der einzelnen Partikel aufgrund des Gesamtfehlers aller paarweisen Distanzen zwischen den Partikeln, die bei der Einbettung in den geometrischen Raum auftreten. Als Resultat aus diesen Berechnungen bewegen sich die einzelnen Partikel auf Positionen mit minimalem Fehler für das gesamte System zu.

Der Ansatz von Lighthouses verwendet lokale Koordinatensysteme, in denen neue Hosts ihre Koordinaten bestimmen. Dazu erhalten sie eine Liste von dynamisch ausgewählten Hosts aus dem bestehenden Netzwerk und ermitteln die Distanzen zu diesen Hosts. Wie bei GNP werden die

Koordinaten der neuen Hosts in diesem lokalen Koordinatensystem berechnet, anschliessend erfolgt aber die Berechnung von globalen Koordinaten für das Gesamtnetzwerk mit Hilfe einer Transitionsmatrix. Das Verfahren weist eine ähnliche Genauigkeit wie GNP auf, arbeitet dabei aber ohne vorgegebene Landmarks. Durch deren dynamische Auswahl wird auch einer Überlastung der Landmarks vorgebeugt, womit sich die Skalierbarkeit des Gesamtsystems erhöht. Als Nachteil des Verfahrens erweisen sich die umfangreichen Vektorrechnungen. Zudem wird mit dem Gram-Schmidtschen Orthogonalisierungsverfahren ein Algorithmus genutzt, der für den Einsatz in Computersystemen aufgrund von Rundungsfehlern nicht geeignet ist (u.a. [GIR03]).

3.2.6 Gewählter Ansatz

Für den im Rahmen dieser Arbeit zu entwickelnden Algorithmus bietet sich die Verwendung eines koordinatenbasierten Ansatzes wie GNP oder Vivaldi an. Das Abbilden von Hosts auf Koordinaten in einem geometrischen Raum und die Berechnung der Abstände zwischen den Koordinaten als Ersatz für Distanzmessungen führt zwar grundsätzlich zu Ungenauigkeiten gegenüber der direkten Messung, jedoch besitzt dieser Ansatz insbesondere im Hinblick auf die Skalierbarkeit des Gesamtsystems deutliche Vorteile. Zudem ist eine schnelle Bestimmung der Position mit wenigen Messungen möglich, anschließend können die Distanzen zu allen Hosts mit Kenntnis ihrer Koordinaten ermittelt werden.

Die Arbeitsweise des ebenfalls vorgestellten IDMaps weist dagegen eine zu große Gefahr von fehlerhaften Ergebnissen auf. Die bestimmten Distanzen zwischen den Hosts sind extrem abhängig von den Positionen der jeweils zugeordneten Tracer. Liegen beide Hosts zwischen diesen Tracern, so ist der berechnete Abstand zwischen dieses Hosts erheblich größer als der reale Abstand im Netzwerk.

Nach der Festlegung auf ein koordinatenbasiertes Verfahren soll an dieser Stelle aber auf einen wichtigen Kritikpunkt am vorgestellten GNP-Ansatz bezüglich der aufgestellten Anforderungen hingewiesen werden: GNP arbeitet mit fest vorgegebenen Landmarks und ist damit nicht unabhängig von der zugrunde liegenden Infrastruktur. Eine Lösung dieser Problematik bieten das ebenfalls vorgestellte PIC-Verfahren und der Ansatz von Lighthouses.

3.3 Ausgewählte Verfahren zur Clusterbildung

Im folgenden Abschnitt werden verschiedene Verfahren zur Bildung von Clustern vorgestellt. Diese Verfahren werden dabei anhand der folgenden Kriterien untersucht:

1. Wie wird für einen Host der zugehörige Cluster ermittelt?
2. Wie erfolgt das Hinzufügen von Hosts zu den bestehenden Clustern?
3. Wie werden neue Cluster erzeugt?

Im Abschnitt 3.1 wurden Verfahren zur Distanzmessung zwischen Hosts evaluiert. Dabei erfolgte die Empfehlung, Echo-Nachrichten als Basis für den zu entwickelnden Algorithmus zu nutzen. Neben Echo-Nachrichten existieren weitere Verfahren zur Abstandsmessung von Hosts als Grund-

lage für das Gruppieren in Cluster. Im folgenden Abschnitt erfolgt daher eine kurze Betrachtung verschiedener Verfahren.

3.3.1 Verfahren zur Clusterbildung ohne direkte Distanzbestimmung

Für die bei der Clusterbildung durchgeführten Distanzbestimmungen zwischen Hosts im Internet können neben Echo-Nachrichten auch die IP-Adressen und Routinginformationen genutzt werden. Da sie aber teilweise umfangreiche Netzwerkinformationen benötigen oder Ungenauigkeiten bei der Clusterbildung aufweisen, werden die Ansätze im Folgenden nur kurz vorgestellt.

Die Bildung von Clustern anhand der IP-Adressen von Hosts wird in [KRI00] vorgestellt und untersucht. Bei diesem Verfahren werden alle Hosts mit n übereinstimmenden Bits im Präfix ihrer IP-Adressen als zusammengehörig klassifiziert. Der Ansatz basiert auf der Annahme, dass Hosts mit ähnlichen IP-Adressen auch aus Netzwerksicht nahe beieinander liegen. Eine Zuordnung von Hosts zu den einzelnen Clustern kann dabei sogar ohne Messungen erfolgen. Die Verwendung der IP-Adressen stellt aber auch den Nachteil dieser Lösung dar, da Übereinstimmungen in den IP-Adresspräfixen nicht zwangsläufig ein Indikator für geringe Entfernungen aus Netzwerksicht sind. Die Fehlerrate bei der Clusterzuordnung in den durchgeführten Simulationen betrug mehr als 50%. Eine sinnvolle Anwendung des Vefahrens zum Clustering ist zudem nur bei einer genügend großen Zahl von Hosts möglich. Andernfalls besteht die Gefahr, dass durch die Verteilung der wenigen Hosts auf verschiedene IP-Adressbereiche das Overlay-Netzwerk in viele und extrem kleine Cluster aufgesplittet wird. Gegen eine Verwendung dieses Ansatzes spricht neben den geschilderten Problemen auch die Kopplung an IP-Netzwerke, wodurch der Einsatz in anderen Umgebungen nicht möglich ist.

Angesichts der Schwierigkeiten bei der Verwendung von IP-Adressen wird in [KRI00] ein weiterer Ansatz vorgestellt. Dabei werden die Routinginformationen des *Border Gateway Protocol* (BGP)[10], einem Standard-Routingprotokoll im Internet, ausgewertet. Das BGP ist auf nahezu allen Internet-Routern implementiert und ermöglicht das Zuordnen von IP-Adressen zu *Autonomous Systems* im Internet. Aufgrund des notwendigen aber nicht sichergestellten Zugriffs auf die BGP-Informationen der Router und der Beschränkung auf das Internet wird dieses Verfahren hier nicht weiter betrachtet.

10 RFC 1265 bis 1268, http://www.rfc-editor.org/

3.3.2 mOverlay

Der in der Arbeit „A Construction of Locality-Aware Overlay Network: mOverlay and its Performance“ [ZHA04] vorgestellte Ansatz bildet lokale Gruppen[11] auf Basis von Distanzmessungen mit ICMP-Echo-Nachrichten (siehe Abschnitt 3.1.1). Dabei werden zwei Hierarchien im Netzwerk verwendet, die oberste Ebene enthält die Gruppen und die untere Ebene die Hosts in diesen Gruppen. Die Informationen werden komplett zwischen allen Hosts einer Gruppe ausgetauscht. Das Verfahren unterteilt sich in zwei Basiskomponenten: der Lokalisierungsprozess zum Auffinden der nächsten Gruppe und die Protokolle zur Bildung und Pflege des Overlay-Netzwerkes mit den lokalen Gruppen.

Die Gruppen in einem mOverlay-Netzwerk besitzen einen lokalen Host Cache, der Informationen über die Hosts der jeweiligen Gruppe enthält. Diese Hosts dienen zur Kommunikation mit Hosts in anderen Gruppen. Zusätzlich pflegen die Gruppen selbständig Informationen zu mehreren Nachbargruppen, wie die Distanzen zu diesen Nachbargruppen und den Hosts in deren Host Caches. Der erste Host in einem Host Cache übernimmt die Funktion des Leaders und ist damit für die Aktualisierung des Host Cache und der Informationen über die Nachbargruppen verantwortlich.

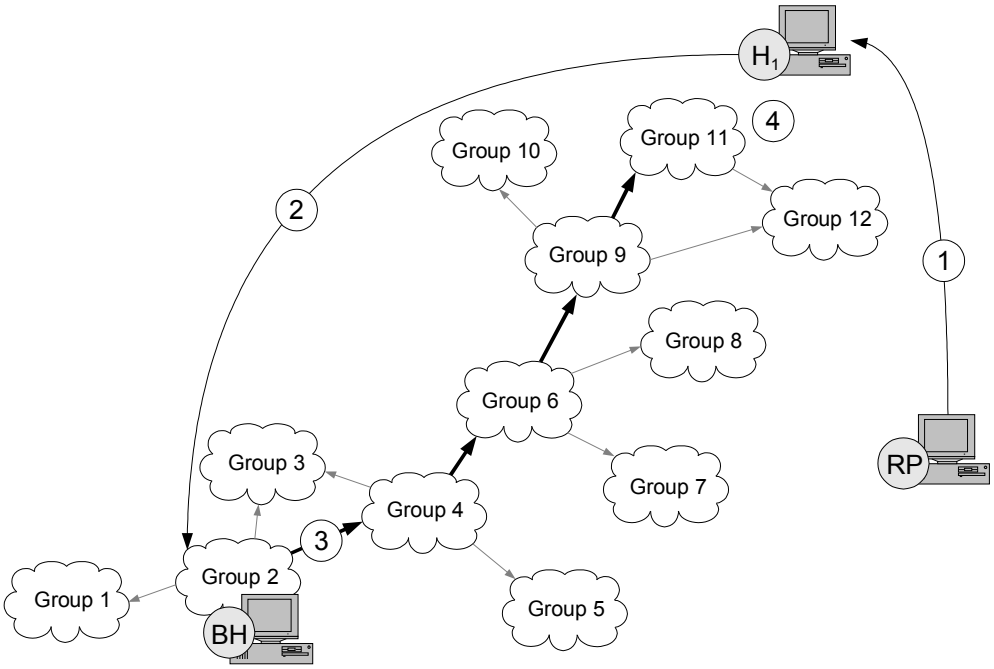

Abbildung 5: Lokalisierung bei mOverlay

Bevor ein Host einer Gruppe beitreten kann, erfolgt in einem Lokalisierungsprozess die Ermittlung der nächstgelegenen Gruppe. Diese Lokalisierung basiert auf Distanzmessungen und benötigt einen globalen Host Cache, den Rendezvous Point (RP) des mOverlay-Netzes. Dieser verwaltet eine Liste von Hosts, die als Startpunkt zur Lokalisierung für die einzelnen Hosts dienen. Der Rendezvous Point weist den neuen Hosts einzelne Boot Hosts aus der Liste zu, in Abbildung 5 als

11 Die lokalen Gruppen bei mOverlay entsprechen den lokalen Clustern aus der vorliegenden Arbeit.

Schritt 1 dargestellt. Die neuen Hosts fragen die Liste der Nachbargruppen des Boot Hosts ab (Group 1, Group 3, Group 4) und führen eine Distanzbestimmung zu diesen Gruppen durch (Schritt 2). Die gefundene nächstgelegene Gruppe (Group 4) wird nun als Startpunkt für die weitere Lokalisierung verwendet (Schritt 3). Dieser Schritt wird iterativ wiederholt, bis keine näher liegende Gruppe mehr gefunden oder ein Abbruchkriterium (z.B. Anzahl der Durchläufe) erreicht wird. Anschließend versucht der neue Host in Schritt 4 der nächstgelegenen Gruppe beizutreten.

Das mOverlay-Verfahren lässt sich wegen der Methode des „Durchlaufens“ der einzelnen Gruppen und dem Ermitteln der jeweiligen Distanz durch den neu hinzukommenden Host sehr gut in P2P-Umgebungen einsetzen. Als Nachteil stellt sich dabei aber die wiederholte Distanzermittlung zu allen Nachbargruppen heraus, die für alle besuchten Gruppen durchgeführt wird und ihre Ursache in der fehlenden globalen Sicht der Hosts auf das Gesamtnetzwerk hat. Bei Verfahren wie GNP mit globalen Koordinatensystemen können Hosts die Abstände zu allen bekannten Clustern bestimmen, ohne die jeweiligen Distanzen zu messen.

3.3.3 CAN

Das von S. Ratnasamy und anderen in [RAT02] vorgestellte Verfahren CAN verwendet wie mOverlay die durch Austausch von ICMP-Echo-Nachrichten ermittelten Distanzinformationen zur Clusterbildung.

Die Zielsetzung beim Entwurf von CAN war die Entwicklung eines einfachen Verfahrens, das schnell und verteilt arbeitet, um eine Skalierbarkeit bis zu mehreren Millionen Hosts zu ermöglichen. Dazu wird ein verteiltes System auf der Basis von bins (Körben) eingesetzt. Die Hosts ordnen sich anhand der ermittelten Distanzinformationen selbständig den einzelnen Körben zu, wobei alle Hosts in einem Korb aus Netzwerksicht relativ nah beieinander liegen und einen Cluster bilden.

Das CAN-Verfahren verwendet wie GNP feste Landmarks, die allen Hosts bekannt sein müssen. Neue Hosts bestimmen zuerst die Latenzzeiten zu allen Landmarks. Die dabei möglichen Werte für die Latenzzeit werden in Level unterteilt, z.B. Level 0 für Latenzzeiten von 0 bis 100ms. Der Host ordnet nun alle ermittelten Werte den vordefinierten Levels zu und bildet eine Kombination aus den Leveln anhand einer vorgegebenen Reihenfolge. Diese Kombination gibt direkt den zugehörigen Korb für den Host an.

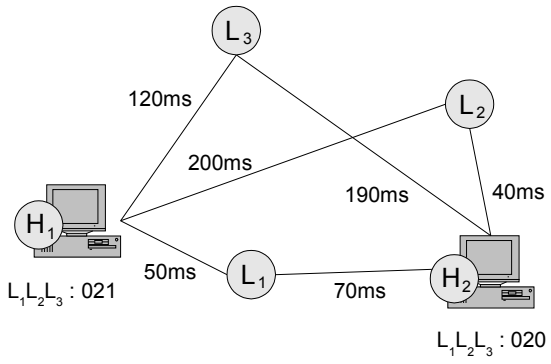

Abbildung 6: Funktionsweise von CAN

In Abbildung 6 ist eine mögliche Netzwerkstruktur abgebildet. Der Host H_1 bestimmt die Latenzzeiten zu den Landmarks L_1 bis L_3 , die Distanzen betragen $l_1 = 50\,ms$, $l_2 = 200\,ms$ und $l_3 = 120\,ms$. Die Level sind wie folgt vordefiniert: Level 0 umfasst die Latenzzeiten von 0 bis 100ms, Level 1 von 101ms bis 150ms sowie Level 2 von 151ms bis 200ms. Längere Latenzzeiten werden in diesem Beispiel nicht betrachtet, die Definition weiterer Level erfolgt aber analog zum vorgestellten Prinzip. Mit den vorgegebenen Leveln ermittelt der Host H_1 nun die Kombination nach der definierten Reihenfolge der Landmarks $L_1 L_2 L_3$ und ordnet sich damit dem Korb 021 zu.

Die im Rahmen des CAN-Verfahrens praktizierte selbständige Zuordnung von Hosts zu den einzelnen Körben ermöglicht eine schnelle Bestimmung des jeweiligen Clusters mit wenigen Distanzermittlungen bei hoher Skalierbarkeit. Die angewendete Methode ist sehr einfach und benötigt keine komplizierten Berechnungen oder globales Wissen über das Netzwerk. Die Kommunikation von neuen Hosts erfolgt ausschließlich mit den Landmarks, Kommunikation mit Hosts in den einzelnen Gruppen ist nicht notwendig.

Nachteilig wirkt sich bei diesem Verfahren die Abhängigkeit von festen Landmarks im System aus. Die Anforderung an das zu entwickelnde System, dass neben den Distanzinformationen auch weitere Kriterien für die Clusterzuordnung nutzbar sein sollen, kann mit dem Ansatz von CAN nicht umgesetzt werden. Dazu müsste das Verfahren die Sortierung der Gruppen anhand der Distanz zum Host unterstützen. Bei CAN existiert diese Funktion nicht, es ist nur die direkte Auswahl eines Korbes für den Host möglich. Die Definition der Level stellt eine weitere Schwierigkeit bei der Anwendung von CAN dar. Bei kleinen Netzen mit weit verteilten Hosts funktioniert eine Gruppenbildung nur mit wenigen Leveln, die aber große Zeitspannen abdecken müssen. Ist das Netz dagegen sehr dynamisch, sind die dabei entstehenden Gruppen nur schwer vorhersehbar.

3.3.4 NICE

Im Rahmen des NICE-Projektes[12] stellten Suman Banerjee, Bobby Bhattacharjee und Christopher Kommareddy in „Scalable Application Layer Multicast“ [BAN02] neben verschiedenen Multicast-Protokollen einen Clusteringmechanismus für Anwendungen mit großer Empfängerzahl und geringen Bandbreitenanforderungen vor. Diese Beschränkung auf bestimmte Bandbreiten dient aber in dieser Arbeit nur zum Herausstellen der geringen Steuerinformationen, die bei NICE benötigt werden. Das vorgestellte Clusteringverfahren wurde dann von den Autoren in der Arbeit „Scalable Peer Finding on the Internet“ [KOM02] unter der Bezeichnung Tiers von den Multicast-Protokollen getrennt.

Das grundlegende Prinzip von Tiers ist die Suche nach dem nächstgelegenen Host durch eine top-down-Suche. Ausgehend von einem festen Host wird die Suche in einer baumartigen Hierarchie verfeinert, bis der nächstgelegene Host gefunden wird. Die verwendete Hierarchie besteht aus mehreren Ebenen, in denen die Hosts des Netzwerkes angeordnet sind. Die unterste Ebene L_0 enthält alle Hosts des Netzwerks.

12 Der Begriff NICE steht für ein Akronym: NICE is the Internet Cooperative Environment

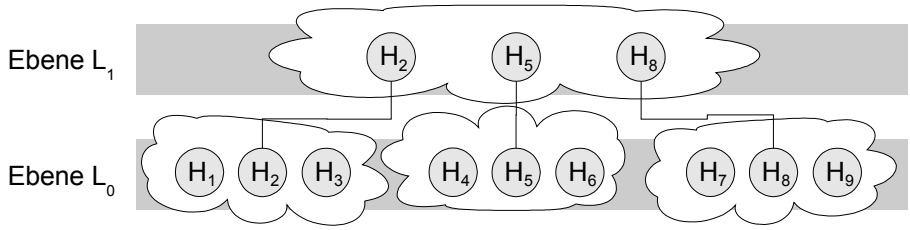

Abbildung 7: Tiers-Hierarchie mit Clustern

Die Hosts einer Ebene werden in Cluster unterteilt, wobei Hosts in einem Cluster aus Netzwerksicht nahe beieinander liegen. Jeder Cluster bestimmt den Host mit minimalstem Abstand zum Mittelpunkt des Clusters als Clusterleader. Die Clusterleader aller Cluster einer Ebene treten der nächsthöheren Ebene in der Hierarchie bei und bilden auf dieser wiederum Cluster. Das Verfahren der Clusterbildung und der Auswahl von Clusterleadern wird für alle Ebenen durchgeführt, bis nur noch ein Host auf der höchsten Ebene vorhanden ist.

Für die Hosts in einem Tiers-Netzwerk gelten folgende Eigenschaften:

- Ein Host gehört immer nur zu einem Cluster pro Ebene der Hierarchie.
- Wenn ein Host Teil eines Cluster auf der Ebene L_i ist, so muss er immer auch einem Cluster auf den Ebenen $L_{0,}\ldots, L_{i-1}$ als Clusterleader angehören.
- Ist ein Host nicht auf Ebene L_i vorhanden ist, kann er nicht Teil aller Ebenen L_j mit $j>i$ sein.
- Die maximale Ebenenzahl ist beschränkt auf $\log_k N$. Die höchste Ebene L_k hat nur ein Mitglied.
- Die Größe eines Clusters ist beschränkt auf mindestens k und maximal $2k-1$ Hosts, wobei k eine Konstante für das System ist.

Die Ermittlung des nächstgelegenen Hosts setzt einen bekannten Bootstrap-Host (BSH) voraus, der durch einen neuen Host angefragt wird. Der BSH verwaltet die Liste mit den Clusterheads auf der höchsten Ebene und übermittelt diese Liste an den neuen Host. Der neue Host bestimmt nun den nächstgelegenen Host aus der übermittelten Liste und erhält von diesem die Liste aller Hosts des Clusters, für die der gefundene Host Clusterleader ist. Diese Aktion führt der neue Host nun für alle verbleibenden Ebenen der Hierarchie durch, bis der nächstgelegene Host gefunden wird. Alternativ kann das Verfahren auch stoppen, sobald der nächstgelegene Host auf der Ebene L_1 und somit der nächstgelegene Cluster gefunden wurde.

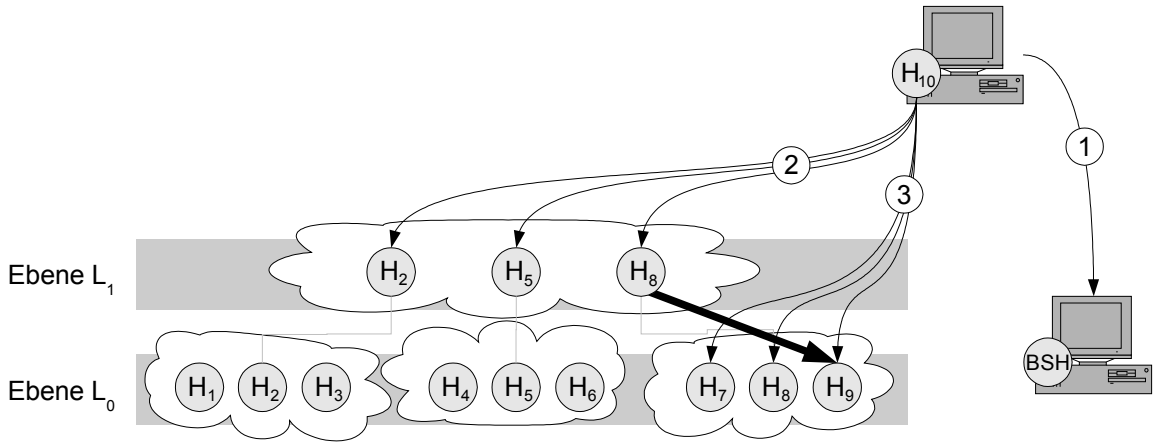

Abbildung 8: Suche des nächstgelegenen Hosts mit Tiers

Abbildung 8 stellt die von einem neuen Host ausgeführten Aktionen in einem Overlay-Netzwerk mit zwei Ebenen $k=2$ vereinfacht dar. Im ersten Schritt wird der Bootstrap-Host angefragt, der dem neuen Host die obersten Einträge in der Hierarchie zurückgibt. Nach der Distanzbestimmung zwischen H_{10} und den Hosts der Ebene L_1 im Schritt 2 wird der nächstgelegene Host H_8 angesprochen. H_8 liefert eine Liste der Hosts, für die er Clusterleader auf der Ebene L_0 ist. Anschließend wird die Distanz zu diesen Hosts ermittelt, bei der im Beispiel H_9 als nächstgelegener Host gefunden wird. Das Verfahren endet an dieser Stelle, denn H_9 ist nur auf der untersten Ebene L_0 vorhanden und somit kein Clusterleader. Der neue Host H_{10} hat damit den nächstgelegenen Host gefunden und tritt dem Cluster ebenfalls bei.

Die im Abfrageprozess zur Ermittlung des nächstgelegenen Hosts durch die übertragenen Verwaltungsinformationen erzeugten Kosten betragen $O(k \log N)$.

Tiers arbeitet grundsätzlich wie mOverlay (siehe Abschnitt 3.3.2) mit dem Durchlaufen der Cluster bis zum nächstgelegenen Host, verwendet aber im Unterschied dazu eine fest vorgegebene Hierarchie. Als Nachteil gegenüber mOverlay wirkt sich deshalb die Verwendung eines Hosts als Startpunkt in der obersten Ebene aus. Alle neuen Hosts, die dem Netzwerk beitreten, kontaktieren immer diesen Clusterleader. Durch die hierarchische Anordnung der Hosts bietet Tiers aber auch Möglichkeiten, die Suche des nächstgelegenen Hosts insbesondere bei Verwendung von Koordinaten zu optimieren. Die Clusterleader können die Koordinaten aller Hosts des Clusters speichern und an den neuen Host übermitteln. Dieser muss nun keine Distanzmessungen mehr durchführen sondern kann direkt den nächstgelegenen Host im Cluster auswählen und mit der Suche fortfahren.

Die Skalierbarkeit des Systems ist bei Verwendung von Tiers durch die Verteilung eines Großteils der Hosts auf den unteren Ebenen der Hierarchie sichergestellt. Die Hosts verwalten somit nur Verbindungen zu einer konstanten durch die maximale Clustergröße vorgegebenen Anzahl von anderen Hosts. Wie bereits beschrieben enthält ein Cluster zwischen k und $2k-1$ Hosts, die beste Performance wurde in der Arbeit mit einem Wert von $k=12$ erreicht. Die Hosts auf der obersten Ebene verwalten den Status von insgesamt $O(\log N)$ Hosts.

3.3.5 Gewählter Ansatz

In den vorangegangenen Abschnitten wurden mehrere Verfahren zum Clustering von Hosts im Internet vorgestellt und hinsichtlich der definierten Anforderungen untersucht.

Für den Einsatz im zu entwickelnden Algorithmus kommen nur die auf Distanzmessung per Echo-Nachrichten basierenden Ansätze in Frage. Die weiteren vorgestellten Lösungen benötigen zusätzliche Netzwerkinformationen, die aber nicht in allen Einsatzbereichen verfügbar sind. Diese Problematik tritt unter anderem bei den Ansätzen mit Nutzung von Routinginformationen aus dem Border Gateway Protocol (BGP) auf. Diese Daten sind ausschließlich in Umgebungen mit Internet-Routern verfügbar, weshalb der Einsatz nur im Internet oder in Intranets möglich ist.

Die vorgestellten Verfahren mOverlay, CAN und NICE ermitteln die Distanzen für die Clusterzuordnung mit Echo-Nachrichten, womit alle drei Methoden prinzipiell verwendbar sind. Die Eignung der einzelnen Ansätze für den zu entwickelnden Algorithmus ist jedoch sehr unterschiedlich.

mOverlay ist aufgrund seiner vollständigen Unabhängigkeit von zentralen Systemen und vordefinierten Strukturen gut in P2P-Umgebungen anwendbar, benötigt aber für die Ermittlung des nächstgelegenen Clusters eine große Anzahl von Messungen. Zudem erweist sich das Fehlen von Strukturen bei Änderungen der Netzwerkstruktur als großer Nachteil von mOverlay. Bei einer Störung oder Änderung des Overlay-Netzwerkes müssen alle Hosts ihre Position innerhalb des Netzwerks erneut bestimmen, was aufgrund der großen Anzahl von Distanzmessungen erhebliche Netzwerklast und eine lange Laufzeit bedeutet.

Das CAN-Verfahren verwendet im Gegensatz zu mOverlay und NICE feste Landmarks für die Zuordnung von Hosts zu den einzelnen Körben. Während des gesamten Zeitraums, in dem das Overlay-Netzwerk verwendet wird, müssen die Landmarks verfügbar sein. Fallen dagegen einzelne Landmarks aus oder ändern sich die Positionen von Landmarks, so werden alle Zuordnungen von Hosts zu den Körben ungültig.

Von den vorgestellten Verfahren erweist sich NICE aufgrund der implementierten Hierarchie als sinnvolle Basis für den zu entwickelnden Algorithmus. Der NICE-Ansatz verwendet wie mOverlay Distanzmessungen zu allen Clusterheads, um die nächstgelegene Gruppe im Overlay-Netzwerk zu finden. Somit bieten sich im direkten Vergleich zwischen beiden Verfahren bis auf die vorhandene baumartige Hierarchie keine Vorteile; beide Systeme müssen für das „Durchwandern“ des Overlay-Netzwerks oder der Hierarchie wiederholte Distanzmessungen durchführen.

In Verbindung mit einer Positionsbestimmung über Koordinaten mit GNP oder PIC kann aber das Finden des nächstgelegenen Clusters bei NICE deutlich optimiert werden. Die Distanzermittlung während des Abfrageprozesses erfolgt dabei nicht mehr mit Messungen per Echo-Nachrichten sondern über die Berechnung der Entfernungen unter Verwendung der Koordinaten.

Der erweiterte Ansatz aus Nutzung von Koordinaten und der Hierarchie des NICE-Verfahrens bietet einen deutlichen Vorteil hinsichtlich der Skalierbarkeit des Systems. Auch bei großen Netzwerken ist das Auffinden des nächstgelegenen Clusters mit einer konstanten Anzahl von Distanzmessungen möglich. Die berechneten Koordinaten eines Hosts sind anschließend fest und müssen bei Änderungen an der Clusterstruktur nicht neu berechnet werden.

4 Entwurf

Der im Rahmen dieser Arbeit erstellte Entwurf ist eine Kombination aus koordinatenbasierter Positionsbestimmung und einem hierarchischen Verfahren zur Bildung und Verwaltung der Cluster. Diese Aufteilung wurde wegen ihrer guten Skalierbarkeit gewählt, wodurch das Verfahren auch für große P2P-Umgebungen mit vielen Hosts geeignet ist.

Die Koordinaten eines Hosts können bei diesem Ansatz unabhängig von der Gesamtzahl der Hosts im Overlay-Netzwerk immer mit einer konstanten Anzahl von Distanzmessungen ermittelt werden. Das „Durchwandern“ des Overlay-Netzwerks wie bei mOverlay ist dagegen mit wiederholten Distanzmessungen verbunden, deren Anzahl bei Erweiterung des Netzwerkes immer weiter zunimmt.

Durch den Einsatz einer Hierarchie für die Verwaltung der Cluster lässt sich die Skalierbarkeit des Systems auch im Betrieb sicherstellen. Die Hosts in den Clustern speichern nur Informationen über eine beschränkte Anzahl von anderen Hosts, können aber aufgrund der implementierten Hierarchie leicht zusätzliche Informationen oder weitere Hosts ermitteln.

4.1 Systemkomponenten

Neben den einzelnen Hosts verwendet das System zwei zentrale Dienste, um neue Hosts in das System aufzunehmen. Die Hosts benötigen einen Bootstrap-Host, der den Zugangspunkt zum System darstellt. Dazu verwaltet der Bootstrap-Host Verweise auf den Clusterhead im obersten Cluster der Hierarchie sowie auf einen Lokalisierungsdienst. Mit Hilfe des Lokalisierungsdienstes kann ein Host beliebige Hosts im Netzwerk auffinden oder eine bestimmte Anzahl von Hosts aus dem Netzwerk zufällig ermitteln. Der Lokalisierungsdienst stellt somit die Basis bei der Ermittlung der Koordinaten eines Hosts dar, indem er diesem die benötigten Landmarks übermittelt.

4.2 Bestimmung der Koordinaten

In Kapitel 3 wurden verschiedene Verfahren zur Abbildung von Hosts in einem Koordinatensystem anhand ihrer Distanzen vorgestellt. Von den verschiedenen Ansätzen bietet sich für den geplanten Einsatzbereich einer P2P-Umgebung die Nutzung von PIC an. Es erweitert das ebenfalls vorgestellte GNP um die Möglichkeit, Koordinaten von Hosts ohne feste Referenzpunkte im Overlay-Netzwerk zu berechnen. Insbesondere in P2P-Umgebungen kann eine dauerhafte Verfügbarkeit der als Referenzpunkte eingesetzten Hosts nicht garantiert werden. Die vorhandenen Hosts können aber für die Ermittlung von Koordinaten bei PIC eingesetzt werden.

Im Abschnitt 3.2.4 wurden drei Möglichkeiten zur Wahl der Landmarks vorgestellt: die zufällige Auswahl von Hosts als Landmarks, die Verwendung der nächstgelegenen Hosts und ein hybrider Ansatz. Da nur das hybride Verfahren ausreichend genaue Koordinaten für die Hosts bestimmen kann, wird diese Lösung im folgenden Entwurf eingesetzt.

Die Berechnung der Koordinaten erfolgt in zwei Schritten. Im ersten Schritt werden die Koordinaten näherungsweise bestimmt, die dann im zweiten Schritt verfeinert werden. Dazu ermittelt der neue Host vom Lokalisierungsdienst eine feste Anzahl von Landmarks. Die benötigte Anzahl dieser

Landmarks ist von der Dimensionszahl des zur Einbettung verwendeten Koordinatensystems abhängig und wird im Kapitel 5 definiert. Der Host misst nun die Distanzen zu den ermittelten Landmarks und berechnet seine groben Koordinaten. Von den nächstgelegenen Landmarks werden die Koordinaten von allen Hosts in den jeweiligen lokalen Clustern abgefragt. Der Host berechnet die Distanzen zu den Hosts mit deren Koordinaten sowie den eigenen groben Koordinaten und wählt die nächstgelegenen Hosts aus. Abschließend wird mit diesen Hosts und den im ersten Schritt zufällig ausgewählten Hosts eine erneute Koordinatenberechnung durchgeführt. Die dabei berechneten Koordinaten weisen deutlich geringere Abweichungen als die groben Koordinaten auf (siehe [COS03]). Die berechneten Koordinaten der Hosts bleiben für die gesamte Aufenthaltszeit im Overlay-Netzwerk gültig und müssen auch bei Wegfall der verwendeten Landmarks nicht neu berechnet werden. Für Einsatzbereiche, in denen die Distanzen zwischen den einzelnen Hosts nicht konstant sind, können die Koordinaten periodisch neu berechnet werden. Notwendig sind dazu die Wahl fester Landmarks in diesem System, die bei der Neuberechnung von Koordinaten als Bezugspunkte für die übrigen Hosts dienen. Die Optimierung des vorgestellten Entwurfs und der Implementierung für Einsatzbereiche mit veränderlichen Distanzen wird in dieser Arbeit nicht vorgenommen sondern kann im Rahmen einer Weiterentwicklung des Entwurfs betrachtet werden.

4.3 Clusterbildung und Verwaltung

Als Grundlage für die Clusterbildung und zur weiteren Verwaltung der Cluster wird in diesem Entwurf das vorgestellte NICE-Verfahren eingesetzt. Dieses Verfahren basiert auf einer baumartigen Hierarchie beim Clustering der Hosts. Die Cluster gruppieren die Hosts anhand ihrer Distanzen untereinander.

Durch die Nutzung einer Hierarchie können die Bandbreitenanforderungen bei der Clusterverwaltung gegenüber einem unstrukturierten System deutlich verringert werden. Zudem müssen die Hosts immer nur einen kleinen Teil der Hierarchie kennen, können aber durch Abfragen innerhalb dieses Teils zusätzliche Informationen erlangen.

Die durch NICE (siehe Abschnitt 3.3.4) gebildete Hierarchie weist folgende Eigenschaften auf:

- Die Hierarchie besteht aus den Ebenen L_0 bis L_n.
- Die unterste Ebene L_0 enthält alle Hosts des Overlay-Netzwerks, die nach ihren Distanzen untereinander in lokale Cluster unterteilt sind.
- Der dem Mittelpunkt des Cluster am nächsten gelegene Host dient als Clusterleader und bildet mit den anderen Clusterleadern auf der Ebene L_1 ebenfalls lokale Cluster.
- Dieses Verfahren wird auch für die weiteren Ebenen fortgesetzt, bis auf der obersten Ebene nur ein lokaler Cluster existiert. Der Clusterleader dieses Clusters ist der Startpunkt für die baumartige Hierarchie.
- Die Cluster besitzen eine Größe von mindestens k und maximal $2k-1$, wobei k eine Konstante für das jeweilige System ist. Sobald ein Cluster aufgrund von neuen Hosts oder

dem Wegfall von Hosts diese Grenzen nicht mehr einhält, werden automatisch zwei kleine Cluster zusammengefasst oder ein großer Cluster wird in zwei kleinere Cluster geteilt.

4.3.1 Unterschiede zum NICE-Ansatz

Für den Entwurf wird eine Hierarchie nach den Prinzipien von NICE eingesetzt, die Algorithmen werden jedoch abgewandelt oder vollständig durch eigene Methoden ersetzt.

NICE verwendet für die Distanzermittlung immer direkte Messungen, die aber im Rahmen dieses Entwurfs nicht mehr nötig sind. Durch die einmalige Berechnung von festen Koordinaten für die Hosts kann bei der Distanzermittlung auf diese Werte zurückgegriffen werden. Die Hosts fragen dabei nur die Koordinaten des entfernten Hosts ab und berechnen dann selbständig mit Hilfe ihrer eigenen Koordinaten die Entfernung.

Mit dem Einsatz der Koordinaten muss nicht mehr zwangsläufig der dem Mittelpunkt nächstgelegenen Host als Clusterleader verwendet werden. Im gewählten Ansatz kann vielmehr ein beliebiger Host als Clusterleader dienen. An einen neuen Host werden zur Distanzermittlung nicht die wirklichen Koordinaten dieses Clusterleaders übermittelt, sondern die Koordinaten des Mittelpunktes eines Clusters. Dessen Werte werden durch den jeweiligen Clusterleader aus den Koordinaten aller Hosts in einem Cluster berechnet und periodisch aktualisiert. An dieser Stelle sei auf den hierarchischen Aufbau von NICE verwiesen, bei dem ein Host als Clusterleader für Cluster auf mehreren Ebenen der Hierarchie dienen kann. Die Koordinaten der Mittelpunkte der einzelnen Cluster sind aber nicht zwangsläufig identisch, weshalb bei der Implementierung eine Lösung zur Unterscheidung der Cluster auf den einzelnen Ebenen gefunden werden muss.

Durch den Einsatz von Koordinaten kann auch der Ablauf beim Verlassen von Clusterleadern gegenüber dem NICE-Ansatz optimiert werden. Bei NICE muss im betroffenen Cluster auf der untersten Ebene L_0 ein neuer Clusterleader gewählt werden, der dann allen Ebenen ausgehend vom obersten Cluster der Hierarchie erneut beitreten muss. Dieser Ablauf ist zeitaufwendig und benötigt bei NICE wiederholte Distanzberechnungen sowie mehrere Neuwahlen von Clusterleadern. Mit dem koordinatenbasierten Verfahren entfällt die Notwendigkeit, immer den Host mit der geringsten Entfernung zum Mittelpunkt als Clusterleader auszuwählen. Jeder betroffene Cluster, bei dem ein Clusterleader nicht mehr verfügbar ist, kann nun unabhängig von der restlichen Hierarchie einen beliebigen Host aus dem Cluster als neuen Clusterleader auswählen. Dieser Host tritt als Clusterleader dem übergeordneten Cluster der nächsthöheren Ebene bei. Aufgrund der periodischen Neuberechnung der Koordinaten des Mittelpunktes werden die Änderungen am Cluster nach einer bestimmten Zeitspanne auch in der Hierarchie bekannt.

4.3.2 Ein neuer Host im Overlay-Netzwerk

Die Aufnahme eines neuen Hosts in das Overlay-Netzwerk und die Zuordnung zu einem der Cluster erfolgt in zwei Phasen: der Koordinatenberechnung und dem Clusterbeitritt.

Die bei der Koordinatenberechnung benötigten Distanzen zwischen den Hosts werden mit Echo-Nachrichten ermittelt, da dieses Verfahren deutlich genauere Ergebnisse als die weiteren im Abschnitt 3.1 betrachteten Verfahren liefert. Für die Abbildung der Hosts auf Punkte in einem Koordinatensystem kommt der im GNP-Ansatz vorgestellte Algorithmus zum Einsatz. Im ersten Schritt werden die Hosts auf beliebige Punkte im Koordinatensystem abgebildet. Mit dem Simplex-Downhill-Algorithmus ([NEL65]), einem globalen multidimensionalem Minimierungsverfahren, werden anschließend die Punkte im Koordinatensystem verschoben, um den Gesamtfehler zwischen den gemessenen Distanzen zu den Hosts im Overlay-Netzwerk und den berechneten Entfernungen zwischen den Punkten im Koordinatensystem zu minimieren.

Um die Berechnung durchführen zu können, ermittelt ein neuer Host die Landmarks und deren Koordinaten über den Lokalisierungsdienst. Sind für eine genaue Berechnung der Koordinaten nicht genügend Landmarks vorhanden, misst der neue Host seine Distanz zu allen Landmarks und erfragt zusätzlich die Distanzen aller Landmarks untereinander. Mit dem Simplex-Downhill-Algorithmus bestimmt der neue Host nun neue Koordinaten für sich und alle Landmarks und übermittelt die neuen Koordinaten an die anderen Hosts.

Sobald eine ausreichende Anzahl von Hosts als Landmarks im Overlay-Netzwerk vorhanden ist, bestimmt ein neuer Host nur seine eigenen Koordinaten, wobei ebenfalls der Simplex-Downhill-Algorithmus eingesetzt wird. Mit den gemessenen Distanzen fragt der neue Host alle Hosts in den Clustern der nächstgelegenen Landmarks ab, berechnet mit deren Koordinaten und seinen groben Koordinaten die Distanz und wählt dann die nächstgelegenen Hosts als zusätzliche Landmarks aus. Der Host misst im nächsten Schritt die Distanzen zu den neuen Landmarks und berechnet dann seine genauen Koordinaten mit den Distanzen zu allen Landmarks.

Mit den berechneten Koordinaten kann der neue Host den nächstgelegenen Cluster in der Hierarchie suchen und diesem beitreten. Dazu durchwandert er die Clusterhierarchie vom Cluster auf der obersten Ebene L_n bis zur Ebene L_0. Für eine Ebene L_i in der Hierarchie wird dabei der folgende Algorithmus durchgeführt:

- Ermittle für den auf Ebene L_{i+1} gefundenen nächstgelegenen Host – dem Clusterleader des Cluster auf Ebene L_i – die Koordinaten aller Hosts.
- Berechne die Distanz mit den eigenen Koordinaten und den ermittelten Koordinaten.
- Wähle den nächstgelegenen Host als Clusterleader für die Ebene L_{i-1} aus und fahre auf der Ebene L_{i-1} fort.

Der Algorithmus endet, sobald auf der Ebene L_1 ein Clusterleader für den nächstgelegenen Cluster auf der Ebene L_0 gefunden wurde.
Zum Beitritt in den gefundenen nächstgelegenen Cluster sendet der neue Host eine Anfrage zur Aufnahme an den Clusterleader. Dieser kann den neuen Host in den Cluster aufnehmen oder

dessen Anfrage ablehnen, wenn der neue Host vordefinierte Bedingungen nicht erfüllt. In diesem Fall kann der neue Host in den jeweils nächstgelegenen Clustern auf den Ebenen L_1 bis L_n mit den bereits ermittelten Koordinaten weitere nahe liegenden Cluster finden und versuchen, diesen beizutreten.

4.3.3 Hosts als Teil eines Clusters

Die Hosts in einem Cluster versenden periodisch Nachrichten an alle anderen Hosts im Cluster, um diese über ihren Status zu informieren. Die Nachrichten enthalten zudem die Koordinaten des jeweiligen Hosts.

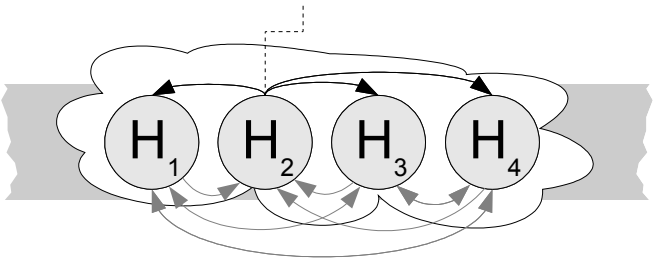

Abbildung 9: Informationsfluss in einem Cluster in der Hierarchie. Der Clusterleader H_2 informiert periodisch die anderen Hosts über alle Hosts im aktuellen Cluster und dem übergeordneten Cluster. Die Hosts tauschen ihre Daten untereinander und mit dem Clusterleader aus.

Der Clusterleader sendet neben den Statusnachrichten zusätzlich periodisch die Koordinaten sämtlicher Hosts im Cluster an alle Hosts in diesem Cluster. Außerdem informiert er die Hosts über die Hosts im Cluster auf der nächsthöheren Ebene, zu dem er als Clusterleader ebenfalls gehört. Somit kennen die Hosts in einem Cluster immer alle anderen Hosts und deren Koordinaten im Cluster sowie zusätzlich alle Hosts im übergeordneten Cluster.

4.3.4 Ein Host verlässt das Overlay-Netzwerk

Für das Verlassen des Clusters durch einen Host existieren verschiedene Möglichkeiten, die jeweils unterschiedliche Reaktionen durch die verbliebenen Hosts erfordern:

- ein Clusterleader verlässt das Overlay-Netzwerk und kündigt das Verlassen an
- ein Clusterleader fällt aus
- ein normaler Host verlässt das Overlay-Netzwerk und meldet dies dem Clusterleader
- ein normaler Host fällt aus.

Die Abläufe für diese Möglichkeiten werden im folgenden Abschnitt erläutert.

Ein Clusterleader verlässt Overlay-Netzwerk mit Abmeldung

Ein Clusterleader, der sich aus dem Overlay-Netzwerk abmelden möchte, wählt aus den verbleibenden Hosts nach bestimmten Kriterien einen neuen Clusterleader aus. Dazu sendet er an diesen Host eine Nachricht, die neben der Information über die Ernennung auch alle benötigten Daten zum Cluster enthält. Dazu zählen die Koordinaten und Adressen der einzelnen Hosts im Cluster und im übergeordneten Cluster, zu dem der Host aufgrund seiner Funktion als Clusterleader ebenfalls gehörte.

Der bisherige Clusterleader informiert anschließend die Hosts im aktuellen und im übergeordneten Cluster über den neuen Clusterleader, der damit direkt in die Hierarchie eingebunden ist.

Ein Clusterleader fällt aus

Die Hosts in einem Cluster können den Ausfall eines Clusterleader durch das Ausbleiben der periodischen Nachrichten erkennen. Zur Sicherheit wird in diesem Fall die Verfügbarkeit des Clusterleaders durch die einzelnen Hosts per Echo-Nachrichten erneut überprüft.

Bei einem Ausfall des Clusterleaders müssen die Hosts in einem Cluster selbständig den neuen Clusterleader bestimmen und die Hosts im übergeordneten Cluster über die Änderung informieren. Dazu verwenden sie die Daten aus den periodisch vom bisherigen Clusterleader empfangenen Statusnachrichten, die auch Angaben zu den Hosts im übergeordneten Cluster enthielten.

Als neuer Clusterleader wird auch bei einem Ausfall der jeweils älteste Host im Cluster bestimmt, der dann die Information über den neuen Clusterleader an die Hosts im Cluster und im übergeordneten Cluster sendet. Die Koordinaten des Clusters werden im Zuge der Clusterverwaltung später neu berechnet, um auch die Koordinaten des Mittelpunktes des Clusters an den Wegfall des bisherigen Clusterleaders anzupassen. Mit den periodischen Statusinformationen erhalten auch die übrigen Hosts nach kurzer Zeit die aktualisierten Koordinaten und eine vollständige Liste mit den im Cluster verbliebenen Hosts.

Ein normaler Host verlässt das Netzwerk

Zum ordnungsgemäßen Verlassen des Cluster sendet ein Host eine Nachricht an den Clusterleader, der dann diesen Host aus dem Cluster entfernt und den verbliebenen Hosts durch das Senden von Statusnachrichten die geänderte Clusterstruktur bekannt gibt. Die Koordinaten des Mittelpunktes für diesen Cluster werden später im Rahmen der allgemeinen Verwaltung durch den Clusterleader aktualisiert und mit den periodisch versandten Statusnachrichten an die Hosts im Cluster übermittelt.

Verlässt ein Host einen Cluster auf einer höheren Hierarchieebene als L_0, so wird die Änderung in dem Cluster durch die einzelnen Hosts mit den regulären Statusnachrichten auch an die Hosts in den untergeordneten Clustern übermittelt.

Ausfall eines Hosts

Durch das dauerhafte Fehlen der periodisch versendeten Statusnachrichten erkennen die verbliebenen Hosts den Ausfall eines Hosts im Cluster. Sie reagieren nun selbständig auf die Änderung der Clusterstruktur und entfernen alle Verweise auf diesen Host aus ihren Daten. Der Clusterleader erkennt ebenfalls den Wegfall des Hosts und übernimmt diese Änderung anschließend in die Statusnachrichten.

Der weitere Ablauf entspricht den bereits im vorangegangenen Abschnitt 4.3.3 geschilderten Schritten.

4.3.5 Das Aufteilen und Vereinigen von Clustern

Das NICE-Verfahren definiert eine obere und untere Grenze für die Größe der Cluster (siehe Abschnitt 3.3.4). Sobald die obere Grenze überschritten wird, teilt der Clusterleader den Cluster in zwei kleinere Cluster auf. Ist ein Cluster dagegen zu klein, wird eine Clustervereinigung mit dem nächstgelegenen Cluster durchgeführt.

Das Aufteilen eines Clusters

Die maximale Größe eines Clusters im NICE-Ansatz ist durch $2k-1$ für eine Systemkonstante k vorgegeben. Für die Implementierung wird diese Grenze auf $3k-1$ erhöht, um nicht sofort nach dem Aufteilen eines Clusters wieder eine Vereinigung mit anderen Clustern durchführen zu müssen. Ein Cluster wird beim Überschreiten der oberen Grenze durch den Clusterleader in zwei Teilcluster aufgeteilt, zu denen die Hosts anhand der Distanzen untereinander jeweils zugeordnet werden. Das Finden einer Aufteilung aller Hosts, bei der die maximalen Radien in den neuen Clustern minimiert werden, entspricht dem k-Center-Problem (beschrieben in [MOU99]) und ist somit NP-hart. Daher wird in der Implementierung ein Approximationsalgorithmus eingesetzt, der die Hosts mit einer Laufzeit von $O(n \log n)$ sowie einer Approximationsgüte von 2 in die beiden Cluster aufteilt.

Nachdem der bisherige Clusterleader die Hosts den beiden neuen Clustern zugeteilt hat, wählt er einen Clusterleader für den neuen Cluster, zu dem er selbst nicht gehört. Er bleibt aber weiterhin der Clusterleader im zweiten Cluster. Dieses Vorgehen unterscheidet sich von dem in [KOM02] vorgestellten Ansatz, bei dem für beide Cluster jeweils neue Clusterleader bestimmt werden müssen. Dieses Vorgehen wird durch die Nutzung von Koordinaten bei der Distanzberechnung möglich, da Clusterleader in den höheren Layern immer die Koordinaten des Mittelpunktes des untergeordneten Clusters an anfragende Hosts übermitteln. Somit entfällt die Notwendigkeit des NICE-Ansatzes, stets den dem Mittelpunkt eines Clusters nächstgelegenen Host als Clusterleader zu bestimmen.

Der bisherige Clusterleader schließt das Aufteilen des Clusters ab, indem er alle Hosts in den neuen Clustern über die Änderungen informiert und den neuen Clusterleader in den übergeordneten Cluster aufnehmen lässt. Dazu sendet er eine entsprechende Nachricht an den Clusterleader des übergeordneten Clusters.

Das Vereinigen von Clustern

Wird die untere Grenze für die Größe eines Clusters unterschritten, so initiiert der Clusterleader eine Vereinigungsoperation. Dazu sucht er den nächstgelegenen Cluster auf der gleichen Hierarchieebene und informiert den Clusterleader dieses Clusters über den Vereinigungswunsch.

Das Finden des nächstgelegenen Clusters erfolgt dabei mit Hilfe der Koordinaten aller Hosts im übergeordneten Cluster, zu dem der Host durch seine Funktion als Clusterleader ebenfalls gehört. Der angefragte Clusterleader sendet anschließend eine Bestätigungsnachricht an den Clusterleader, woraufhin dieser alle Cluster auf den übergeordneten Hierarchieebenen verlässt (siehe Abschnitt 4.3.3).

5 Die LocalCluster-Implementierung

Das folgende Kapitel gibt einen Überblick über Lösungen und einige ausgewählte Technologien, die bei der Implementierung des vorgestellten Entwurfs eingesetzt wurden. Zudem werden die Beweggründe für die Verwendung der einzelnen Komponenten dargelegt.

5.1 Annahmen

Für den Entwurf auf Basis einer koordinatenbasierten Positionsbestimmung und der hierarchischen Clusterverwaltung sind mehrere grundlegende Annahmen nötig, die im praktischen Einsatz aber nicht immer garantiert werden können. Diese Annahmen und die bei der Implementierung genutzten Lösungen werden im folgenden Abschnitt vorgestellt.

Die Bestimmung der Distanzen zwischen zwei Hosts anhand von berechneten Koordinaten geht von der Annahme aus, dass die bei der Kommunikation genutzten Routen im Netzwerk symmetrisch sind. Diese Voraussetzung ist vor allem im Internet als einem möglichen Einsatzbereichen nicht garantiert, da die verwendeten Routen bei der Kommunikation zwischen zwei Hosts von vielfältigen Faktoren beeinflusst werden, darunter die Routing-Policies der Netzbetreiber und die Verfügbarkeit von Netzwerkverbindungen. Im Entwurf wird dieser Problematik eine erweiterte Distanzmessung entgegen gesetzt, die unabhängige Verbindungen in beide Kommunikationsrichtungen einsetzt und zudem mehrere Messungen durchführt. Aus den ermittelten Werten bestimmt das Verfahren dann einen Mittelwert.

Neben der Problematik von unterschiedlichen Routen in eine einzelne Kommunikationsrichtung basiert der Entwurf zudem auf der Annahme, dass das Routing der Nachrichten effizient und statisch über die kürzesten möglichen Routen erfolgt. Da die einzelnen Host sich aber unter Umständen über einen langen Zeitraum im Overlay-Netzwerk aufhalten, ist diese Annahme im praktischen Einsatz nicht garantiert. Die verwendeten Routen zwischen den Hosts können durch die Netzbetreiber aus Gründen der Lastverteilung oder Kostensenkung geändert werden, wodurch sich die Distanzen der Hosts erheblich ändern können.

Der im Kapitel 4 vorgestellte Entwurf nutzt einen Lokalisierungsdienst für die Auswahl von zufälligen Hosts als Landmarks bei der Koordinatenberechnung. In der aktuellen Implementierung wurde dieser Dienst aus Performancegründen durch einen Bootstrap-Host ersetzt. Dieser ermöglicht die Ermittlung der Landmarks in nur einem Schritt, bei Nutzung eines Lokalisierungsdienstes benötigen neue Host dagegen erst längere Zeit zum Auffinden von Landmarks.

Der Bootstrap-Host verwaltet die Koordinaten einer definierten Anzahl von Hosts in einem Cache und wählt aus diesem Cache die benötigten Landmarks aus. Aus Gründen der Performance und Skalierbarkeit ist die Anzahl der vom Bootstrap-Host zwischengespeicherten Koordinaten beschränkt. Sobald die Kapazität erreicht ist, werden beim Einfügen von neuen Koordinaten die ältesten Einträge aus dem Cache entfernt. Die Größe des Caches muss dabei aber so gewählt werden, dass bei Ausfall oder der Abmeldung von Hosts weiterhin genügend gültige Hosts als Landmarks an die neuen Hosts übermittelt werden können. Die Daten des Caches werden von den

Hosts im Overlay-Netzwerk geliefert, die ihre Koordinaten nach der Berechnung an den Bootstrap-Host übermitteln.

5.2 Systemaufbau

Die Implementierung für einen einfachen Host im Overlay-Netzwerk verwendet mehrere Komponenten zur Trennung der einzelnen Funktionen. Das System besteht dabei aus den folgenden Komponenten:

- *DatagramSender* und *DatagramReceiver* als Schnittstelle zum Netzwerk
- dem *Messenger* als Nachrichtenzentrale des Systems
- dem *EchoManager* und den *EchoSessions* für die Distanzermittlung
- dem *CoordinateService* zur Berechnung und Verwaltung der Koordinaten
- dem *ClusterService* zur Verwaltung der Clusterinformationen
- dem *ClusterWatcher* für die Überwachung der Cluster
- Operationen zum Beitreten, Verlassen, Aufteilen und Vereinigen von Clustern
- sowie der PIC-Implementierung zum Berechnen der Koordinaten der Hosts.

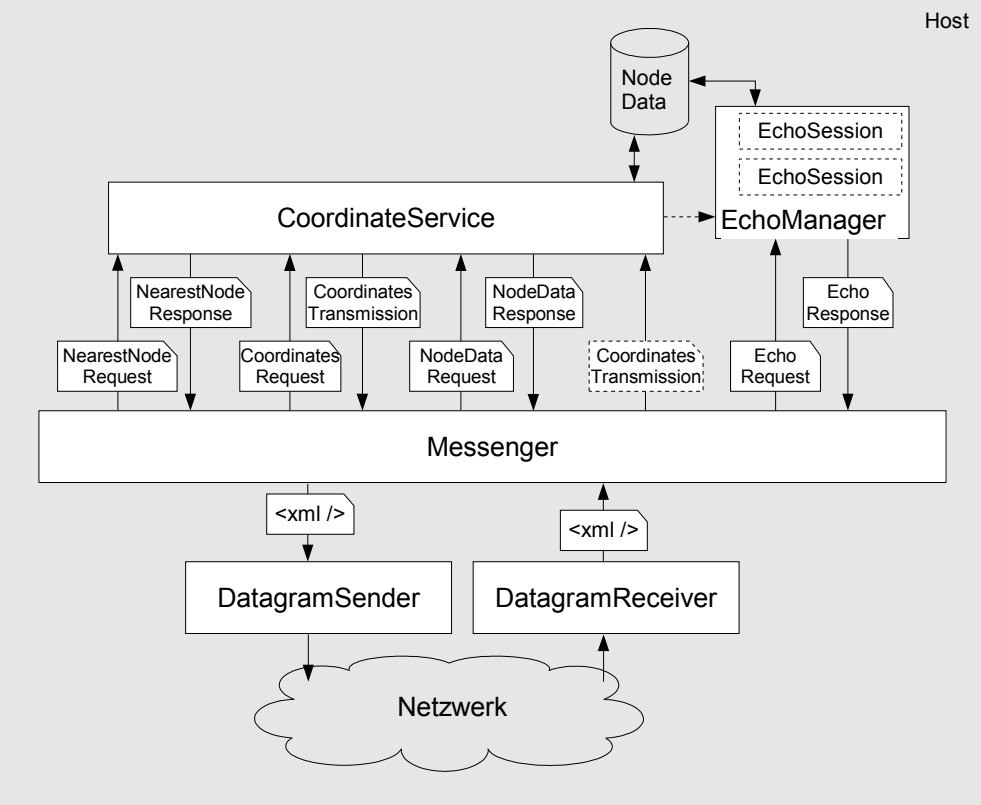

Abbildung 10: Komponenten eines Host (Koordinatenberechnung und -verwaltung)

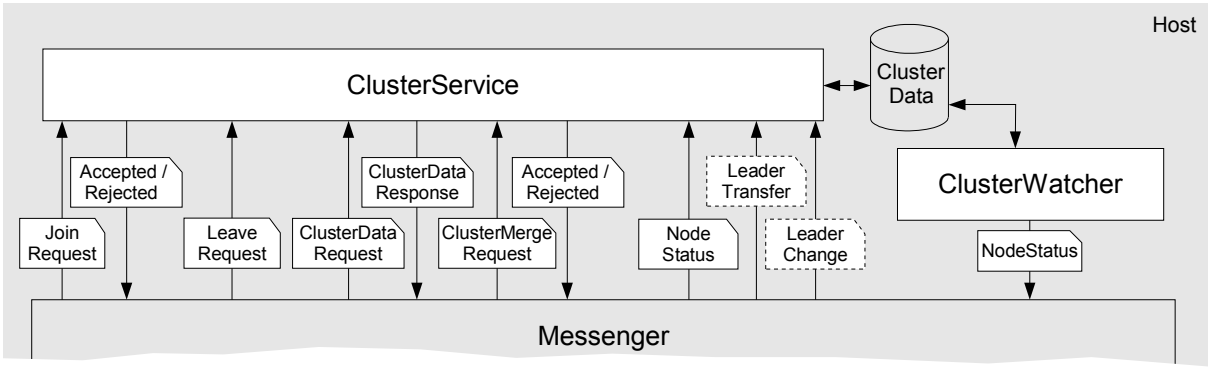

Abbildung 11: Komponenten eines Hosts (Clusterverwaltung)

Die Operationen zur Clusterverwaltung und die PIC-Implementierung sind in den Darstellungen der Systemkomponenten nicht abgebildet, werden aber im Abschnitt 5.3 beschrieben.

Ergänzend zu den einzelnen Komponenten stellen die Abbildung 10, 11 und 12 auch die Typen der ausgetauschten Nachrichten und die eingesetzten Datenspeicher dar. Die Nachrichtentypen werden im Abschnitt 5.3.1 erläutert.

Der Bootstrap-Host besitzt neben dem *Messenger* und der Netzwerkschnittstelle drei weitere Komponenten: dem *LandmarkService,* dem *NodeCacheService* und dem *ClusterStartService*.

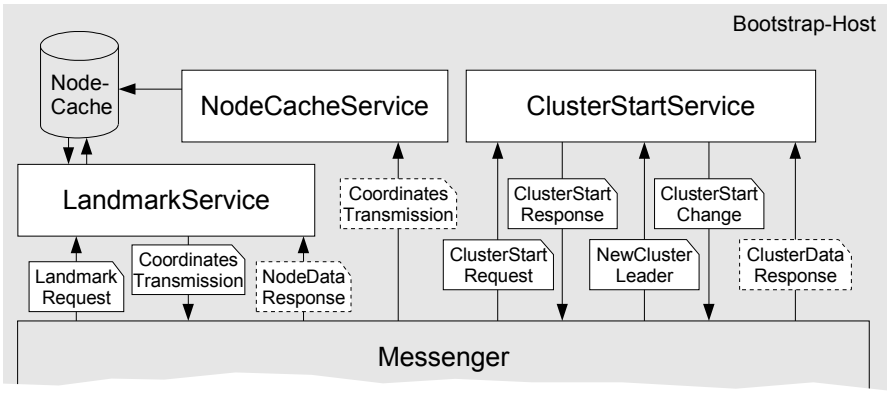

Abbildung 12: Komponenten des Bootstrap-Host

Der *LandmarkService* ist eine Schnittstelle für die Hosts des Overlay-Netzwerks zur Abfrage der Landmarks für die Koordinatenberechnung. Dazu wird ein *NodeCache* eingesetzt, der als Datenspeicher für die Hosts mit bereits berechneten Koordinaten dient. Die Hosts übermitteln nach der Berechnung ihre Koordinaten an den *NodeCacheService*, der diese Koordinaten in den *NodeCache* schreibt. Der *ClusterStartService* speichert und übermittelt den obersten Eintrag in der Clusterhierarchie, um neuen Hosts das Auffinden des nächstgelegenen Clusters zu ermöglichen.

5.3 Gewählte Technologien

Im folgenden Abschnitt werden die zur Implementierung genutzten Technologien und Ansätze vorgestellt und erläutert.

5.3.1 Der Nachrichtenaustausch

Für den Nachrichtenaustausch werden unterschiedliche Container für die Kommunikation zwischen den Java-Objekten eines Hosts und der Übertragung zwischen zwei Hosts im Overlay-Netzwerk eingesetzt. Zur Kommunikation zwischen den Objekten werden *Message*-Objekte verwendet, die für die Übertragung im Netzwerk durch den *Messenger* in XML-Nachrichten (XML-Schema siehe Abbildung 13) umgewandelt werden.[13]

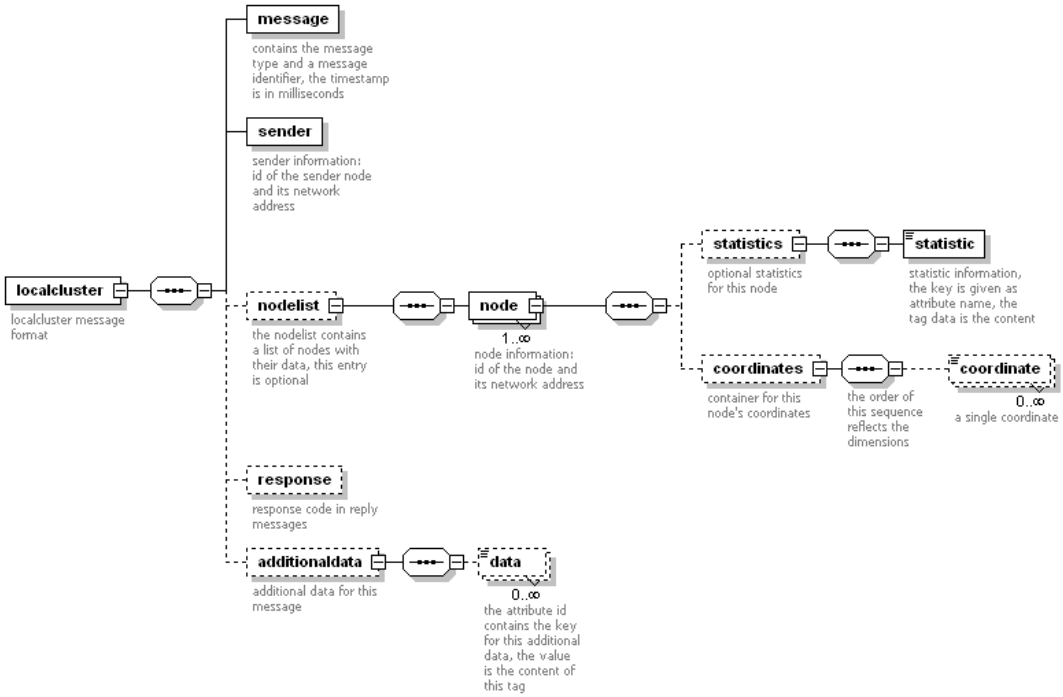

Abbildung 13: XML-Nachrichtenformat für LocalCluster

Der Einsatz von XML als Basis für die Datenübertragung im Overlay-Netzwerk erfolgte aufgrund der dadurch ermöglichten Interoperabilität. Die Hosts können neben dem in Java entwickelten Ansatz auch problemlos Implementierungen in anderen Programmiersprachen einsetzen. Zudem bietet der Einsatz eines XML-Nachrichtenformats die Möglichkeit, die Nachrichten später leicht um zusätzliche Informationen zu erweitern.

Die Konvertierung der XML-Nachrichten in die *Message*-Objekte wird durch den *Messenger* durchgeführt, der dazu ein *MessageConverter*-Objekt aufruft. Dieses Objekt dient als Schnittstelle zwischen der XML-Repräsentation und den *Message*-Objekten und verwendet in der Imple-

13 Nähere Erläuterungen und Verweise auf Literatur zu XML und XML Schema bieten die Websites http://www.w3.org/XML/ und http://www.w3.org/XML/Schema.

mentierung mit dom4j[14] eine externe Bibliothek. Bei geringen verfügbaren Bandbreiten kann die Kommunikation durch den Austausch der *MessageConverter*-Klasse auch ohne XML erfolgen.

Das XML-Schema für die bei der Implementierung verwendeten XML-Nachrichten enthält Elemente, mit denen die gesamte Kommunikation zwischen den Hosts im Overlay-Netzwerk durchgeführt werden kann. Dabei müssen nur die immer benötigten Informationen in den Nachrichten angegeben werden, alle weiteren Elemente sind im XML-Schema als optional gekennzeichnet. Durch die Verwendung eines einzelnen XML-Schema für die gesamte Kommunikation lässt sich die Konvertierung zwischen den über das Netzwerk verschickten Nachrichten und den *Message*-Objekten in der Anwendung leicht realisieren.

Die einzelnen Nodes in den XML-Nachrichten haben die folgenden Funktionen:

Node in der XML-Nachricht als XPath	**Inhalt**
//localcluster/message	Dieses Element enthält den Typ der Nachricht, den Zeitpunkt des Versands und eine eindeutigen ID. Optional kann dieses Element auch die ID einer vorangegangenen Nachricht enthalten, für die die aktuelle Nachricht als Antwort dient.
//localcluster/sender	Der Sender einer Nachricht schickt seine eigenen Daten in diesem Element mit.
//localcluster/nodelist	Die Informationen zu einer oder mehreren Hosts werden in den Nachrichten als Sequenz von Node-Elementen unterhalb des Nodelist-Elements in die Nachricht aufgenommen.
//localcluster/response	Die Antworten auf empfangene Nachrichten können einen Status im Element response zurücksenden.
//localcluster/additionaldata	Zum Versand von beliebigen Text-Informationen können zusätzliche data-Elemente unterhalb des Elements additionaldata zur XML-Nachricht hinzugefügt werden.

14 dom4j steht unter einer BSD-artigen Lizenz und ist auf der Website http://dom4j.org verfügbar.

Zur Klassifizierung von Nachrichten werden verschiedene Typen definiert, die es dem *Messenger* ermöglichen, die Nachrichten an die jeweiligen Handler zu übermitteln. Bei der Implementierung des Verfahrens werden die folgenden Nachrichtentypen eingesetzt:

Kennung des Nachrichtentyps	Nachricht
LandmarkRequest	Die *LandmarkRequest*-Nachrichten dienen zur Ermittlung der Landmarks vom Bootstrap-Host.
JoinRequest	Zum Signalisierung eines Beitrittswunsches zu einem Cluster senden die Hosts eine *JoinRequest*-Nachricht an den Clusterleader.
LeaveRequest	Mit einer *LeaveRequest*-Nachricht melden sich die Hosts aus einem Cluster ab.
CoordinatesRequest	Mit der *CoordinatesRequest*-Nachricht können Hosts die Koordinaten von einem oder mehreren anderen Hosts abfragen. Dazu muss die Nachricht eine Liste von Hosts enthalten, deren Koordinaten ermittelt werden sollen.
CoordinatesTransmission	Eine Nachricht mit diesem Typ enthält die Koordinaten und optional weitere Daten über einen oder mehrere Hosts. Dieser Nachrichtentyp tritt insbesondere bei Antworten auf *CoordinatesRequest*-Nachrichten auf.
NodeDataRequest	Mit diesem Nachrichtentyp können Hosts die Daten eines anderen Hosts von diesem direkt abfragen. Als Antwort sendet der angefragte Host dann eine NodeDataResponse-Nachricht.
NodeDataResponse	Die *NodeDataResponse*-Nachrichten werden von den Hosts verwendet, um die eigenen Daten an andere Hosts zu übertragen. Dieser Nachrichtentyp wird hauptsächlich bei Antworten auf eine *NodeDataRequest*-Nachricht eingesetzt.

Kennung des Nachrichtentyps	Nachricht
NodeStatus	Die periodischen Statusinformationen, die die einzelnen Hosts und der Clusterleader innerhalb des Clusters versenden, werden durch den Nachrichtentyp *NodeStatus* gekennzeichnet.
EchoRequest	Die Implementierung unterstützt das Versenden von Echo-Nachrichten im System zur Ermittlung der Distanz zwischen zwei Hosts. Ein Host sendet dazu eine *EchoRequest*-Nachricht, die von dem entfernten Host mit einer *EchoResponse*-Nachricht beantwortet wird.
EchoResponse	Dieser Nachrichtentyp kennzeichnet die Antwort auf eine *EchoRequest*-Nachricht.
ClusterDataRequest	Mit dem Nachrichtentyp *ClusterDataRequest* können Hosts die Daten zu einem Cluster abfragen. Dazu gehören die Hosts in diesem Cluster sowie zusätzliche Informationen über diesen Cluster.
ClusterDataResponse	Die Antworten auf einen *ClusterDataRequest* werden durch den Nachrichtentyp *ClusterDataResponse* gekennzeichnet.
NearestNodesRequest	Für die genauere Berechnung der eigenen Koordinaten bei der PIC-Implementierung benötigen die Hosts mehrere nahe liegende Hosts. Dazu können sie von den nächstgelegenen Landmarks mit einer *NearestNodesRequest*-Nachricht die Hosts in deren Cluster abfragen.
NearestNodesResponse	Der Nachrichtentyp *NearestNodesResponse* kennzeichnet die Antwort auf einen *NearestNodesRequest*.
ClusterStartRequest	Ein neuer Host ermittelt den obersten Host der Clusterhierarchie vom Bootstrap-Host mit einer Nachricht vom Typ *ClusterStartRequest*.

Kennung des Nachrichtentyps	Nachricht
ClusterStartResponse	Der Bootstrap-Host übermittelt den obersten Host der Hierarchie mit einer *ClusterStartResponse*-Nachricht.
ClusterStartChange	Sobald sich der oberste Host der Clusterhierarchie aus dem Netzwerk entfernt, bestimmt der Bootstap-Host einen neuen Host als Einstiegspunkt der Hierarchie und informiert diesen Host mit einer Nachricht vom Typ *ClusterStartChange*.
LeaderTransfer	Zur Übertragung der Funktion des Clusterleaders an einen anderen Host sendet der Clusterleader eine *LeaderTransfer*-Nachricht an den neuen Clusterleader.
LeaderChange	Bei der Auswahl eines neuen Clusterleaders informiert der bisherige Clusterleader alle Hosts im Cluster mit einer *LeaderChange*-Nachricht über den neuen Clusterleader.
NewClusterLeader	Nach dem Aufteilen eines Clusters bei Überschreiten der Maximalgröße informiert der bisherige Clusterleader den übergeordneten Cluster mit einer *NewClusterLeader*-Nachricht über den neuen Clusterleader für den zweiten Cluster. Dadurch wird der neue Clusterleader in die Hierarchie eingebunden.
AcceptNewClusterLeader	Der Clusterleader des übergeordneten Clusters sendet eine Nachricht vom Typ *AcceptNewClusterLeader* an den Clusterleader des neu gebildeten zweiten Clusters als Antwort auf eine *NewClusterLeader*-Nachricht. Die Antwort enthält zusätzlich alle Hosts im übergeordneten Cluster.
ClusterMergeRequest	Zum Vereinigen mit einem weiteren Cluster beim Unterschreiten der Mindestgröße sendet ein Clusterleader eine Nachricht vom Typ *CluterMergeRequest* an den Clusterleader des anderen Clusters.

Der implementierte *Messenger* übergibt empfangene Nachrichten an die für einen Nachrichtentyp registrierten *MessageHandler*. Eine Nachricht wird durch den *Messenger* verworfen, wenn für ihren Nachrichtentyp kein *MessageHandler* registriert ist.

Die Nachrichtenübertragung zwischen zwei Hosts erfolgt mit UDP-Datagrammen. Diese Entscheidung wurde getroffen, da der Einsatz dieses Protokoll eine schnelle Übertragung der Daten ermöglicht. Zudem stellt die Anwendung keine besonderen Anforderungen an eine gesicherte Übertragung der Daten. Die Übertragung von UDP-Datagrammen ist zustandslos, daher muss die Anwendung eine Möglichkeit bieten, dass ein entfernter Host empfangene Nachrichten beantworten kann. Die Antwort muss dann auf dem lokalen Host wieder an den Sender der ersten Nachricht übermittelt werden.

Bei LocalCluster wird dieser Ablauf erreicht, indem der *Messenger* das *Message*-Objekt der Anfrage in einem Zwischenspeicher ablegt. Die Antwort enthält neben den Daten zusätzlich die Kennung der Anfrage, mit der der *Messenger* die ursprüngliche Nachricht und das zugehörige anfragende Objekt aus dem Zwischenspeicher ermitteln kann. Anschließend übergibt der *Messenger* das aus der Antwort generierte *Message*-Objekt an das anfragende Objekt.

Die Netzwerkübertragung erfolgt durch Instanzen der Empfängerklasse *DatagramReceiver* und der Senderklasse *DatagramSender*, deren Funktionalitäten dem *Messenger* über vordefinierte Schnittstellen zur Verfügung gestellt werden. Durch diese festen Schnittstellen können beide Klassen problemlos gegen eine andere Implementierung ausgetauscht werden, um die Anwendung auch in anderen Umgebungen als dem Internet einsetzen zu können.

Die *DatagramReceiver*-Klasse stellt ihre Funktionen über Handler-Schnittstellen für registrierte *PacketHandler* zur Verfügung. Alle empfangenen Nachrichten werden dann als Zeichenkette an die Handler übergeben. In der Implementierung wird der *Messenger* als Handler beim *DatagramReceiver*-Objekt registriert.

Die Senderklasse *DatagramSender* besitzt eine Methode, mit der beliebige Zeichenketten über das Netzwerk gesendet werden können. Bei LocalCluster wird diese Methode nur durch den *Messenger* aufgerufen, nachdem die *Message*-Objekte mit dem *MessageConverter* in eine XML-Zeichenkette konvertiert wurden.

5.3.2 Der Messenger

Der *Messenger* ist die Zentrale für den Nachrichtenaustausch innerhalb der Implementierung. Er ermöglicht allen Objekten das Senden von Nachrichten an entfernte Hosts und verteilt empfangene Nachrichten an die verantwortlichen Objekte. Dabei werden die Nachrichten zwischen dem XML-Nachrichtenformat und den *Message*-Objekten mittels eines *MessageConverters* umgewandelt.

Die Anbindung der verarbeitenden Objekte erfolgt über Handler-Schnittstellen, bei denen sich die Objekte für die jeweiligen Nachrichtentypen registrieren können. Beim Empfang von Nachrichten übergibt der *Messenger* diese je nach Typ der Nachricht an die registrierten Handler.

5.3.3 Die Distanzbestimmung

Die Distanzbestimmung zwischen zwei Hosts erfolgt mit einem Echo-Verfahren, bei dem die Nachrichten ebenfalls in dem definierten XML-Nachrichtenformat übertragen werden. Mit dieser Herangehensweise ist die Distanzermittlung bei funktionierender Netzwerkverbindung für die Clusterverwaltung immer sichergestellt. Werden dagegen Verfahren wie *Ping* eingesetzt, die eine Distanzmessung unabhängig von der Anwendung durchführen, ist das gleichzeitige Funktionieren der Distanzmessung und der Implementierung der Clusterverwaltung nicht sichergestellt und muss daher zusätzlich abgesichert werden. Gegen den Einsatz speziell von *Ping* spricht auch die fehlende native Unterstützung für ICMP-Nachrichten in Java. Bei Verwendung von *Ping* würden für die unterschiedlichen Plattformen jeweils eigene Implementierungen benötigt, was aber eine Abkehr von der gewünschten Plattformunabhängigkeit bedeuten würde.

Als Nachteil des gewählten Ansatzes erweist sich die Verfälschung des gemessenen Wertes durch die Verarbeitung und Konvertierung der Nachrichten beim Sender und Empfänger. Zur Minimierung dieser Verfälschung übermittelt der *Messenger* auf Empfängerseite in der Antwortnachricht die Empfangs- und Sendezeit der Echo-Nachrichten als zusätzliche Daten. Der Sender kann nun die auf dem entfernten System benötigte Zeit berechnen und das Ergebnis korrigieren.

Die Distanzberechnung wird in der Implementierung durch den *EchoManager* durchgeführt, der dazu als Eingabe eine Liste von Hosts erhält. Zur Begrenzung der maximalen Laufzeit besitzt der *EchoManager* ein vordefiniertes Timeout, bis zu dem die einzelnen Messungen beendet sein müssen. Die Messung der Distanz zu einem entfernten Host wird durch ein Objekt der Klasse *EchoSession* ausgeführt. Die *EchoSession* sendet eine vordefinierte Anzahl von parallelen und gestaffelten Echo-Nachrichten an den entfernten Host und berechnet aus den einzelnen Resultaten einen Mittelwert. Dieses Vorgehen ist notwendig, um eventuelle Schwankungen der Netzwerklast auszugleichen und auftretende Extremwerte bei den Übertragungen zu erkennen.

Die *EchoSession* wartet eine vorgegebene Zeitspanne auf die eintreffenden Antworten. Dieser Wartezustand wird automatisch beendet, sobald die Antworten auf alle versendeten Echo-Nachrichten eingetroffen sind. Mit den empfangenen Antworten berechnet die *EchoSession* die durchschnittliche Distanz und gibt diese über eine Schnittstelle an den *EchoManager* zurück.

5.3.4 Die PIC-Implementierung

Die Koordinatenberechnung für die Hosts erfolgt in der Implementierung auf Basis des im Abschnitt 3.2.4 vorgestellten PIC-Ansatzes. Dabei berechnen neue Hosts ihre Koordinaten anhand der Distanzen zu mehreren Landmarks und deren bereits bestimmten Koordinaten.

Die im Rahmen der Arbeit durchgeführte Implementierung von PIC verwendet die in [COS03] vorgestellten Methoden. Als Grundlage zur Koordinatenberechnung wird bei PIC der GNP-Algorithmus eingesetzt, für dessen Implementierung auf die Verfahren aus [ZHA01] und die Quellcodefragmente von der Website des Projektes[15] zurückgegriffen wurden.

15 http://www.cs.rice.edu/~eugeneng/research/gnp/

Für die Funktionsweise des Ansatzes ist eine vorgegebene Mindestzahl von bereits im Netzwerk vorhandenen Hosts nötig, wobei diese Voraussetzung beim Aufbau des Overlay-Netzwerks aber nicht erfüllt ist. Die Implementierung löst dieses Problem durch die Aufteilung der PIC-Implementierung in zwei unterschiedliche Phasen, die Initialisierungsphase und die Betriebsphase. Der Bootstrap-Host übernimmt die Aufgabe, die Phasen zu verwalten und die aktuelle Phase an die Hosts zu übermitteln.

Die Initialisierungsphase wird angewendet, solange nicht genügend Landmarks verfügbar sind, um genaue Koordinaten berechnen zu können. Sobald ausreichend viele Hosts im Overlay-Netzwerk vorhanden sind, erfolgt der Wechsel in die Betriebsphase. Dazu berechnet der Bootstrap-Host die genauen Koordinaten für alle bereits vorhandenen Hosts im Overlay-Netzwerk und überträgt diese an die einzelnen Hosts.

Die Unterscheidung der jeweiligen Phase durch die Hosts erfolgt im Zustand *GetLandmarks*, bei dem die Landmarks vom Bootstrap-Host abgefragt werden. Die Antwortnachricht des Bootstrap-Host enthält in den zusätzlichen Daten die Information, ob sich das Overlay-Netzwerk in der Initialisierungs- oder in der Betriebsphase befindet.

Die PIC-Implementierung während der Initialisierungsphase

Die PIC-Implementierung verwendet in der Initialisierungsphase drei verschiedene Zustände:

1. *GetLandmarks*
   In diesem Zustand werden die Landmarks ermittelt. Der Host sendet eine Nachricht vom Typ *LandmarkRequest* an den Bootstrap-Host, die dieser mit einer *Coordinates-Transmission*-Nachricht beantwortet. Diese Antwortnachricht enthält neben einer Liste von Hosts die Information, ob sich der Bootstrap-Host noch in der Initialisierungsphase befindet. Dazu sendet der Bootstrap-Host als zusätzliche Daten *init=1* in der Antwortnachricht mit.
   Im Initialisierungsprozess enthält die Liste von Hosts aus der Antwortnachricht des Bootstrap-Hosts alle bisher im Netzwerk vorhandenen Hosts.
2. *InitMeasureLandmarkDistance*
   Nachdem die Landmarks ermittelt wurden, wechselt die PIC-Implementierung während der Initialisierungsphase in den Zustand *InitMeasureLandmarkDistance*. In diesem Zustand misst der neue Host mit dem *EchoManager* die Distanzen zu den bereits im Netzwerk vorhandenen Hosts. Anschließend wechselt die Implementierung in den Zustand *InitFinishedState*.
3. *InitFinishedState*
   Zum Abschluss der PIC-Implementierung während der Initialisierungsphase werden die gemessenen Distanzen mit einer *NodeDataResponse*-Nachricht an den Bootstrap-Host übermittelt.

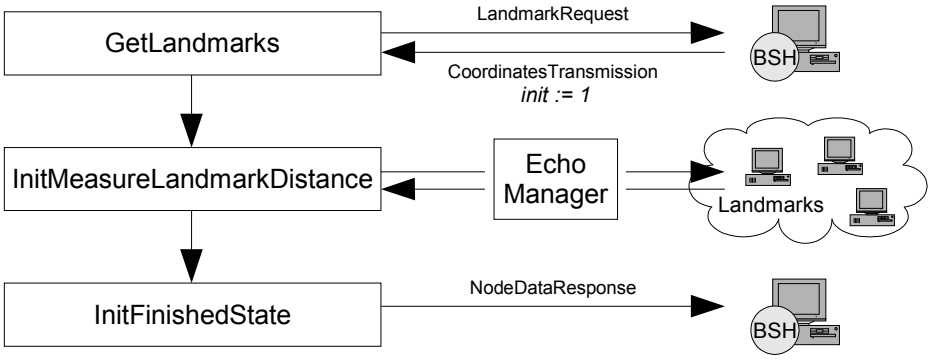

Abbildung 14: Die PIC-Implementierung in der Initialisierungsphase

Während der Initialisierungsphase berechnen die Hosts keine eigenen Koordinaten. Die Berechnung erfolgt durch den Bootstrap-Host, sobald ausreichend Landmarks verfügbar sind. Die Anzahl der für die initiale Koordinatenberechnung benötigten Landmarks liegt höher als die Anzahl der später für die Koordinatenbestimmung von einzelnen Hosts verwendeten Landmarks. Diese Vorgehensweise wird gewählt, damit bei den späteren Berechnungen genügend Landmarks verfügbar sind, auch wenn zwischenzeitlich einige Hosts das Netzwerk verlassen.

Nach der initialen Berechnung überträgt der Bootstrap-Host die Koordinaten an die Landmarks in einer *CoordinatesTransmission*-Nachricht.

Die gewählte Implementierung weicht während der Initialisierungsphase von dem in [COS03] vorgestellten Ansatz ab. Dort berechnen neue Hosts die Koordinaten aller bereits im Overlay-Netzwerk vorhandenen Hosts erneut, bis genügend Landmarks zur Verfügung stehen. Diese Lösung stellt aber eine besondere Fehlerquelle dar, da die Hosts die Koordinaten anderer Hosts ändern können und nicht sichergestellt ist, dass die Berechnungen der Koordinaten korrekt durchgeführt werden.

Die PIC-Implementierung in der Betriebsphase

Im Gegensatz zur Initialisierungsphase berechnen neue Hosts in der Betriebsphase ihre Koordinaten selbst. Dabei werden die folgenden Zustände verwendet:

1. *GetLandmarks*
   Ein neuer Host ermittelt in diesem Zustand die Landmarks vom Bootstrap-Host. In dieser Phase überträgt der Bootstrap-Host keine zusätzlichen Informationen zur Kennzeichnung der aktuellen Phase.
2. *MeasureLandmarkDistance*
   Nachdem ein Host die Landmarks vom Bootstrap-Host ermittelt hat, misst er die Distanzen zu allen Landmarks. Dazu wird der *EchoManager* eingesetzt.

3. *RawCoordinateCalculation*
   Mit den Distanzen und den im ersten Zustand abgefragten Koordinaten der Landmarks erfolgt in diesem Zustand die näherungsweise Berechnung der Koordinaten. Diese Berechnung entspricht dem Ansatz „Zufällige Auswahl von Landmarks“ aus dem PIC-Ansatz (siehe Abschnitt 3.2.4).

4. *GetNearestNodes*
   Für die Umsetzung des hybriden Ansatzes von PIC werden für die genaue Berechnung der Koordinaten zusätzlich weitere Landmarks benötigt. Diese Landmarks müssen nahe beim neuen Host liegen. Im Zustand *GetNearestNodes* werden nun von den nächstgelegenen Landmarks die ihnen jeweils nächstgelegenen Hosts abgefragt. Der neue Host bestimmt mit den eigenen näherungsweise berechneten Koordinaten und den Koordinaten der ermittelten Hosts die Distanzen zu diesen Hosts und wählt die gefundenen nächstgelegenen Hosts aus.

5. *MeasureNearestNodesDistance*
   Zur genauen Berechnung der Koordinaten werden im Zustand *MeasureNearestNodesDistance* die Distanzen zu den ausgewählten nächstgelegenen Hosts gemessen.

6. *FineCoordinateCalculation*
   In diesem Zustand berechnet der neue Host mit den ursprünglichen Landmarks und den nächstgelegenen Host die eigenen Koordinaten erneut. Diese Berechnung entspricht dabei dem hybriden Ansatz von PIC.

7. *FinishedState*
   Der Host überträgt im Zustand *FinishedState* die berechneten Koordinaten an den Bootstrap-Host und beendet anschließend den PIC-Algorithmus. Die Übertragung der Koordinaten an den Bootstrap-Host ist notwendig, damit dieser Host als Landmark verwendet werden kann. Der Bootstrap-Host stellt in dieser Implementierung den Zugangspunkt zum Overlay-Netzwerk dar und übermittelt neuen Hosts die für die Koordinatenberechnung benötigten Landmarks.

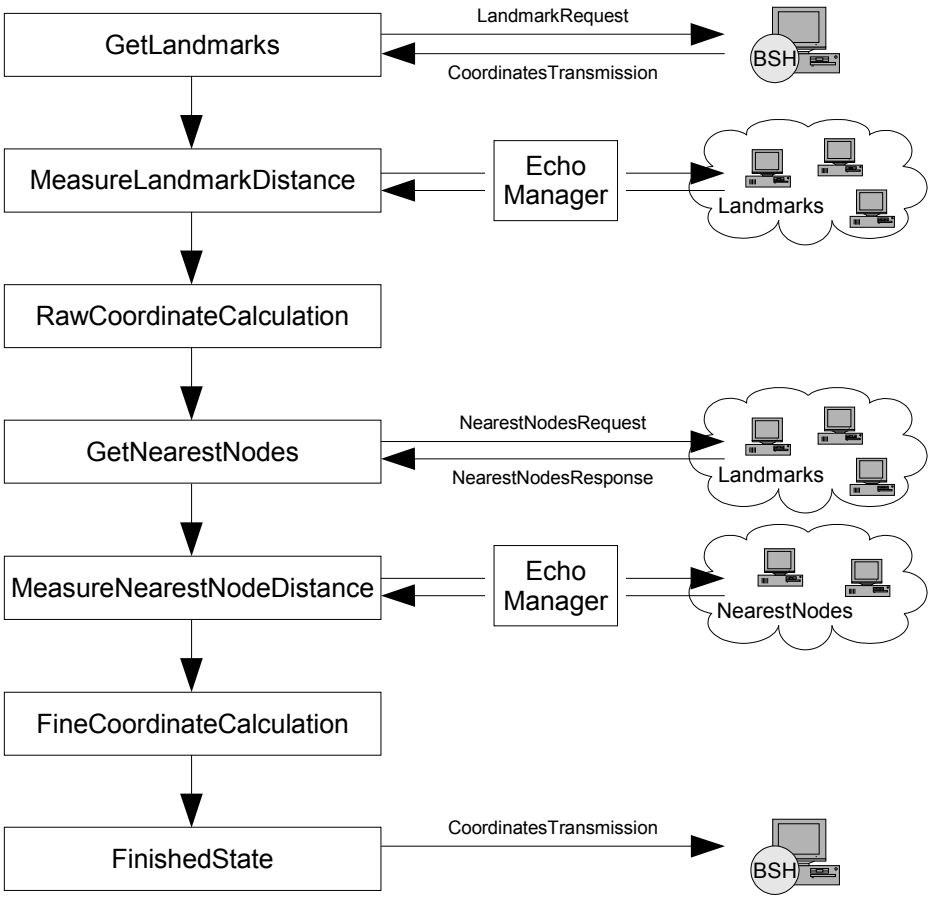

Abbildung 15: Die PIC-Implementierung in der Betriebsphase

5.3.5 Das Clustering

Die Bildung und Verwaltung der lokalen Cluster wird für die Implementierung in die folgenden vier Aufgaben unterteilt:

- Beitreten von neuen Hosts zu einem Cluster
- Verlassen der Clusterhierarchie
- Aufteilen von Clustern
- Vereinigen von Clustern.

Diese Teilaufgaben wurden entsprechend den Entwürfen aus dem Abschnitt 4.3 sowie den in [BAN01] vorgestellten Ansätzen implementiert.

Beitreten zu einem Cluster

Das Beitreten zu einem Cluster wird durch die Operation *ClusterJoinOperation* durchgeführt. Diese ermittelt den obersten Host der Clusterhierarchie, sucht dann sukzessive den nächstgelegenen Cluster auf der untersten Ebene der Hierarchie und versucht anschließend, diesem Cluster beizutreten. Bei einer Ablehnung der Beitrittsanfrage durch den Clusterleader wählt der

Host aus den übrigen Clustern wiederum den nächstgelegenen Cluster aus und versucht einen Beitritt in diesen Cluster. Sollte der neue Host in keinen der Cluster auf der untersten Ebene der Hierarchie aufgenommen werden, startet er nach einer vorgegebenen Wartezeit die *ClusterJoinOperation* erneut.

Die Implementierung der *ClusterJoinOperation* verwendet die folgenden Zustände:

1. *GetClusterStart*
   Im Zustand *GetClusterStart* ermittelt der Host den obersten Host der Clusterhierarchie. Dazu fragt er den Bootstrap-Host an, der die Adresse des obersten Hosts der Clusterhierarchie verwaltet, und erhält als Antwort die Daten des obersten Hosts der Hierarchie. Dieser Host wird dann als momentan nächstgelegener Host zwischengespeichert.
   Existiert bisher noch keine Clusterhierarchie, so übermittelt der feste Host diese Information als zusätzliche Daten in der Antwortnachricht. In diesem Fall wechselt der neue Host in den Zustand *CreateNewClusterTree* und erzeugt eine neue Clusterhierarchie.

2. *TraverseTree*
   Dieser Zustand dient zur Ermittlung des lokalen Clusters. Dabei fragt der neue Host die Liste der Hosts im Cluster des aktuell nächstgelegenen Hosts auf der momentan betrachteten Hierarchieebene ab. Im ersten Aufruf dieses Zustands wird zusätzlich die Gesamtzahl der Hierarchieebenen durch den obersten Host der Hierarchie übermittelt.
   Aus der ermittelten Liste von Hosts wählt der neue Host den nächstgelegenen Host aus. Für diese Auswahl misst er während der Initialisierungsphase der PIC-Implementierung die Distanzen mit Echo-Nachrichten. In der Betriebsphase der PIC-Implementierung ist die Messung nicht mehr notwendig, sondern wird durch eine Distanzberechnung mit den eigenen Koordinaten und den Koordinaten der Hosts ersetzt (zu den Phasen der PIC-Implementierung siehe Abschnitt 5.3.4).
   Mit dem gefundenen nächstgelegenen Host wird der Zustand *TraverseTree* erneut ausgeführt, wobei in diesem Schritt die momentan betrachtete Hierarchieebene um 1 verringert wird.
   Diese Aktion wird solange durchgeführt, bis der nächstgelegene Host auf der Hierarchieebene 1 ermittelt wurde. In diesem Fall versucht der neue Host dem Cluster dieses nächstgelegenen Hosts beizutreten.

3. *JoinCluster*
   Der neue Host sendet in diesem Zustand eine Beitrittsanfrage an den Clusterleader des nächstgelegenen Clusters. Wird die Anfrage vom Clusterleader akzeptiert, wechselt der neue Host in den Zustand *JoinFinished*.
   Lehnt der Clusterleader die Beitrittsanfrage dagegen ab, ermittelt der neue Host aus den Nachbarn des Clusterleaders den nächstgelegenen Clusterleader und die Aktionen des Zustandes *JoinCluster* werden erneut ausgeführt. Hat der Clusterleader keine Nachbarn oder haben alle Nachbarn bereits eine Beitrittsanfrage des neuen Hosts abgelehnt, so wird aus sämtlichen im Zustand *TraverseTree* gefundenen Hosts der nächstgelegene Host ausge-

wählt. Dabei werden Hosts, die eine Beitrittsanfrage des neuen Hosts abgelehnt haben, nicht mehr verwendet.
Anschließend wechselt der neue Host mit dem gefundenen nächstgelegenen Host wieder in den Zustand *TraverseTree*, filtert in den folgenden Aktionen aber alle Hosts heraus, die eine Beitrittsanfrage abgelehnt haben.

4. *JoinFinishedState*
   Nachdem der Beitritt zu einem Cluster im Zustand *JoinCluster* erfolgt ist, ermittelt der neue Host die Liste der Hosts im Cluster und weitere benötigte Informationen vom Clusterleader und speichert diese Daten.

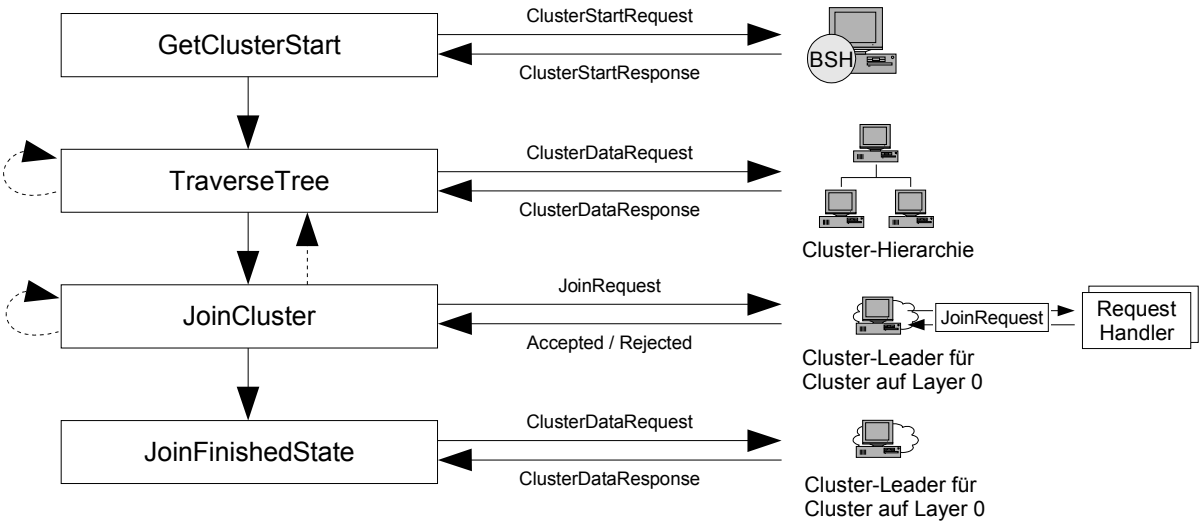

Abbildung 16: Die ClusterJoinOperation beim Beitritt eines neuen Hosts

Das Verlassen der Clusterhierarchie

Zum Verlassen der Clusterhierarchie ruft ein Host die Operation *ClusterLeaveOperation* auf. Diese implementiert die in Abschnitt 4.3.3 dargestellten Schritte ausgehend von der bisherigen Funktion des Hosts.

Bei Clusterleadern wird während der *ClusterLeaveOperation* ein neuer Clusterleader für den jeweiligen Cluster bestimmt, der über seine neue Funktion anschließend informiert wird. Zudem sendet der alte Host allen einfachen Hosts in diesem Cluster eine Nachricht, die über den geänderten Clusterleader informiert.

Bei einer *ClusterLeaveOperation* treten folgenden Zustände auf:

1. *LeaveCluster*
   Im Zustand *LeaveCluster* wird der Host aus allen Clustern entfernt, denen er bisher angehörte. Bei Hosts, die bisher in keinem Cluster als Clusterleader fungierten, wird dazu lediglich eine *LeaveRequest*-Nachricht an den Clusterleader des Clusters auf der Hierarchieebene 0 gesendet. Damit ist die *ClusterLeaveOperation* für diesen Host beendet.

Hosts, die bisher als Clusterleader für einen oder mehrere Cluster dienten, müssen dagegen für diese Cluster jeweils neue Clusterleader bestimmen. Dazu wählen sie aus den Hosts in den einzelnen Clustern jeweils den ältesten Host aus und übertragen diesem die Funktion des Clusterleaders. Zudem informieren sie die Hosts in diesem Cluster und gegebenenfalls die übergeordneten Cluster über den neuen Clusterleader.

2. *LeaveFinished*
   Im Zustand *LeaveFinished* überprüft der Host, ob er der oberste Host in der Clusterhierarchie ist. In diesem Fall informiert er den Bootstrap-Host für die Clusterhierarchie über den neuen obersten Host der Hierarchie, den er als Clusterleader des obersten Clusters der Hierarchie im Zustand *LeaveCluster* ausgewählt hat.

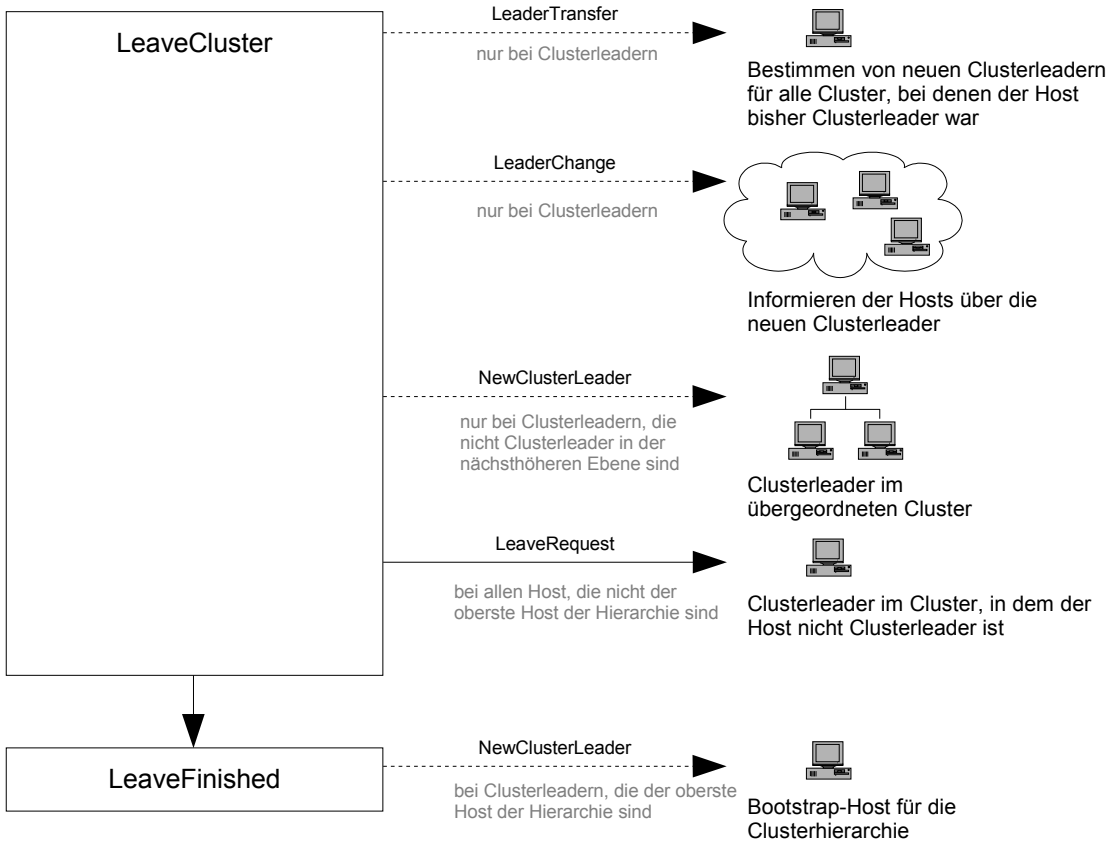

Abbildung 17: Die ClusterLeaveOperation zum Verlassen der Clusterhierarchie

Das Aufteilen von Clustern

Durch den Beitritt neuer Hosts können Cluster die maximale Größe für einen Cluster überschreiten (siehe Abschnitt 4.3.4). In diesem Fall führt der Clusterleader eine *ClusterSplit-Operation* durch, die den Cluster in zwei kleinere Cluster aufteilt. Die Hosts aus dem bisherigen Cluster werden gleichmäßig den beiden Clustern zugewiesen, wobei die Zuordnung anhand der Distanzen der Hosts untereinander erfolgt.

Die *ClusterSplitOperation* verwendet die folgenden Zustände:

1. *SplitCluster*
   In diesem Zustand werden aus dem bisherigen Cluster die beiden kleineren Teilcluster gebildet und die Clusterleader ausgewählt. Der bisherige Clusterleader bleibt aber weiterhin der Clusterleader für einen der neuen Cluster.
   Zur Berechnung der Teilcluster wird ein Approximationsalgorithmus verwendet.
2. *FinishSplit*
   Nachdem der Clusterleader die beiden Cluster gebildet hat, informiert er den neuen Clusterleader und die Hosts in dessen neuem Cluster über die Änderung. Da die Clusterleader in einer Hierarchieebene auch Teil des übergeordneten Clusters auf der nächsthöheren Ebene sind, informiert er dessen Clusterleader über den neuen Clusterleader im zweiten Teilcluster.

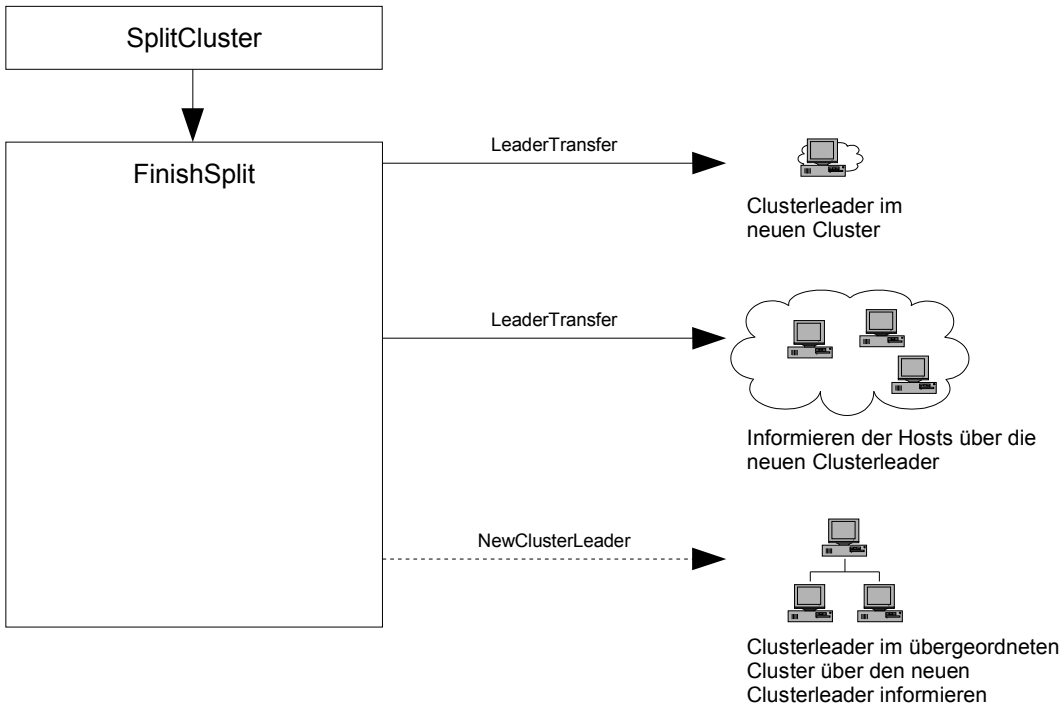

Abbildung 18: Die ClusterSplitOperation zum Teilen von Clustern

Das Vereinen von Clustern

Sinkt die Größe eines Clusters durch das Verlassen von Hosts unter die Mindestgröße, so führt der Clusterleader eine *ClusterMergeOperation* durch. Diese Operation wird aus Performancegründen nur ausgeführt, nachdem die Koordinaten für die Hosts bereits berechnet wurden (siehe dazu auch Abschnitt 5.3.4).

Bei einer *ClusterMergeOperation* werden die Hosts eines Clusters dem nächstgelegenen Nachbarcluster zugeordnet, der einer Vereinigung zustimmt. Lehnen alle Nachbarcluster den *ClusterMergeRequest* ab, wird die *ClusterMergeOperation* abgebrochen.

Die *ClusterMergeOperation* besteht aus den folgenden Zuständen:

1. *FindClusterForMerging*
   Der Clusterleader ermittelt im Zustand *FindClusterForMerging* alle Nachbarcluster und sortiert sie nach der Distanz zum aktuellen Cluster.

2. *MergeCluster*
   Der Zustand *MergeCluster* dient zum Durchführen der eigentlichen Vereinigungsoperation, wobei die sortierte Liste der Nachbarcluster verwendet wird. Der Clusterleader wählt den nächstgelegenen Cluster aus der Liste und sendet eine Nachricht vom Typ *ClusterMerge-Request* an dessen Clusterleader, worauf dieser entweder eine *Accepted-* oder *Rejected-*Nachricht zurücksendet. Im ersten Fall informiert der bisherige Clusterleader alle Hosts in seinem Cluster über den neuen Clusterleader und führt anschließend eine *ClusterLeave-Operation* durch, um gegebenenfalls aus allen übergeordneten Clustern entfernt zu werden.
   Wird die *ClusterMergeRequest*-Anfrage dagegen abgelehnt, so entfernt der Clusterleader den Cluster des angefragten Clusterleaders aus der Liste der nächstgelegenen Cluster und ruft den Zustand *MergeCluster* mit den verbleibenden Einträgen aus dieser Liste erneut auf. Lehnt auch der Clusterleader des letzten verbliebenen Clusters aus der Liste der Nachbarcluster die *ClusterMergeRequest*-Anfrage ab, wird die *ClusterMergeOperation* beendet.

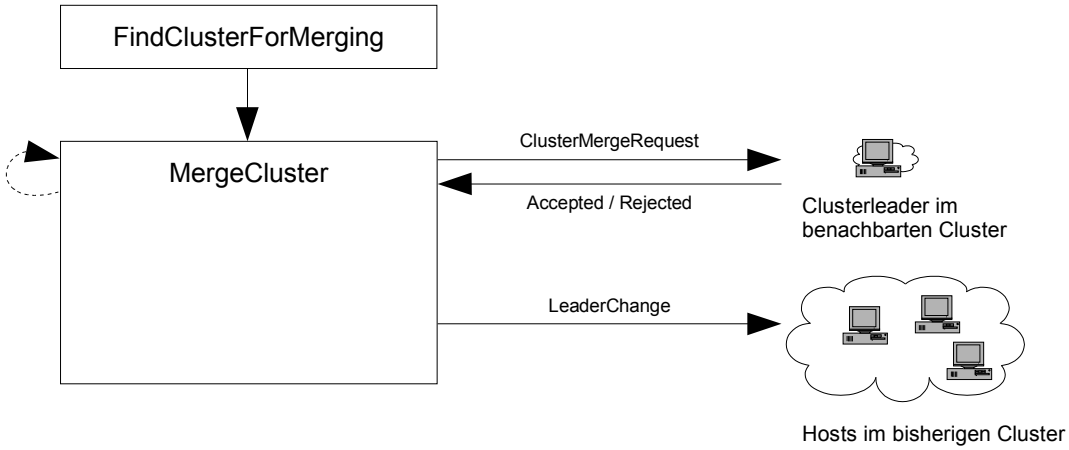

Abbildung 19: Die ClusterMergeOperation zum Vereinen von Clustern

5.4 Kenngrößen und Standardwerte

Bei der Implementierung wurden für die Koordinatenbestimmung und die Clusterbildung die in den jeweiligen Arbeiten ermittelten optimalen Werte verwendet.

Die Koordinatenbestimmung erfolgt dabei in einem 8-dimensionalen Koordinatensystem mit 16 Landmarks, wobei 12 Landmarks zufällig ausgewählt und die restlichen 4 anschließend über den Ansatz „Nächstgelegene Hosts verwenden“ ermittelt werden. Die Bestimmung der Distanzen zwischen den Hosts erfolgt in mehreren unabhängigen Messungen, um eventuelle Schwankungen

in der Netzwerkauslastung ausgleichen zu können. Für die Implementierung wird die Anzahl dieser Messungen auf 10 Einzelmessungen festgelegt.

Die initialen Koordinaten werden durch den Bootstrap-Host erst berechnet, sobald ausreichend Hosts im Netzwerk verfügbar sind. Diese Anzahl wird in der Implementierung mit 25 Hosts etwas größer als die Anzahl der zur Koordinatenberechnung benötigten Landmarks gewählt, damit neue Hosts auch beim Wegfall einzelner Landmarks die Koordinaten bestimmen können.

Die Clustergröße bei der NICE-Implementierung wird durch die Konstante k beeinflußt, mit der die obere und untere Grenze für die Anzahl von Hosts in einem Cluster festgelegt wird. In der Arbeit [KOM02] wird als optimaler Wert $k=12$ ermittelt, weshalb dieser Wert auch bei der Implementierung von LocalCluster eingesetzt wird.

6 Bewertung

Im folgenden Kapitel werden der im Rahmen dieser Arbeit erstellte Entwurf sowie die Implementierung hinsichtlich des Aufwandes für die Kommunikation und des Speicherbedarfs untersucht.

Daran schließt sich eine Betrachtung von möglichen Problemen und Risiken an, die beim Einsatz der Implementierung auftreten können.

6.1 Aufwandsanalyse

Die Koordinatenberechnung unterteilt sich in die Initialisierungsphase und die Betriebsphase (siehe Abschnitt 5.3.4).

Während der Initialisierungsphase muss ein neuer Host n die Koordinaten zu den bisherigen Hosts H im Netzwerk berechnen und wird dann selbst zu diesen Hosts hinzugefügt $H \cup \{n\}$. Die obere Grenze h_{max} für die Anzahl der Hosts in H ist etwas größer gewählt als die für die Koordinatenberechnung benötigten Landmarks $|L|$ und wurde für die Implementierung auf $h_{max}=25$ festgelegt. Somit benötigen die Hosts während der Initialisierungsphase der Koordinatenberechnung höchstens h_{max} Distanzmessungen.

In der Betriebsphase ist die Anzahl der Distanzmessungen für die Koordinatenberechnung immer konstant. Ein neuer Host misst im ersten Schritt immer die Distanzen zu genau $|L|$ anderen Hosts, zur Optimierung der Koordinaten werden anschließend die Distanzen zu weiteren m Hosts gemessen. Die Werte von m und $|L|$ sind konstant und werden für die Implementierung festgelegt. Während der Betriebsphase werden für die Koordinatenberechnung somit durch jeden neuen Host genau $|L|+m$ Distanzmessungen durchgeführt, wobei $|L|+m<h_{max}$.

Eine obere Abschätzung für die bei der Koordinatenberechnung durchgeführten Distanzmessungen ist daher der für die Implementierung vorgegebene konstante Wert von h_{max}. Dieser Wert gibt die Anzahl der Distanzmessungen zu entfernten Hosts an, wobei in der Implementierung pro Distanzmessung mehrere einzelne Messungen durchgeführt werden, um die Genauigkeit der Ergebnisse zu erhöhen (siehe Abschnitt 5.3.3).

Bei der Koordinatenberechnung werden - unabhängig von der aktuellen Phase - neben der Distanzmessung immer 3 Nachrichten im System übertragen. Der neue Host sendet zuerst eine Anfrage an den Bootstrap-Host, um die Landmarks zu ermitteln. Dieser sendet eine Liste von Hosts als Antwort an den neuen Host. Nachdem die Koordinatenberechnung durch den Host abgeschlossen wurde, sendet er eine weitere Nachricht an den Bootstrap-Host. In der Initialisierungsphase enthält diese Nachricht die gemessenen Distanzen zu allen Landmarks, in der Betriebsphase hingegen die berechneten Koordinaten des neuen Hosts.

Beim Wechsel von der Initialisierungsphase in die Betriebsphase werden durch den Bootstrap-Host die initialen Koordinaten an die bisher im Netzwerk vorhandenen Hosts übermittelt. Die Anzahl der dafür notwendigen Nachrichten entspricht der Anzahl der für die initiale Berechnung benötigten Landmarks h_{max}.

Der Zeitaufwand für die Berechnung der initialen Koordinaten durch den Bootstrap-Host und der Koordinatenberechnung durch die einzelnen Hosts entspricht den in [ZHA01] ermittelten Kosten. Für einen geometrischen Raum mit Dimension d betragen diese Kosten $O(|L^2| \cdot d)$ bei der initialen Koordinatenberechnung für die Landmarks und $O(|L| \cdot d)$ bei der Koordinatenberechnung für einzelne Hosts.

Durch das Clustering wird eine Clusterhierarchie gebildet, bei der die Struktur einem k-nären Baum mit n Knoten und einer Höhe von $h \geq \log_k(n+1)$ entspricht. Die Höhe des Baumes ist somit durch $O(\log n)$ beschränkt.

Beim Beitritt zu einem Cluster sendet ein neuer Host jeweils an einen Host pro Hierarchieebene i, mit $i \geq 1$, eine Nachricht. Mit dieser Nachricht werden die Koordinaten aller Hosts in dem Cluster auf der nächsttieferen Ebene ermittelt, für den der angefragte Host als Clusterleader fungiert. Die Summe der benötigten Anfragen für die gesamte Clusterhierarchie ist daher beschränkt auf $O(\log n)$ Abfragen von Koordinaten.

Die Verwendung von Koordinaten beim Finden des nächstgelegenen Clusters ist nur in der Betriebsphase der Koordinatenberechnung möglich. Während der Initialisierungsphase der Koordinatenberechnung müssen neue Hosts dagegen die Distanzen zu allen Hosts in der Hierarchie messen, wobei höchstens max_H Distanzmessungen benötigt werden.

Nachdem ein Host einem Cluster beigetreten ist, sendet er periodisch Statusinformationen an die Hosts in seinem Cluster. Clusterleader müssen diese Nachrichten an die Hosts in allen Clustern senden, zu denen sie gehören. Hosts, die in keinem Cluster als Clusterleader fungieren, senden dabei Nachrichten an maximal $3k-2$ andere Hosts, da die maximale Größe eines Clusters auf $3k-1$ Hosts beschränkt ist. Beim Überschreiten der oberen Grenze $3k-1$ für die Größe eines Clusters wird dieser in zwei kleinere Teilcluster aufgeteilt (siehe Abschnitt 4.3.5).

Die Clusterleader senden in jedem Cluster, dem sie angehören, maximal $3k-2$ Statusnachrichten. Der Clusterleader auf der höchsten Ebene der Hierarchie muss diese Statusnachrichten daher an höchstens $3k-2$ Hosts auf jeder Ebene der Hierarchie senden. Die Anzahl der Statusinformationen für einen Clusterleader ist somit auf $O(k \cdot \log n)$ beschränkt, bei einfachen Hosts auf $O(k)$.

Die Hosts in der Clusterhierarchie speichern zudem die Informationen zu allen anderen Hosts in den jeweiligen Clustern. Bei einfachen Hosts werden dabei Informationen zu maximal $3k-2$ anderen Hosts gespeichert, die verursachten Kosten betragen somit $O(k)$. Die Clusterleader speichern die Informationen zu den Hosts in allen Clustern, zu denen sie gehören. Die obere Grenze für die durch einen Clusterleader gespeicherten Informationen liegt daher bei $O(k \cdot \log n)$.

6.2 Probleme und Risiken

Der vorgestellte Entwurf basiert auf der Grundannahme, dass sich die Distanzen zwischen den Hosts durch die Berechnung der Abstände ihrer Abbildung in einem Koordinatensystem abschätzen lassen. Diese Berechnung geht dabei von symmetrischen und optimalen Verbindungen

zwischen den Hosts aus. Unter realen Bedingungen können diese Annahmen bei der Kommunikation zwischen den Hosts aufgrund unterschiedlicher Routingpfade aber nicht garantiert werden. Allerdings zeigen die mit umfangreichen Daten durchgeführten Tests in [COS03] nur eine geringe Abweichung zwischen den realen Werten und den berechneten Distanzen aus dem Modell.

Für den Einsatz der aktuellen Implementierung müssen in dem Netzwerk bestimmte Voraussetzungen erfüllt sein. Bei der Koordinatenberechnung und dem anschließenden Clustering verwenden die Hosts Informationen von einem Bootstrap-Host, der im Netzwerk immer verfügbar sein muss. Weiterhin setzt die Implementierung voraus, dass nach der Initialisierungsphase immer genügend Hosts als Landmarks für die Berechnung der Koordinaten von neuen Hosts verfügbar sind.

Die einmal berechneten Koordinaten für die einzelnen Hosts bleiben für den gesamten Aufenthalt des Hosts im Netzwerk konstant. Für mobile Umgebungen oder dynamische Netzwerke, die gekennzeichnet sind durch häufiges Beitreten oder Verlassen von Hosts, ist der Einsatz wegen der festen Koordinaten nur bedingt geeignet. Zur Nutzung des vorgestellten Ansatzes in diesen Einsatzbereichen muss die Implementierung um Funktionen zur Erkennung von Abweichungen und der anschließenden Neuberechnung der Koordinaten erweitert werden. Als Alternativen in diesen Umgebungen bieten sich daher Ansätze wie Vivaldi an, bei denen die Koordinaten bei der Kommunikation zwischen den Hosts angepasst werden (siehe Abschnitt 3.2.3). Dieses Verfahren eignet sich aber nur bedingt als Basis für ein Clustering, da die Koordinaten und somit auch die Distanzen der Hosts nicht feststehen sondern laufend geändert werden.

Die Validierung der Koordinaten eines Hosts nach der im Abschnitt 3.2.4 vorgestellten Lösung bleibt einer Erweiterung vorbehalten. Daher kann die Koordinatenberechnung durch fehlerhafte Angaben von einzelnen Hosts verfälscht werden und letztlich zu falschen Clusterzuordnungen führen.

7 Zusammenfassung

Im Rahmen dieser Arbeit war eine Lösung zur Reduzierung der Netzwerklast bei P2P-Anwendungen unter besonderer Berücksichtigung der Anforderungen beim P2P-Multimedia-streaming zu erarbeiten.

Die Reduzierung der Netzwerklast wird im vorgestellten Ansatz durch eine Strukturierung des Netzwerks erreicht, bei dem die Hosts in lokale Cluster gruppiert werden. Die Zuordnung der Hosts zu den lokalen Clustern erfolgt anhand ihrer Distanzen untereinander, wobei die Hosts in einem Cluster aus Netzwerksicht nahe beieinander liegen.

Der Datenaustausch in diesem Netzwerk erfolgt hauptsächlich zwischen den Hosts in einem Cluster, nur die nicht im Cluster verfügbaren Daten werden von Hosts außerhalb dieses Clusters abgefragt. Durch die verkürzten Kommunikationswege wird die Belastung des Netzwerks deutlich reduziert.

Die erarbeitete Lösung beinhaltet zwei Teilkomponenten – die Distanzbestimmung zwischen den Hosts und die Clusterverwaltung. Für beide Komponenten wurden verschiedene Ansätze betracht und evaluiert. Als Grundlage für die Distanzbestimmung wurde das Verfahren PIC (Practical Coordinates for Distance Estimation) ausgewählt. Dabei werden die Hosts als Punkte in einem geometrischen Raum abgebildet, der Abstand zwischen den Abbildungen entspricht der Distanz zwischen den Hosts. Die entwickelte Clusterverwaltung basiert auf dem NICE-Ansatz, wurde jedoch an die aufgestellten Anforderungen angepasst.

Das entworfene Verfahren ist unabhängig von externen Systemen und lässt sich somit leicht an neue Einsatzgebiete anpassen. Die Cluster werden entsprechend dem P2P-Ansatz durch die beteiligten Hosts selbständig verwaltet.

Aufgrund der konstanten Anzahl von Distanzmessungen zur Abbildung der Hosts in dem geometrischen Raum sowie einer hierarchischen Struktur der Cluster ist die vorgestellte Lösung skalierbar und schnell. In kleinen Netzwerken mit nur wenigen Hosts oder in dynamischen Netzwerken, bei denen sich die Abstände zwischen den Hosts ändern, führt der vorgestellte Ansatz zu erheblichem Overhead bei der Kommunikation zwischen den einzelnen Hosts. Für große P2P-Netzwerken bietet der Ansatz dagegen eine gute Grundlage, da dort die Netzwerke nicht hochdynamisch sind. Diese Netzwerke bestehen vielmehr aus Hosts, deren Position für die Dauer des Aufenthaltes in ihnen fest ist.

Die vorgestellte Implementierung arbeitet eigenständig und kann somit auch als Ergänzung für bestehende Systeme genutzt werden.

Der Lösungsansatz ist ausbaufähig. Speziell bei mobilen und dynamischen Netzwerken sind die Distanzen zwischen den Hosts nicht fest, somit müssen in diesen Einsatzbereichen auch die Clusterzuordnungen bei Änderungen an der Netzwerkstruktur angepasst werden.

Ein Vergleich der vorgestellten Lösung mit anderen Clusteringansätzen und eine Bewertung der Effizienz der Implementierung bleiben einer weiteren Untersuchung vorbehalten.

8 Anhang

8.1 Klassendiagramme

Die Klassendiagramme zeigen die wichtigsten Klassen, die bei der Implementierung verwendet werden.

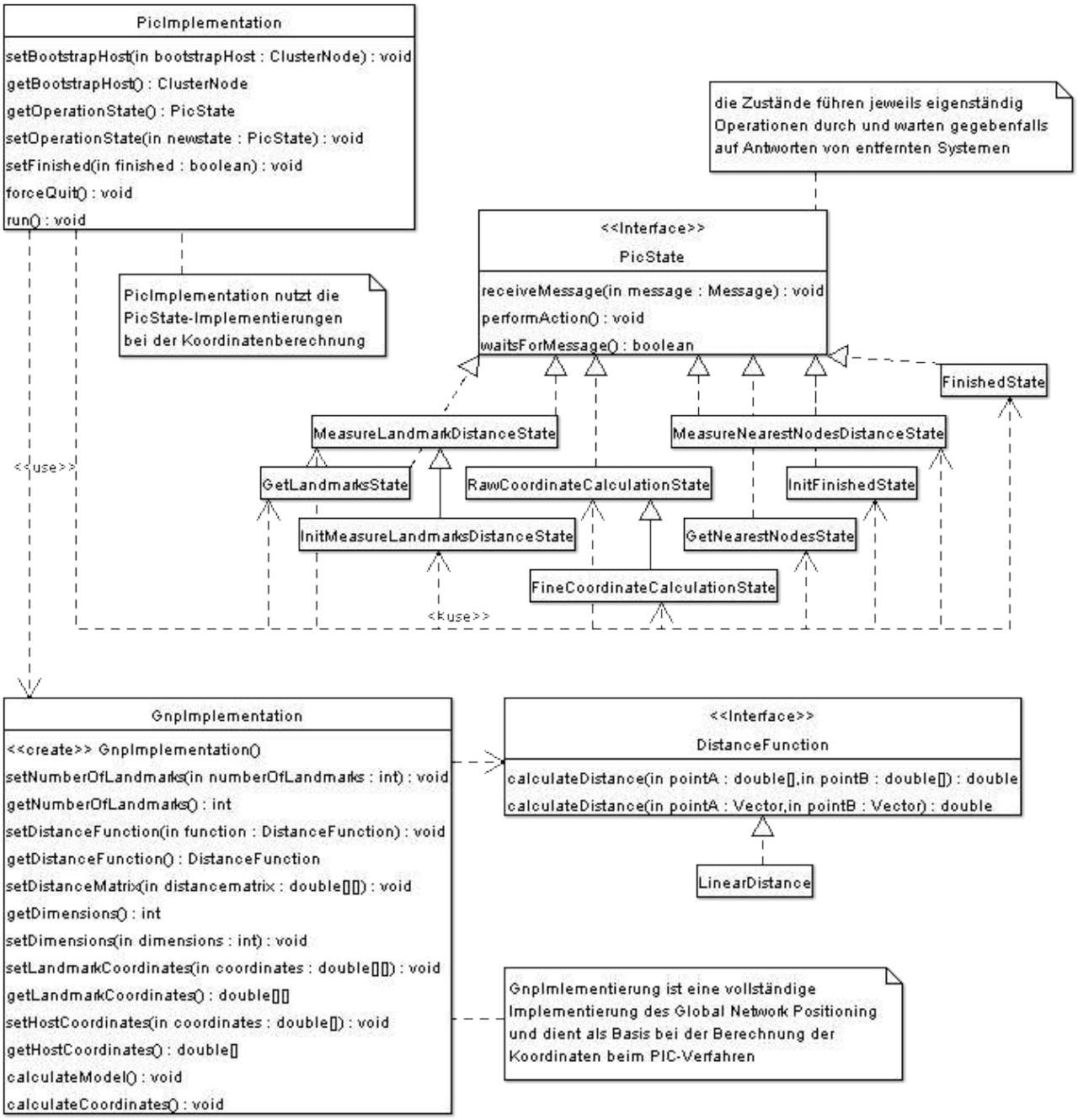

Abbildung 20: Klassendiagramm für die Koordinatenberechnung

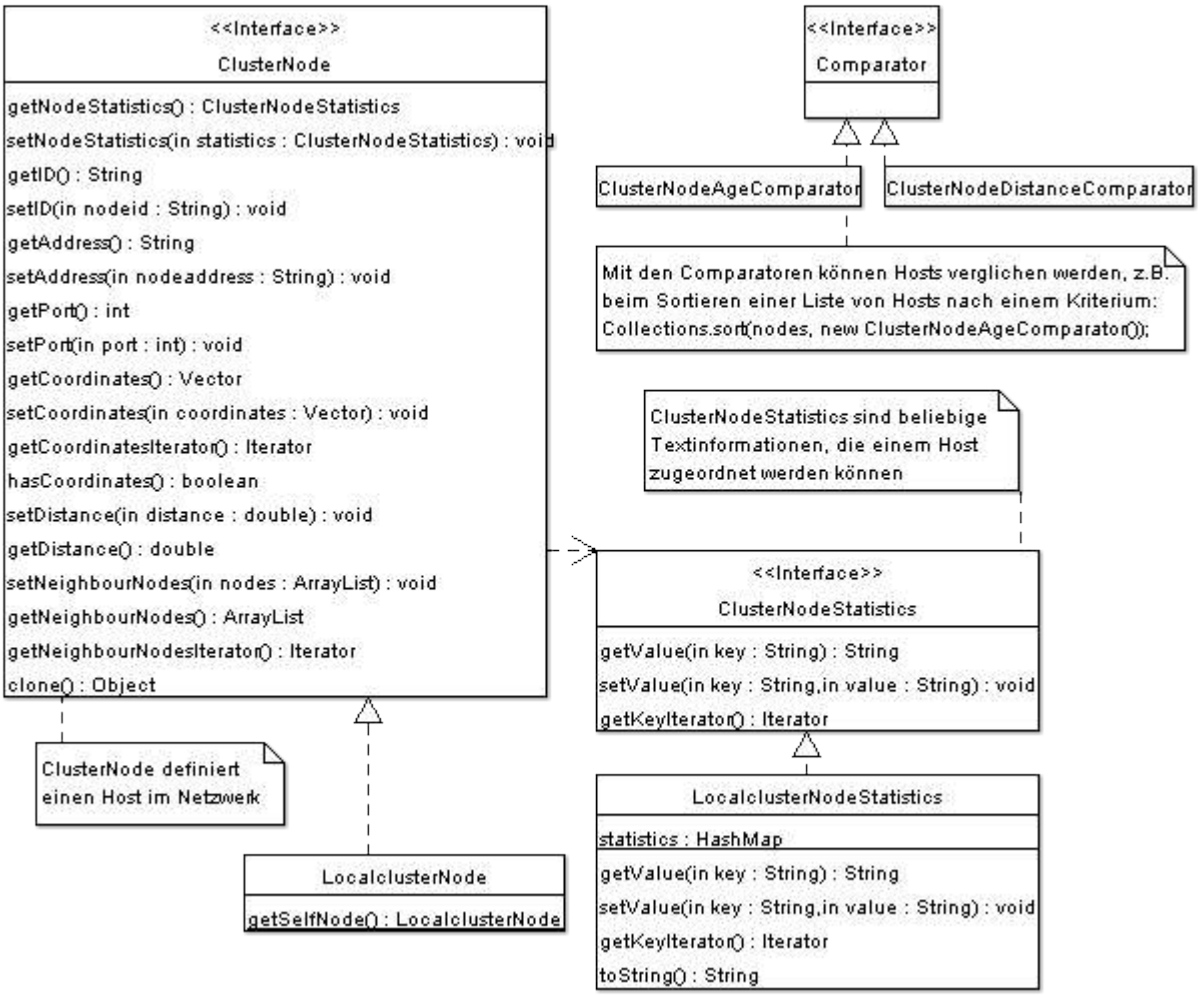

Abbildung 21: Klassendiagramm für die Nodeverwaltung

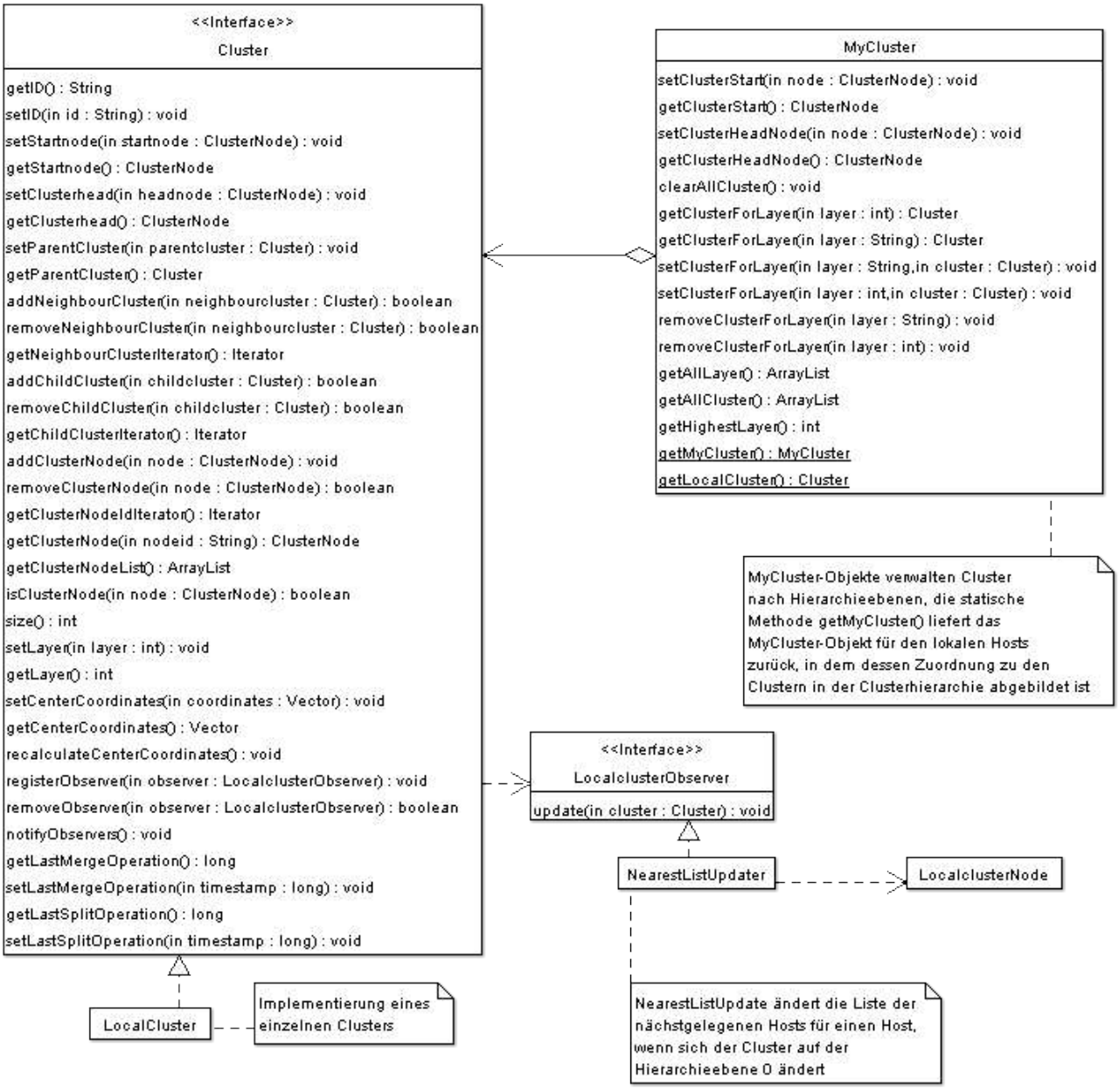

Abbildung 22: Klassendiagramm für die Clusterstrukturen

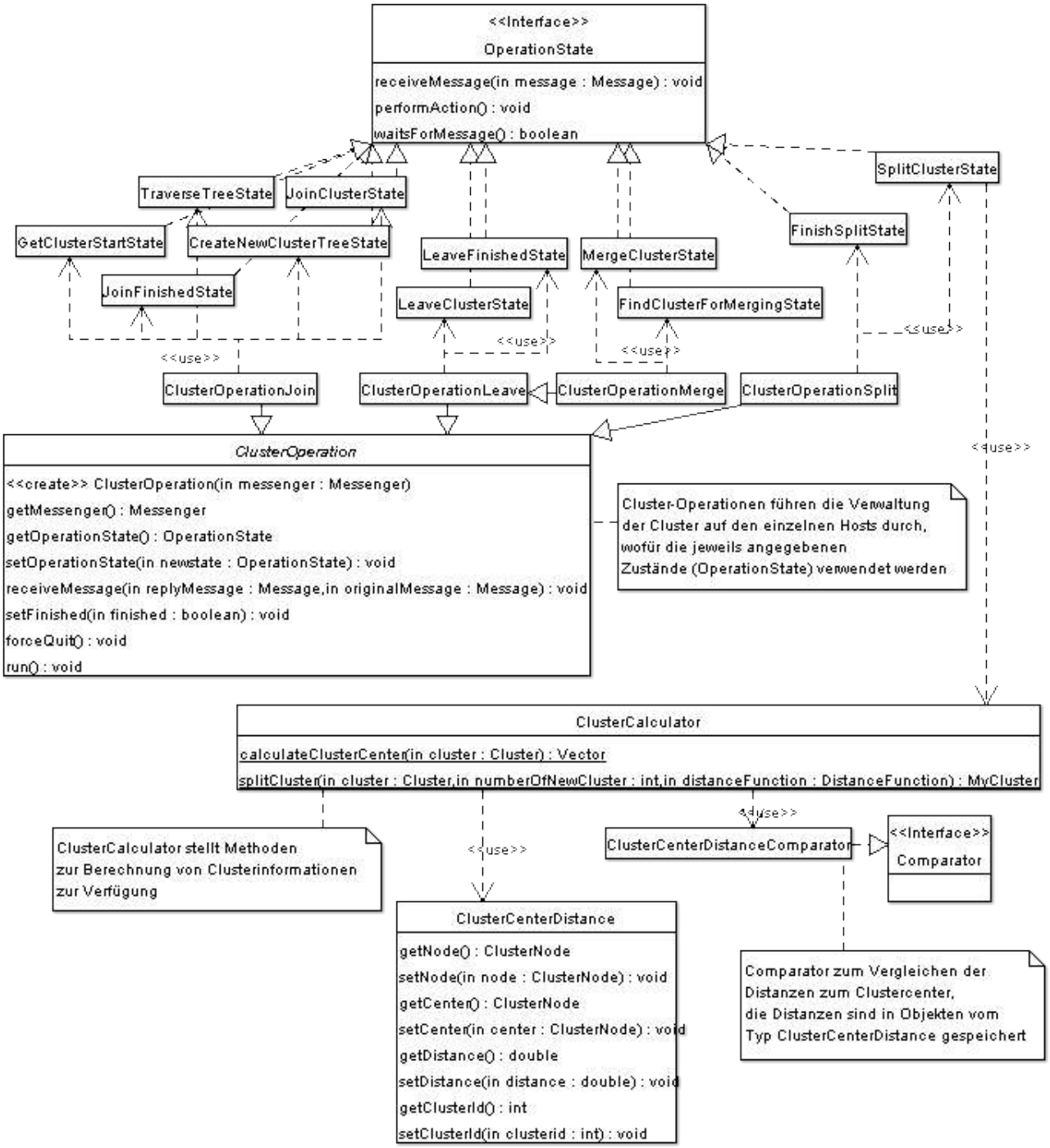

Abbildung 23: Klassendiagramm für die Clusterverwaltung

8.2 Nachrichtenaustausch in der Implementierung

Die folgenden Abbildungen zeigen den Nachrichtenaustausch zwischen den Instanzen der angegebenen Klassen mit den jeweils verwendeten Nachrichtentypen (siehe Abschnitt 5.3.1).

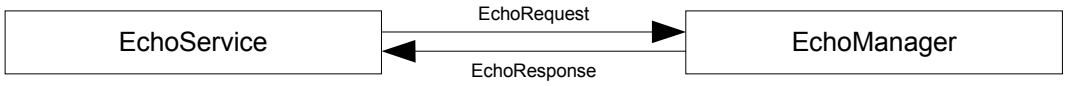

Abbildung 24: Nachrichtenaustausch bei der Distanzmessung mit Echo-Nachrichten

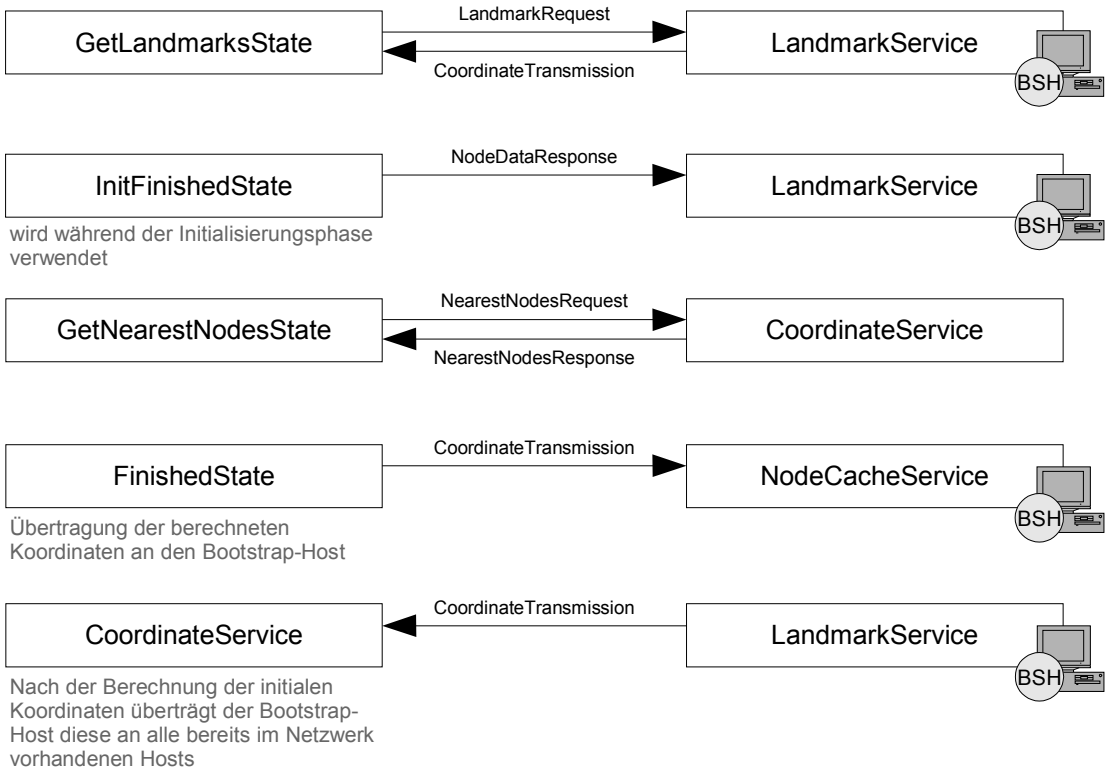

Abbildung 25: Nachrichtenaustausch bei der PIC-Implementierung

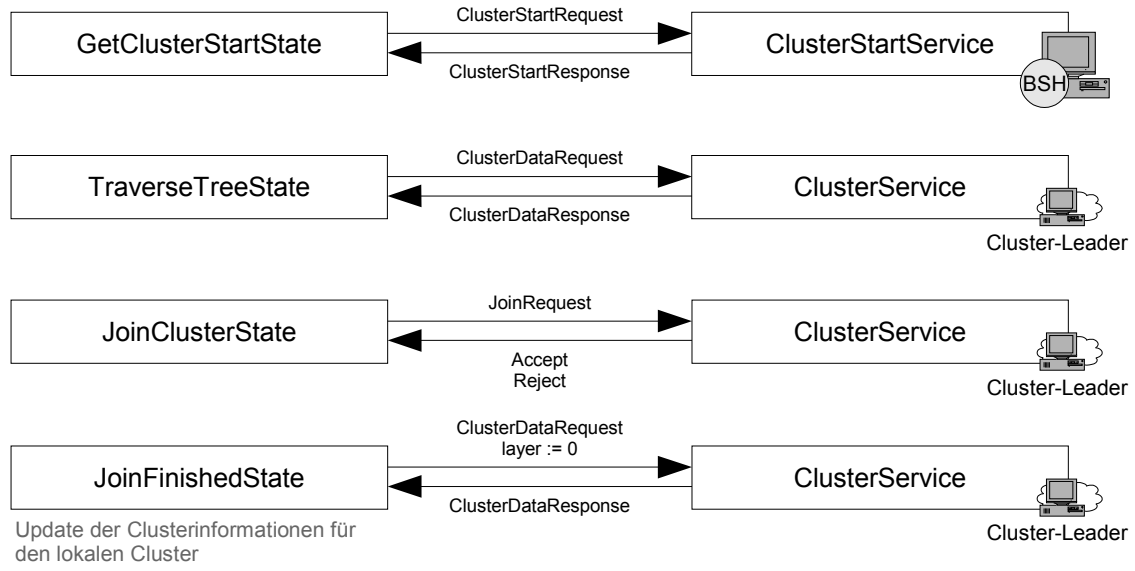

Abbildung 26: Nachrichtenaustausch beim Beitritt zu einem Cluster

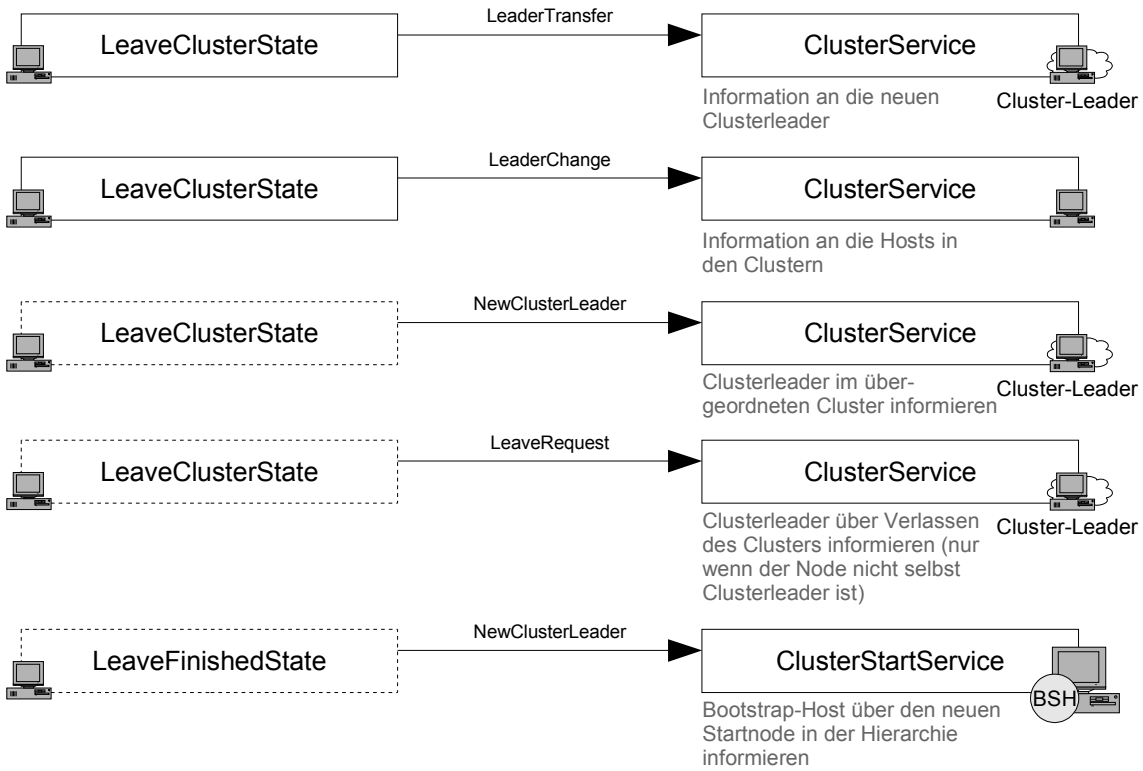

Abbildung 27: Nachrichtenaustausch beim Verlassen eines Clusters

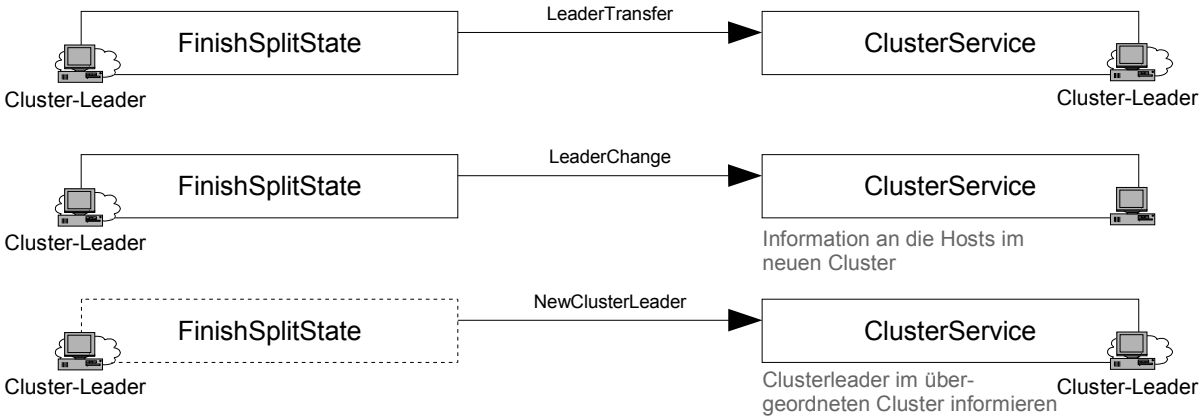

Abbildung 28: Nachrichtenaustausch beim Teilen eines Clusters

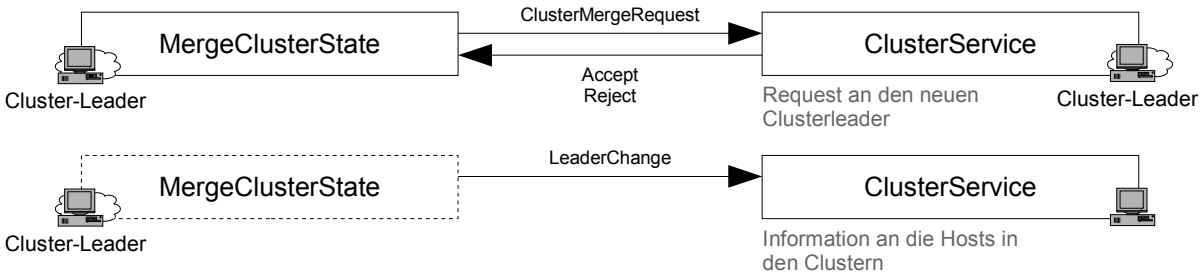

Abbildung 29: Nachrichtenaustausch beim Vereinen von Clustern

9 Literaturverzeichnis

[AGR03] A.Agrawal, H.Casanova, Clustering Hosts in P2P and Global Computing Platforms, 2003

[BAN01] S. Banerjee, B.Bhattacharjee, S.Parthasarathy, A Protocol for Scalable Application Layer Multicast, 2001

[BAN02] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scalable Application Layer Multicast, 2002

[COS03] M. Costa, M. Castro, A. Rowstron, P. Key, PIC: Practical Internet Coordinates for Distance Estimation, 2003

[COX03] R. Cox, F. Dabek, F. Kaashoek, J. Li, R. Morris, Practical, Distributed Network Coordinates, 2003

[FRA00] P. Francis, S. Jamin, IDMaps: A global Internet host distance estimation service, 2000

[GIR03] L.Giraud, J.Langou, M.Rozloznik, On the loss of orthogonality in the Gram-Schmidt orthogonalization process, 2003

[HEF02] M. Hefeeda, B. Bhargava, D. Yau, A hybrid architecture for cost-effective on-demand media streaming, 2002

[HEI98] M. Hein, TCP/IP, International Thompson Publishing 1998

[HUF02] B. Huffaker, M. Fomenkov, D.Plummer, D. Moore, K. Claffy, Distance Metrics in the Internet, 2002

[JAI00] S. Jain, R. Mahajan, B. Niswonger, Scalable Self-Organizing Overlays, 2000

[KOM02] S. Banerjee, C. Kommareddy, B. Bhattacharjee, Scalable Peer Finding on the Internet, 2002

[KRI00] B. Krishnamurthu, J.Wang, On Network-Aware Clustering of Web Clients, 2000

[MOU99] D. Mount, Design and analysis of computer algorithms (Lecture Notes), 1999

[NEL65] J.A.Nelder, R.Mead, A simplex method for function minimization, 1965

[ORE01] T. O'Reilly u.a., Peer-to-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly 2001

[PAD01] V. Padmanabhan, Determining the Geographic Location of Internet Hosts, 2001

[PAD03] V. Padmanabhan, H. Wang, P. Chou, Resilient Peer-to-Peer Streaming, 2003

[PIA03] M.Pias, J.Crowcroft, S. Wilbur, S. Bhatti, T. Harris, Lighthouses for Scalable Distributed Location, 2003

[RAT02] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, Topologically-Aware Overlay Construction and Server Selection, 2002

[SAR01] S. Saroiu, P.K.Gummadi, S.D.Gribble, A Measurement Study of Peer-to-Peer File Sharing, 2001

[SCH02] D. Schroder, R. Teichmann, K. Fischbach, Peer-to-Peer, Springer Verlag 2002

[SED92] Robert Sedgewick, Algorithmen, Addison-Wesley 1992

[SHA02] Y. Shavitt, T. Tankel, Big-Bang Simulation for embedding network distances in Euclidean space, 2002

[THE00] W. Theilmann, K. Rothermel, Dynamic Distance Maps of the Internet, 2000

[TRA03] D. Tran, K. Hua, T. Do, ZIGZAG: An Efficient Peer-to-Peer Scheme forMedia Streaming, 2003

[WEB04] K. Weber, Effizientes kooperatives Mulitmedia-Streaming in Overlay-Netzen, 2004

[ZHA01] T.Ng, H.Zhang, Predicting Internet Network Distance with Coordinates-Based Approaches, 2001

[ZHA04] X.Zhang, Q.Zhang, Z.Zhang, G.Song, W.Zhu, A Construction of Locality-Aware Overlay Network: mOverlay and its Performance, 2004