Diplomarbeit

Marcus Dramburg

Mobilität und Proximität in strukturierten Overlay-Netzen – Analysen zur Routing-Optimierung auf der Basis realer Internettopologien

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Marcus Dramburg

Mobilität und Proximität in strukturierten Overlay-Netzen – Analysen zur Routing-Optimierung auf der Basis realer Internettopologien

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Thomas C. Schmidt
Zweitgutachter : Prof. Dr. rer. nat. Hans H. Heitmann

Abgegeben am 14. August 2008

Marcus Dramburg

Thema der Diplomarbeit

Mobilität und Proximität in strukturierten Overlay-Netzen – Analysen zur Routing-Optimierung auf der Basis realer Internettopologien

Stichworte

Netzwerktopologie, Overlay-Netzwerke, Mobilität, Proximität, Internet Topologien

Kurzzusammenfassung

Diese Diplomarbeit beschäftigt sich mit der Gewinnung realer Internettopologien unter Zuhilfenahme des Verfahrens der IP Alias Resolution, um diese zur Simulation eines Pastry Overlays mit stationären und mobilen Knoten einzusetzen. Hierbei soll der Einfluss der Mobilität auf die Stabilität der DHT untersucht werden, um Aussagen über mögliche Verbesserungen des Routingverhaltens im Overlay treffen zu können.

Marcus Dramburg

Title of the paper

Mobility and Proximity in Structured Overlay Networks – Analysis to Optimizing Routing based on Realistic Internet Topologies

Keywords

Network Topology, Overlay Networks, Mobility, Proximity, Internet Topologies

Abstract

This thesis deals with generating realistic internet topologies using a method called ip alias resolution. The generated topologies should be used for simulating a pastry overlay with mobile nodes. Based on this the influence of mobile nodes on the DHT should be analysed. The results should be used to perform better routing in the overlay.

Danksagung

Ich möchte mich hiermit bei Prof. Dr. Thomas C. Schmidt und Prof. Dr. rer. nat. Hans H. Heitmann für die Betreung während der Diplomarbeit bedanken.
Dank gebührt auch Herrn Matthias Wählich für Literaturhinweise und die Unterstüzung bei der Arbeit mit Origin.
Mein besonderer Dank gilt Kerstin Dosenbach, die mir während der Diplomarbeit den Rücken frei hilt und mich mit Lob und Kritik versah, wie es gerade vonnöten war, und Korrektur las.
Weiterhin möchte ich mich noch bei Jana Westermann-Blawert für das Korrekturlesen bedanken.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Peer-to-Peer Systeme sind seit Erscheinen der ersten File-sharing Systeme Napster und Gnutella ein Feld intensiver Forschungsarbeit. Liegt in diesem Ansatz doch das verheissungsvolle Versprechen einfach aufzubauender und zu verwaltender, sowie wegen ihrer dezentralen Struktur, robuster verteilter Systeme. Auch können die komplexen Aufgaben, die solche Systeme wahrnehmen, nicht einfach durch herkömmliche Server-Clientsysteme wahrgenommen werden. Allerdings hat die Forschung der letzten Jahre gezeigt, dass mit dem Ansatz des sich selbst organisierenden verteilten Systems, dem sogenannten Overlay, ganz neue Problemstrukturen und -klassen auftreten und gelöst werden müssen. Hier hat sich gezeigt, dass die einen einheitlichen Adressraum schaffenden strukturierten Overlay Netzwerke, den teilweise noch auf Client-Server Strukturen zurückgreifenden unstrukturierten Peer-to-Peer Systemen der ersten Generation in vielerlei Hinsicht überlegen sind. Aus diesem Grund enstanden zu Anfang dieses Jahrzehnts in kurzer Abfolge viele verschiedene Algorithmen zum Aufbau solcher strukturierter Overlay Netzwerke. Genannt seien hier die eine gewisse Initialzündung auslösenden Systeme *Chord* [29] der Autoren Stoica u. a., *Pastry* [25] von Rowstron und Druschel und das *Content Addressable Network (CAN)* [24] der Autoren Ratnasamy u. a.. In der Nachfolgezeit enstanden viele Abwandlungen dieser Systeme, mit dem Ziel, bestimmte Probleme zu behandeln, die von den angeführten Algorithmen nicht gelöst wurden, wie z. B. Verfahren zur Routingoptimierung oder zur optimierten Behandlung von *churn*[1], einem der drängendsten Probleme innerhalb strukturierter Overlay Netzwerke.

Die meisten der im Zusammenhang mit Peer-to-Peer Systemen publizierten Untersuchungen bezogen sich hierbei auf stationäre Rechner- und Routernetzwerke. Die Beachtung mobiler Aspekte von am Overlay teilnehmenden Geräte ist innerhab des Forschungsgegenstands Peer-to-Peer Netzwerke jüngeren Datums. Hierbei ist anzumerken, dass wenn Mobilität Betrachtungsgegenstand wird, dann meist ausschliesslich, so dass Untersuchungen von Overlays in gemischten Umgebungen seltener sind.

Ein weiterer Aspekt der aufgeführten Entwicklung sind die Umgebungen in denen die entwickelten Algorithmen und Verfahren getestet werden können. Da, um zu statistisch relevanten Aussagen zu gelangen, hier Netzwerke mit mehreren tausend Knotenrechnern benötigt werden und diese als reales Netzwerk aufzubauen die Etats der meisten Forschungseinrichtungen überanspruchen würde, hat sich hier das Testen auf Netzwerk- und Proto-

[1] *churn* bezeichnet das Auftreten von starken Schwankungen in der Overlay Topologie durch viele neu in das Overlaynetzwerk eintretende und viele das Netz verlassende Knoten innerhalb eines kurzen Zeitraums.

kollsimulatoren, die auf diskreter Ereignissimulation basieren, durchgesetzt. Als wichtigste Simulationsumgebungen seien *ns2*[2], *OPNET*[3] oder das auf *OMNeT++*[4] aufbauende *OverSim*[5], welches speziell zum Testen von Overlay Netzwerken entwickelt wurde und bereits mit einer Vielzahl implementierter Peer-to-Peer Algorithmen aufwartet, genannt. Alle hier aufgeführten Simulationsumgebungen verfügen über Möglichkeiten die Simulationen parallel, z. B. auf einem Rechnercluster auszuführen, so dass theoretisch für die Gösse der simulierten Netzwerke keine Grenze vorgegeben ist. Hier stellt sich naturgemäss die Frage nach dem Aufbau der zum Testen benötigten Netzwerktopologien, die im einfachsten Fall erzeugte Topologien sind, aber auch auf Basis von Messungen realer Topologien gewonnen werden können.

1.1 Eingrenzung und Problemstellung

Ziel dieser Arbeit ist die Untersuchung von Veränderungen in den Routingtabellen der einzelnen Knoten eines Pastry Overlay Netzwerks unter dem Auftreten von Handover Prozessen, die sich durch den Eintritt mobiler Pastry Knoten in das Overlay ereignen. Dies soll dazu dienen, zu untersuchen ob Möglichkeiten existieren oder geschaffen werden können mit denen das Handover der mobilen Knoten beim Wechsel des Access Points durchgeführt werden kann, ohne das sich ein Einfluss auf das Routing im Overlay ergibt. Hierzu beschäftigt sich diese Arbeit zunächst mit der Gewinnung realer Internettopologien unter Zuhilfenahme eines *IP Alias Resolution* genannten Verfahrens. Auf der Basis dieser gewonnenen Topologien werden dann Testszenarien für das Testen in der OverSim Umgebung entwickelt. Die während der Simulationen gewonnenen Resultate werden einer eingehenden Analyse unterzogen, um den genauen Einfluss des Handover auf die Routingprozedur im Overlay zu bestimmen.

1.2 Aufbau und Gliederung dieser Arbeit

In Kapitel 2 wird zunächst eine Einordnung des Begriffs Internettoplogie vorgenommen, sowie die Verfahren zur Beschreibung und Analyse von Topologien vorgestellt. Mit den verschiedenen Methoden zur *IP Alias Resolution* beschäftigt sich Kapitel 3. Es schliesst mit einer Besprechung der Ergebnisse der durchgeführten Alias Resolution. In Kapitel 4 wird die Funktionsweise, des für die Simulation benutzten Pastry Algorithmus erläutert.

[2] `http://www.isi.edu/nsnam/ns/`
[3] `http://www.opnet.com/solutions/network_rd/modeler.html`
[4] `http://www.omnetpp.org/`
[5] `http://www.oversim.org/`

Anschliessend wird in Kapitel 5 auf die besonderen Problemstellungen von mobilen Overlay Knoten eingegangen und die Proximitätsbeziehungen zwischen den Teilnehmern des Overlay beleuchtet. Daraufhin wird in Kapitel 6 die benutzte Simulationsumgebung, die notwendigen Anpassungen der Simulationsumgebung, sowie die Ergebnisse der Simulationsläufe vorgestellt und diskutiert. Die Arbeit schliesst mit einem Fazit.

2 Topologien in Computer Netzwerken

Seit dem Anfang der 1990er Jahre unterliegt das Internet einem immensen Wachstum, sowohl in Hinsicht auf den strukturellen Ausbau in den Routernetzwerken und End-Host-Systemen, als auch im Hinblick auf die zu verarbeitenden Datenmengen. Dies führt allmählich zu einer Überlastung der bestehenden Infrastruktur, vor allem im Bereich der Router-Netzwerke, der mit der Entwicklung neuer Protokolle und dezentraler Kommunikationsstrukturen zu begegnen versucht wird. Damit neue Kommunikationsverfahren effizient auf der Basis der bestehenden Netzwerkinfrastruktur implementiert werden können, ist es wichtig, Erkenntnisse über den Aufbau bestehender Rechnernetzwerke zu gewinnen. Da die Kommunikationsnetze nicht öffentlicher Natur sind, sondern sich in der Hand von wirtschaftlich operierenden Unternehmen befinden, die Interna zu ihrer Netzwerkstruktur als Firmengeheimnis ansehen, ist das Entdecken von realen Netzwerkstrukturen bisweilen nicht ganz einfach, da Informationen meistens nur von ausserhalb der bestehenden Teilnetze zu gewinnen sind. Hierbei ist von zentralem Interesse, vor allem der Aufbau bestehender Routernetzwerke, als auch die Frage welche Autonomen Systeme (AS), die die Routernetzwerke bilden, wie mit anderen Autonomen Systemen verbunden sind. Um die zur Beschreibung der betrachteten Topologien notwendigen Daten zu gewinnen kommen meist *traceroute* basierte Verfahren zur Anwendung. Solche Datensammlungen unterliegen, da die Strukturen des Internets einem permanenten Wandel ausgesetzt sind, einem natürlichen Alterungsprozess und müssen somit in Intervallen neu erhoben werden. Dabei haben sich im besonderen CAIDA[1] (Cooperative Association for Internet Data Analysis) und das DIMES[2] Projekt durch den Aufbau einer Messinfrastruktur und das kontinuierliche Sammeln von Topologie Daten ausgezeichnet.
Damit die gesammelten Daten analysiert werden können, besteht die Notwendigkeit die gewonnenen Topologien in eine formale Struktur zu überführen und für diese eine Metrik zu finden, anhand derer Topologien vergleichbar sind. Die formale Struktur zur Beschreibung einer Topologie wird durch einen Graphen gebildet. Vergleichbar und analysierbar werden die Graphen, aufgrund ihrer Grösse von bis zu mehreren Millionen Knoten allein über statistische Verfahren. Hierzu sind innerhalb der sich mit Netzwerkstrukturen beschäftigenden Graphentheorie in den letzten Jahren eine Vielzahl neuer Verfahren enstanden (siehe hierzu

[1] CAIDA Homepage unter: `http://www.caida.org/home/`
[2] siehe unter: `http://www.netdimes.org/new/`

[22]), wie die *degree distribution* und *joint degree distribution*. Im folgenden wird nun eine begriffliche Einordnung der verschiedenen Sichten auf die von Computernetzwerken gebildeten Topologien vorgenommen, diese orientiert sich im wesentlichen an der Darstellung von Donnet und Friedman in [5]. Anschliessend werden die mathematischen Verfahren zur Beschreibung und Analyse von Netzwerktopologien vorgestellt. Das Kapitel schliesst mit einer Betrachtung der verwendeten Topologiedatensätze.

2.1 Einordnung

Es lassen sich im wesentlichen drei verschiedene Ebenen, von denen ausgehend eine Beschreibung der Netzwerk-Topologie erfolgen kann, unterscheiden. Dies sind die *link layer topology*, die *network layer topology*, die oft auch einfach als *internet topology* bezeichnet wird und die *overlay topology*.

2.1.1 Link Layer Topologie

Diese Sichtweise beschreibt die Netzwerktopologie über den physikalischen Aufbau des Netzes. Gegenstand des Interesses sind hier Netzwerkkarten, Switches, Bridges und die Art ihrer Verbindungen, wie Kupfer- oder Glasfaserkabel, sowie die Eigenschaften dieser Verbindungen, wie Signallaufzeiten und Übertragungsfehler. Diese Informationen sind nur schwer von ausserhalb einer Domain zu gewinnen und beschränken sich deshalb meist auf Intra-Domain-Sicht. Wichtig sind solche Topologiebeschreibungen für das Netzwerkmanagement und die Fehlersuche bei Störungen.

2.1.2 Overlay Topologie

Eine Overlay-Topologie ist die Topologie eines Peer-to-Peer-Systems. Diese kann strukturiert oder unstrukturiert sein. Strukturierte Overlays wie Chord oder CAN können über eine DHT beispielhaft beschrieben werden. Da der Aufbau eines Overlays hinsichtlich der partizipierenden Knoten über die Zeit einer starken Fluktuation ausgesetzt sind, muss hier mit Schnappschüssen auf den Systemzustand gearbeitet werden, wenn ein Gesamtzustand des Systems zu einem bestimmten Zeitpunkt betrachtet werden soll. Alternativ kann auch die zeitliche Entwicklung des Systems beschrieben werden.

2.1.3 Internet Topologie

Der Bereich der Internet Topologie lässt sich wiederum in vier unterschiedliche Betrachtungsebenen gliedern. Ausschlaggebend für die Unterteilung ist das im Netzwerkgraph knotenbildende Element. Unterschieden wird in *IP interface level*, *router level*, sowie in *point of presence (PoP) level* und *autonomous system (AS) level*.

2.1.3.1 IP Interface Level

Auf der Ebene des IP Interface Levels werden die Netzwerkschnittstellen als Knotenelemente des Netzwerkgraphen betrachtet. Dabei werden als Kanten nicht nur bloße Punkt zu Punkt Verbindungen der Netzwerkschnittstellen untereinander betrachtet, sondern z. B. auch getunnelte Verbindungen zwischen den Knoten. Die Topologie auf Interface Level kann mit *traceroute* aufgedeckt werden.

2.1.3.2 Router Level

Router Level Topologien können als Verdichtung des Interface Level angesehen werden, da diese die Tatsache ausser Acht lässt, das auf einem Hostsystem mehrere Netzwerkschnittstellen vorhanden sein können. Es sollen hierbei möglichst realistische Abbilder der im Internet vorhandenen Routernetzwerke entstehen. Um aus der schnittstellenbasierten Sicht, wie sie die Ausgabe von *traceroute* liefert, eine Router Level Topologie zu gewinnen wird ein Verfahren namens *IP Alias Resolution* angewendet. Unter diesem Begriff werden verschiedene Methoden zusammengefasst, mit deren Hilfe verschiedene von einem Router beheimatete (*multihoming*) Netzwerkinterfaces zu einem Knoten vereint werden können (hierzu s. a. 3). Die Kanten werden von den Punkt zu Punkt Verbindungen der Netzwerkkarten gebildet.

2.1.3.3 PoP Level

Auf dieser Ebene kommen die geografischen Standorte der Routersysteme zur Geltung. Dabei werden Ansammlungen von Routern eines Autonomen Systems, die an einem geographischen Ort angesiedelt sind, zu einem Knoten des Netzwerkgraphen zusammengefasst. Hierbei kann ein geografischer Ort eine Stadt, ein Ortsteil oder ein Gebäude bedeuten. Die Kanten zwischen zwei Knoten eines AS an unterschiedlichen Orten können dabei als das Backbone des AS angesehen werden, die Kanten zwischen verschiedenen Autonomen Systemen als *access*- oder *peering links*.

2.1.3.4 AS Level

Auf Ebene der Autonomen Systeme werden alle einem AS zugehörigen Routersysteme zu einem Knoten zusammengefasst. Die Kanten des Graphen können verschiedene Beziehungen zwischen Autonomen Systemen anschaulich machen. Dies sind im wesentlichen *access links* und *peering links*, können aber auch als die Modellierung von Geschäftsbeziehungen zwischen Providern und Customern verstanden werden, siehe hierzu auch Donnet und Friedman in [5].

2.2 Mathematische Modelle zur Beschreibung und Analyse

Die formale Struktur einer Topologie wird durch einen Graphen beschrieben. Dieser besteht aus Knoten *(vertex, pl. vertices)* und Kanten *edge*, sowie den Verbindungen, die die Kanten zwischen den Knoten erzeugen. Der Begriff *degree* beschreibt hierbei die Anzahl der Kanten, die an einem Knoten anliegen (dies muss nicht notwendigerweise gleich der Anzahl von Knoten sein, zu denen eine Verbindung besteht). Für den Fall eines gerichteten Graphen existieren für jeden Knoten ein *indegree* und ein *outdegree*. Im folgenden werden die für die Topologiebetrachtung wesentlichen Eigenschaften des Graphen erläutert.

2.2.1 Clustering

In vielen Netzwerken lässt sich beobachten, dass wenn Knoten X mit Knoten Y und Knoten Y mit Knoten Z verbunden ist auch eine erhöhte Wahrscheinlichkeit besteht, dass Knoten X mit Z verbunden ist. Diese Eigenschaft wird als Clustering oder Transivität bezeichnet. Die Maßzahl für das Clustering ist der *clustering coefficient C* [22] und berechnet sich wie folgt:

$$C = \frac{3 \times Anzahl\ vorhandener\ Dreiecke}{Anzahl\ verbundener\ Triple} \tag{2.1}$$

Hierbei stellt der Faktor 3 sicher, dass das Ergebnis im Intervall $0 \leq C \leq 1$ liegt, da es dem Umstand Rechnung trägt das ein Dreieck aus drei Triplen besteht. In einfachen Worten ist der *clustering coefficient* die mittlere Wahrscheinlichkeit, dass zwei Knoten die Nachbarn eines dritten Knoten sind, selber auch Nachbarn sind. Ein alternatives Verfahren zur Berechnung des *clustering coefficient* stellen Watts und Strogatz in [31] vor. Hierbei

wird zunächst für jeden Knoten ein lokaler *clustering coefficient* C_i berechnet:

$$C = \frac{Anzahl\ Dreiecke\ verbunden\ mit\ Knoten\ i}{Anzahl\ zentrierter\ Triple\ auf\ Knoten\ i} \tag{2.2}$$

Für Knoten mit 0 oder 1 Kante wird C_i auf Null gesetzt. Somit berechnet sich der *clustering coefficient* für das gesamte Netzwerk als Durchschnitt der lokalen Werte:

$$C = \frac{1}{n}\sum_i C_i \tag{2.3}$$

Für das in Abb. 2.1 dargestellte Netzwerk sind die lokalen *clustering coefficients* $1,\ 1,\ \frac{1}{6},\ 0\, und\, 0$, somit liesse sich *C* nach Gleichung 2.3 zu $\frac{13}{30}$ berechnen.

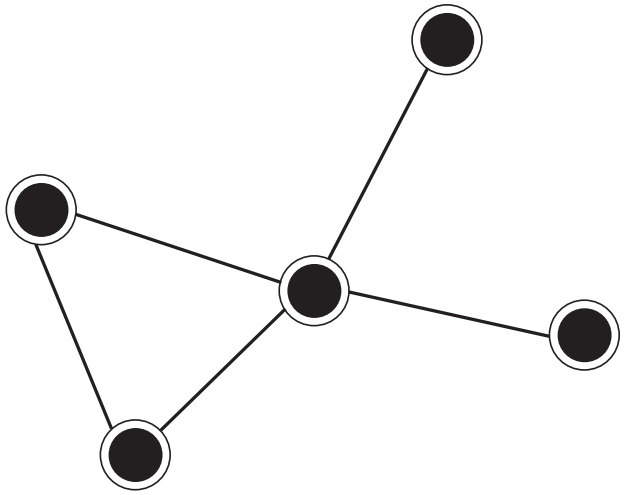


Abbildung 2.1: Darstellung zum clustering coefficient C in Gleichung (2.1). Das dargestellte Netzwerk enthält ein Dreieck und acht Triple, so dass nach Gleichung 2.1 $C = \frac{3}{8}$, Nach Quelle: [22]

2.2.2 Degree Distribution

Die Degree Distribution beschreibt die Vorkommenshäufigkeit von Knoten mit einer bestimmten Kantenanzahl. Dabei finden sich in der Literatur zwei Betrachtungsweisen. Einmal die absolute Verteilung, die inhaltlich einem Histogramm gleicht, aber wegen der ungleichen Verteilung am oberen Rand (wenige Knoten mit grosser Kantenanzahl) meist als doppelt logharitmischer Plot dargestellt wird. Zum anderen als Verteilung der Wahrschein-

lichkeiten dafür, dass ein zufällig betrachteter Knoten die Kantenanzahl k hat. Hierzu wird oft die kumulative Wahrscheinlichkeitsfunktion benutzt [22]:

$$P_k = \sum_{k \leq k'} p_{k'}. \tag{2.4}$$

Dies dient wiederum dazu den dünn gestreuten oberen Rand, besser in die Verteilung mitaufzunehmen.

2.2.3 Joint Degree Distribution

Die Joint Degree Distribution beschreibt die Verteilung der Wahrscheinlichkeiten dafür, dass eine zufällig ausgewählte Kante Knoten mit der Kantenanzahl k_1 und k_2 verbindet. Für einen ungerichteten Netzwerkgraphen mit der Kantenanzahl M und $m\,(k_1, k_2)$ als Anzahl der Kanten, die einen Knoten des Typs k_1 mit einem Knoten des Typs k_2 [15] verbinden, ermittelt man die Wahrscheinlichkeit über

$$p_{k_1,k_2} = \frac{m\,(k_1, k_2)}{M}. \tag{2.5}$$

2.2.4 Assortativität

Assortativität beschreibt die Neigung von Knoten eines bestimmten Typs Verbindungen zu Knoten eines bestimmten Typs einzugehen. Dies wird auch als *assortative mixing* bezeichnet. Der Grad der Assortativität wird dabei über den *assortativity coefficient r* beschrieben. Für r gilt $-1 \leq r \leq 1$. Hierbei steht -1 für ein Netzwerk in dem alle Kanten Knoten unterschiedlichen Typs verbinden *(perfect disassortativity)*, 1 für ein Netzwerk in dem alle Kanten ausschliesslich Knoten gleichen Typs verbinden *(perfect assortativity)* und 0 für ein sogenanntes *random mixed network* (hierzu s. a. [22]). Nach Newman in [21] gilt für den einfachen Fall eines ungerichten Graphen mit N Knoten und M Kanten, sowie der Wahrscheinlichkeitsverteilung p_k der Kanten, dass man, wählt man eine zufällige Kante aus, für den Knoten, in den diese Kante läuft, die Menge der übrigen Kanten, wieder als Verteilung betrachten kann. Newman nennt dies *remaining degree* q_k. Diese Verteilung berechnet sich in der normalisierten Form über

$$q_k = \frac{(k+1)\,p_{k+1}}{\sum_j j p_j}. \tag{2.6}$$

Mit der *joint degree distribution* der *remaining degrees* e_{jk}, für die im ungerichteten Graphen $e_{jk} = e_{kj}$, sowie

$$\sum_{jk} e_{jk} = 1, \qquad \sum_{j} e_{jk} = q_k \tag{2.7}$$

gelten und die in einem *random mixed network* gleich $q_j q_k$ ist, sowie der Varianz

$$\sigma_q^2 = \sum_k k^2 q_k - \left(\sum_k k q_k \right)^2 , \tag{2.8}$$

lässt sich der normalisierte *assortativity coefficient* wie folgt ermitteln:

$$r = \frac{1}{\sigma_q^2} \sum_{jk} jk \left(e_{jk} - q_j q_k \right) . \tag{2.9}$$

Zur einfacheren rechnergestützten Berechnung gibt Newman in [21] noch die folgende äquivalente Formel an:

$$r = \frac{M^{-1} \sum\limits_i j_i k_i - \left(M^{-1} \sum\limits_i \frac{1}{2} \left(j_i + k_i \right) \right)^2}{M^{-1} \sum\limits_i \frac{1}{2} \left(j_i^2 + k_i^2 \right) - \left(M^{-1} \sum\limits_i \frac{1}{2} \left(j_i + k_i \right) \right)^2} . \tag{2.10}$$

2.3 Topologiedatensätze

Im folgenden Abschnitt werden die für Topologieerzeugung verwendeten Datensätze beschrieben und ihre Weiterverarbeitung mit dem Topologiegenerator BRITE[3] erläutert.

[3] Homepage unter: `http://www.cs.bu.edu/BRITE/`

2.3.1 Die verwendeten Datensätze

Für die durchzuführenden Simulationen mit Omnet++[4]/Oversim[5] wurden Topologiedaten aus drei verschiedenen Quellen genutzt. Die ersten beiden Datensätze sind die vom DIMES Projekt zur Verfügung gestellten Router Level Topologien von Dezember 2007 und Januar 2008. Die nächsten beiden Datensätze stammen von CAIDA [17] aus der *skitter*-basierten IPv4 Routed Topologie Datensammlung. Hier wurden die jeweils letzten Messzyklen der Monate Dezember 2007 und Januar 2008 ausgewählt, diese von den elf Monitorpunkten, die in beiden Monaten Messdaten lieferten. Die letzten beiden Datensätze wurden aus den Beständen des CAIDA ARK Projekts genommen. Die Messungen sind in zwei Teams aufgeteilt. Team 1 startete im September 2007 mit elf Monitorpunkten und UDP-basiertem *traceroute*. Team 2 startete im Januar 2008 zunächst mit nur einem Monitorpunkt und *paris-traceroute*[6] als Messwerkzeug. Da von Team 2 erst ab Januar 2008 Messdaten zur Verfügung stehen und die Messungen über den ersten Monat nur von einem Monitorpunkt aus erfolgten, wurden zwei Datensätze aus den Messungen von Team 1 gewählt. Hierbei wieder die beiden letzten Messzyklen der Monate Dezember 2007 und Januar 2008. Trotzdem wurden aus dem Datenbestand von Team 2 drei Zyklen aus den Monaten März, April, Mai 2008 zu Vergleichen herangezogen.

2.3.2 Aufarbeitung der Daten und Topologieerzeugung

Da in den DIMES Datensätzen das Problem entdeckter Kanten zwischen Interfaces, deren Adresse nicht ermittelt werden kann, über die Markierung mit künstlichen Adressen gelöst wird, und diese zu Fehlern in der Analyse führen kann, müssen die Datensätze mit einem Perl-Skript[7] vorbehandelt werden. Die Skitter Datensätze müssen mithilfe der Software *sk-analysis-dump*[8] vom binären *arts++*-Format in ein textuelles umgewandelt werden, dazu wird die *arts++*-Bibliothek[9] benötigt. Ähnliches gilt für die neueren CAIDA Datensätze aus dem *Archipelago*-Projekt. Diese liegen im binären *warts*-Format vor und können mit dem Programm *sc-analysis-dump* das Teil des *scamper*-Softwarepakets[10] ist oder mit der in Ruby geschriebenen *wartslib*[11], die ein einfaches Skripting-Interface zum Auslesen von

[4] Omnet++ ist eine Simulationsumgebung für Netzwerkstrukturen auf der Basis diskreter Ereignisse. Zu beziehen ist das Programm unter: `http://www.omnetpp.org/filemgmt/viewcat.php?cid=2`

[5] Oversim setzt auf einer Erweiterung zu Omnet++, dem INET-Framework(Nachbildung von Ethernet und gängigen Netzwerkprotokollen), auf. Es stellt verschieden Overlay-Protokolle zur Verfügung, wie Chord und Pastry. Download unter: `http://www.oversim.org/wiki/OverSimDownload`

[6] siehe hierzu: `http://www.paris-traceroute.net/newtraceroute.htm`

[7] download unter: `http://www.realmv6.org/brite-extension/dimessyntaxfix.pl`

[8] download unter: `http://www.caida.org/tools/measurement/skitter/sample_code/code/`

[9] zu beziehen unter: `http://www.caida.org/tools/utilities/arts/download/`

[10] erhältlich auf: `http://www.wand.net.nz/scamper/`

[11] download unter: `http://rb-wartslib.rubyforge.org/`

warts-Dateien zur Verfügung stellt, in das Textformat der Skitter-Dateien überführt werden. So muss für das neue Format kein neuer Import für den BRITE Topologie Generator entwickelt werden. Anschliessend wurden die Datensätze einzeln in BRITE importiert und der Graph im BRITE eigenen Format exportiert. Die Eigenschaften der erzeugten Topologien wurden mit einem in BRITE vorhandenen Analyse Modul extrahiert. Hierbei interessierten vor allem der *mean degree*, die *degree distribution*, die *joint degree distribution*, sowie der *assortativity coefficient*. Abschliessend wurde mit der Statistik Software *Origin*, die Standardabweichung der *degree distribution* ermittelt und *plots* der *degree distribution* (Abb. 3.6 bis 3.7) und der normalisierten *joint degree distribution* (Abb. 3.8 bis 3.14) erzeugt. Die Besprechung der Ergebnisse folgt in Kapitel 3 im Abschnitt 3.8.

3 IP Alias Resolution

Die IP Alias Resolution dient der Zusammenführung von verschiedenen Netzwerkschnittstellen eines Routers zu einem Knoten im erzeugten Netzwerkgraphen. Dies soll realistischere Abbilder der im Internet vorhandenen Routernetzwerke ermöglichen. So nutzt CAIDA in ihrem im September 2007 gestarteten *Archipelago Measurement Project* (ARK) mit *iffinder* eine Software zur IP Alias Resolution[1]. Zum Zwecke der Alias Resolution sind seit 1997 verschiedene Verfahren entwickelt worden. Diese lassen sich nach Donnet und Friedman in [5] in aktive und analytische Verfahren unterteilen. Für Datensätze mit IPv4 Adressen existieren zur Zeit sechs verschiedene Methoden. Die vier Verfahren *address based method*, *IP ID based method*, *DNS based method* und *TTL-limited with Record Route Option method* werden zu den aktiven gezählt und die zwei Methoden *graph based method* und *analytical alias resolver method* zu den analytischen. Zur Zeit sind zwei Programme zur IP Alias Resolution erhältlich, von CAIDA das Tool *iffinder* und als Teil des *scriptroute* Projekts die Softwarekomponente *ally*[2]. Die beiden Programme nutzen jeweils mehrere der nachfolgend aufgeführten und erläuterten Verfahren.

3.1 Address Based Method

Diese Methode findet zuerst Erwähnung bei Pansiot und Grad in [23]. Dies ist eine vom Prinzip sehr einfache Methode (hierzu s. a. Abb. 3.1). Von einer Quelle wird ein UDP Datagramm an eine hohe, nicht belegte Portnummer einer angenommenen Router Netzwerkschnittstelle X verschickt. Antwortet nun der Router mit einem *port unreachable* und die Quelladresse der ICMP Nachricht ist Y, dann sind X und Y Aliase desselben Routers. Das Verhalten, dass ein Router seine ICMP Nachrichten nicht über das Interface absetzt auf dem die Ursprungsnachricht empfangen wurden, sondern über ein anderes Interface, wurde von Pansiot und Grad beobachtet. Die Autoren weisen allerdings auch darauf hin, dass so nur ein Teil der Aliase entdeckt werden können, in ihrer Untersuchung fanden sie in einem Satz mit

[1] CAIDA Homepage unter: `http://www.caida.org/home/`
zum ARK Project siehe: `http://www.caida.org/projects/ark/`
zur IP Alias Resolution siehe: `http://www.caida.org/tools/measurement/iffinder/`

[2] dowload unter: `http://www.cs.washington.edu/research/networking/rocketfuel/software/ally-0.1.0.tar.gz`

10.000 IP Adressen 200 Aliase. Mittlerweile existieren verschiedene Varianten, die als Testnachricht kein UDP Datagramm, sondern ICMP *echo requests* oder *timestamp* Nachrichten versenden. Aufgrund der Einfachheit des Verfahrens wird es von allen IP Alias Resolution Programmen implementiert.

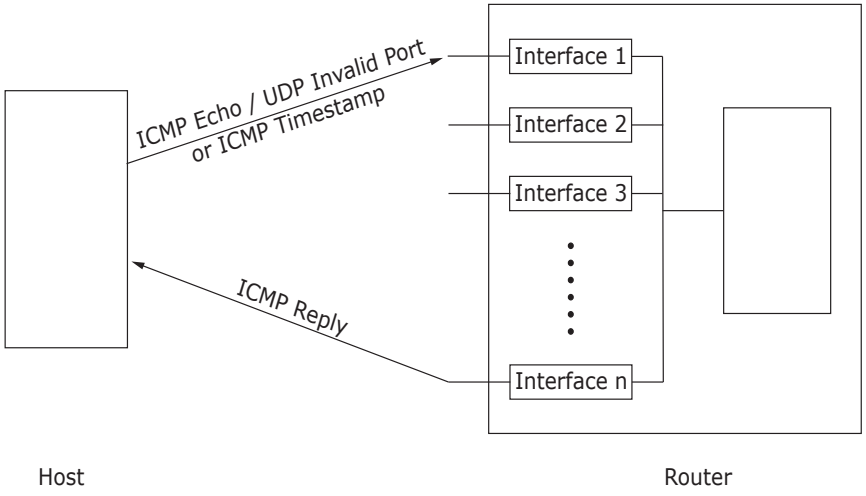


Abbildung 3.1: Darstellung des erhofften Routerverhalten bei adressbasierten Verfahren.

3.2 IP Identification Based Method

Dieses Verfahren basiert auf der Nutzung des ID Feldes im IP Header, das eigentlich für die Zuordnung von Paketen bei Fragmentierung angedacht war, und wurde von Spring, Dontcheva, Rodrig und Wetherall entwickelt und in [27] erläutert. Besitzt ein Router für alle seine Netzwerkschnittstellen einen einzigen IP ID Zähler, inkrementiert er diesen mit jeder neu versendeten Nachricht, und setzt das ID Feld für jede der von ihm versendeten Nachrichten, kann dieses Verfahren genutzt werden. Eine weitere Vorraussetzung ist, dass durch vorhergende Anwendung eines analytischen Verfahrens zwei Kandidaten ermittelt wurden. An diese beiden Alias Kandidaten werden kurz hintereinander UDP Datagramme mit einer nicht erreichbaren Portnummer gesendet und auf die *ICMP port unreachable messages* gewartet, diese sind in ein IP Datagramm gehüllt, haben somit eine ID, z. B. x und y. Nun wird an beide Adressen ein weiteres UDP Datagramm verschickt. Angenommen die ID der dritten Antwort ist z und das der ersten Antwort war x, sowie die ID der vierten Antwort ist w und die der zweiten war y. Gilt dann $x < y < z < w$ und sind $y - x$ und $w - z$ sehr klein (Spring u. a. nennt hier einen Richtwert von 200), dann sind die Adressen wahrscheinlich Aliase (s. a. Abb. 3.2). Dieses Verfahren wird von *ally* als hauptsächliches Verfahren genutzt und von *iffinder* als Verfahren um gefundene Aliase abzusichern.

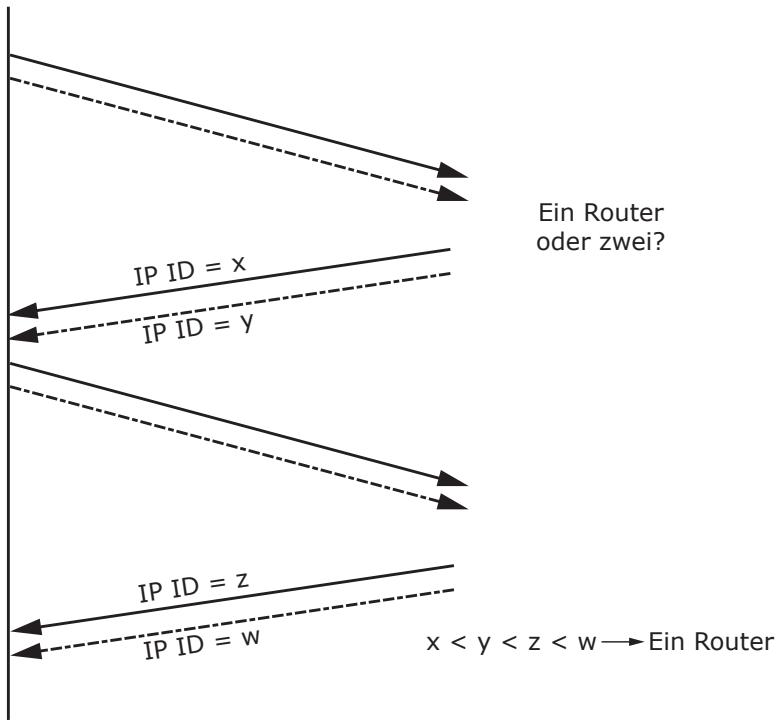


Abbildung 3.2: IP Alias Resolution mit IP ID. Nach Quelle: [27]

3.3 DNS Based Method

Benutzt ein *Internet Service Provider* (ISP) innerhalb seines AS eine systematische Namenskonvention für die Netzwerkschnittstellen seiner Router (so sind nach Spring u. a. z. B. *sl-bb21-lon-14-0.sprintlink.net* und *sl-bb21-lon-8-0.sprintlink.net* Aliase für denselben Backbone-Router in London), kann dieses Verfahren benutzt werden. Dieses Verfahren ist arbeitsintensiv, hat aber den Vorteil, dass es auch bei Routern verwendet werden kann, die sich der Antwort auf Test Datagramme entziehen. Aufwendig ist das Verfahren, da für jedes AS ein Regelsatz für den Namensraum von Hand erstellt werden muss. Das Verfahren ist in *ally* implementiert, lässt sich aber nur nutzen, sofern Regelsätze für die untersuchten Netze vorhanden sind.

3.4 TTL-Limited with Record Route Option Method

Dieses Verfahren basiert auf der IP Option *record route*, wie sie in RFC 791 aufgeführt ist und mit der die Route, die ein Datagramm nimmt, im Header aufgezeichnet werden kann. Eine Darstellung des von Sherwood und Spring entwickelten Verfahren findet sich in [26]. Da der IP Header eine Größenbegrenzung auf maximal 60 Bytes hat, wovon 20 Bytes auf den Standard IP Header entfallen, und von den übrigen 40 Bytes eines für das Optionsfeld, eines für den Pointer auf den nächsten freien Platz im Record und eines für die Länge des optionalen IP Header gebraucht werden, lassen sich maximal 9 Adressen aufzeichnen. Das Verfahren versucht sich zu Nutze zu machen, dass die *ICMP time exceed* Nachrichten bei einer traceroute Anfrage auf dem eingehenden Netzwerkinterface abgesetzt werden und somit dessen Quelladresse im traceroute Pfad auftaucht, mit der record route Option hingegen die Adresse der Schnitsstelle aufgezeichnet wird, über die das Datagramm den Router verlässt. Werden nun die IP Datagramme mit der eingeschalteten record route Option und mit einem begrenzten TTL-Feld (zur Veranschaulichung siehe Abb. 3.3) entlang eines traceroute Pfades verschickt, ist in den dann ankommenden *ICMP time exceed* Nachrichten der IP Header mit der aufgezeichneten Route enthalten. Durch den Vergleich der letzten aufgezeichneten Adresse mit der Quelladresse der ICMP Nachricht des vorigen *hops* lassen sich Aliase entdecken. Diese Methode ist in *iffinder* implementiert und lässt sich als Option zusammen mit einem *traceroute* zur Zieladresse einschalten.

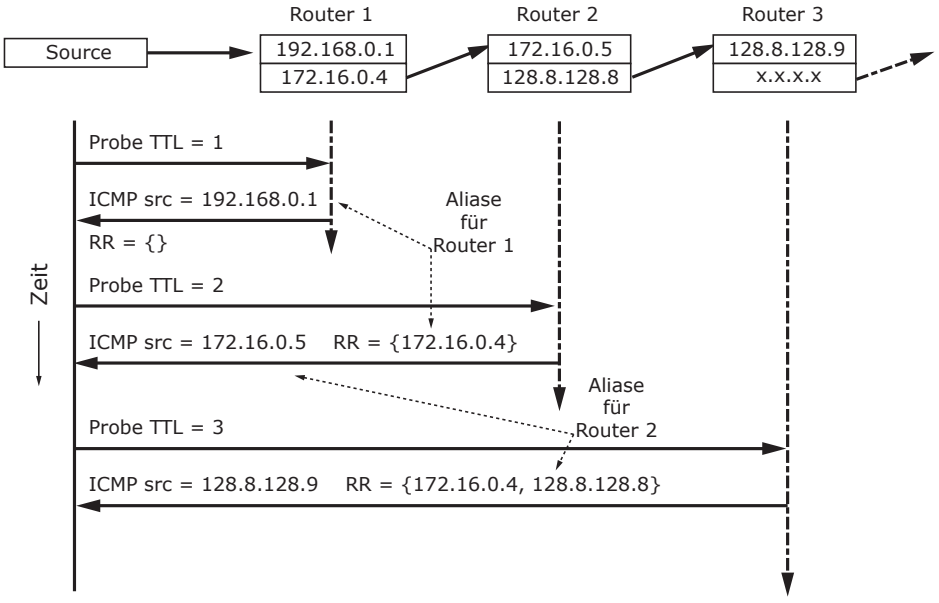


Abbildung 3.3: Traceroute mit gesetzter *record route* Option im IP Datagramm. Nach Quelle: [26]

3.5 Graph Based Method

Diese Methode, wie sie von Spring u. a. in [27] erwähnt wird, basiert auf zwei Vorraussetzungen. Die eine ist die Annahme von Punkt-zu-Punkt Verbindungen in Routernetzwerken und die andere ist das Fehlen von sogenannten *routing-loops*. Treffen diese Annahmen zu, dann gilt:

- Zwei Knoten die einen gemeinsamen Nachfolger (*common successor*) haben sind Aliase.
- Knoten innerhalb einer von *traceroute* aufgezeichneten Route können keine Aliase sein.

Diese Methode ist sehr gut geeignet, um aus Datensätzen mit *traceroute* Ausgaben geeignete Kandidatenpaare zum aktiven Testen mit *ally* zu extrahieren.

3.6 Analytical Alias Resolver Method

Dieser von Gunes und Sarac in [9] formulierte Ansatz geht von vier Voraussetzungen aus:

- Die Routerinterfaces sind über *point-to-point links* miteinander verbunden.
- Es sind jeweils die traceroute Ausgaben für Quelle → Ziel, sowie für Ziel → Quelle vorhanden.
- Die von traceroute ausgegebenen Routen sind symmetrisch.
- Die Provider nutzen für die direkt verbundenen Schnittstellen kleine Subnetze, also /30er- oder /31-Subnetze.

Sind diese Voraussetzungen erfüllt kann durch die Gegenüberstellung der Hinroute mit der umgekehrten Rückroute auf Aliase geschlossen werden (hierzu s. a. Abb. 3.4 und Abb. 3.5). In dem unten abgebildeten *traceroute* Resultat lässt sich eine Korrelation der IP Adressen von der zweiten bis zur zehnten Stelle der Tabelle ersehen. Dieses Ergebnis scheint die Annahme kleiner Subnetze zu bestätigen unter der Voraussetzung, dass es sich hier um Direktverbindungen handelt. Algorithmisch ist diese Methode über die graphentheoretische Formulierung des *Two-Path Alias Resolution Problem* (TPARP) mithilfe des *Analytical Alias Resolver* (AAR) lösbar, so beschrieben in [9].

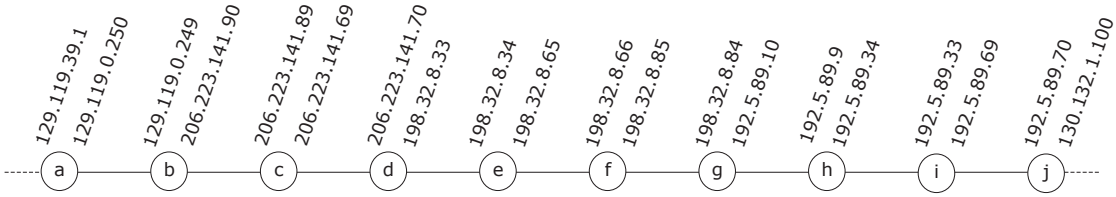


Abbildung 3.4: Aliase auf einem symmetrischen Pfad. Quelle: [9]

#	SMU to Yale (Direct Path)	Yale to SMU (Reverse Path)
1	129.119.39.1	129.119.223.249
2	**129.119.0.249**	**129.119.0.250**
3	**206.223.141.89**	**206.223.141.90**
4	**206.223.141.70**	**206.223.141.69**
5	**198.32.8.34**	**198.32.8.33**
6	**198.32.8.66**	**198.32.8.65**
7	**198.32.8.84**	**198.32.8.85**
8	**192.5.89.9**	**192.5.89.10**
9	**192.5.89.33**	**192.5.89.34**
10	**192.5.89.70**	**192.5.89.69**
11	130.132.1.19	130.132.1.100
12	130.132.252.244	130.132.23.1

Abbildung 3.5: Traceroute Ergebnisse zwischen SMU und Yale. Quelle: [9]

3.7 Durchgeführte IP Alias Resolution

Da die DIMES Datensätze bereits als Topologiedaten bezogen werden, also keine *traces* mit IP Adressen enthalten und auf den Datensätzen aus dem CAIDA ARK Projekt bereits IP Alias Resolution durchgeführt wurde, boten sich zunächst die Skitter Datensätze für eigene Versuche an. Zur Anwendung kamen die beiden Programme *ally* und *iffinder*, wobei das Erste als Übergabe Parameter eine Liste mit Alias Kandidaten (Textdatei mit zwei IPv4 Adressen per Zeile) und das zweite eine Liste mit IPv4 Adressen (Textdatei mit

einer Adresse per Zeile) erwartet. Hierbei zeigt sich schon der unterschiedliche Ansatz beider Programme, während *ally* dem Ansatz folgt Kandidatenpaare, die durch Anwendung der graphbasierten Methode ermittelt wurden zu bestätigen, versucht *iffinder* zu den gegebenen Adressen beliebige Aliase aufzudecken. Dazu bietet *iffinder* eine Option an, die es ermöglicht neu gefundene Adressen mit in die Testliste aufzunehmen. Dies ist besonders erfolgreich im Zusammenhang mit eingeschalteter *traceroute* und *record route* Option.
Für die Tests wurden zunächst aus den entsprechenden Datensätzen nach der graphbasierten Methode die IP Adressen von Alias Kandidaten extrahiert. Dabei fanden sich in allen Datensätzen IP Adressen aus den nicht routbaren, für die private Nutzung vorgesehenen Adressbereichen. Diese wurden, da nicht aktiv testbar, aus den Kandidatenlisten entfernt. Dann wurden die Listen in das für das jeweilige Programm entsprechende Format ausgegeben. Anschliessend wurde der erste Testlauf mit beiden Programmen gestartet. Hierbei zeigte sich, dass die Konzeption hinter *ally* für längerfristige Tests angelegt ist, denn während *iffinder* eine der Kandidatenlisten mit im Mittel 200.000 IP Adressen unter Auffindung von durchschnittlich 40.000 Aliasen in 10 bis 13 Stunden abarbeitete, hatte *ally* die erste Kandidatenliste nach sechs Tagen erst zur Hälfte getestet. Der Test wurde zu diesem Zeitpunkt abgebrochen, da ein vollständiger Testlauf mit *ally* den zeitlichen Rahmen dieser Arbeit gesprengt hätte. So wurden mit *iffinder* zuerst die Kandidatenliste abgearbeitet. Mit den gefundenen Aliasen wurden neue Graphen unter Verwendung einer Neuimplementierung des SkitterImport in BRITE erstellt. Dieses Modul SkitterAliasImport gibt am Ende die Summe der eliminierten Knoten und Kanten aus. Die Ausgaben finden sich in den Tabellen 3.1, 3.2 und 3.3. Da die Ergebnisse (siehe Tabelle 3.1) zunächst nicht überzeugten, wurden in einem zweiten Schritt aus allen CAIDA Datensätzen, die IP Adressen extrahiert und mit *iffinder* auf Aliase getestet. Hierbei konnten im Mittel doppelt so viele Aliase in den untersuchten Datensätzen gefunden werden (hierzu siehe Tabelle 3.2). Abschliessend wurde noch untersucht, wie stark sich die Topologie verändert, wenn die mittels graphbasierter Methode ermittelte Kandidatenliste als Aliasliste genutzt wird. Dabei liessen sich bis zu drei Viertel der Knoten eliminieren (s. a. Tabelle 3.3).

Datensatz			Anzahl		iffinder	
			Knoten	Kanten	Aliase	Kanten-reduktion
Skitter		12/07	207809	895682	3928	24786
		01/08	213974	927438	4168	25905
Ark	Team 1	12/07	174131	569132	–	–
		01/08	181155	598588	–	–
	Team 2	01/08	174448	498326	–	–
		02/08	178980	541300	–	–
		05/08	183781	585714	–	–
Dimes		12/07	387327	2529586	–	–
		01/08	376338	2394570	–	–

Tabelle 3.1: Die Ergebnisse der IP Alias Resolution auf den verschiedenen Datensätzen mit *iffinder* und Kandidatenliste. Vermerkt sind für die verschiedenen Ansätze jeweils die Anzahl eliminierter Knoten und Kanten des Graphen. Zu Anfang sind ursprüngliche Knoten- und Kantenanzahl festgehalten.

Datensatz			Anzahl		iffinder	
			Knoten	Kanten	Aliase	Kanten-reduktion
Skitter		12/07	207809	895682	5771	36967
		01/08	213974	927438	6170	39306
Ark	Team 1	12/07	174131	569132	3154	14426
		01/08	181155	598588	3454	16279
	Team 2	01/08	174448	498326	2076	7398
		02/08	178980	541300	2890	11187
		05/08	183781	585714	3222	13819
Dimes		12/07	387327	2529586	–	–
		01/08	376338	2394570	–	–

Tabelle 3.2: Die Ergebnisse der IP Alias Resolution auf den verschiedenen Datensätzen mit *iffinder* und allen IPv4 Adressen. Vermerkt sind für die verschiedenen Ansätze jeweils die Anzahl eliminierter Knoten und Kanten des Graphen. Zu Anfang sind ursprüngliche Knoten- und Kantenanzahl festgehalten.

Datensatz			Anzahl		graph based	
			Knoten	Kanten	Aliase	Kanten-reduktion
Skitter		12/07	207809	895682	154745	701682
		01/08	213974	927438	160388	733079
Ark	Team 1	12/07	174131	569132	66196	261146
		01/08	181155	598588	68312	276066
	Team 2	01/08	174448	498326	60278	203080
		02/08	178980	541300	62845	222515
		05/08	183781	585714	59606	226075
Dimes		12/07	387327	2529586	–	–
		01/08	376338	2394570	–	–

Tabelle 3.3: Die Ergebnisse der IP Alias Resolution auf den verschiedenen Datensätzen nach der *graph based method*. Vermerkt sind die Anzahl eliminierter Knoten und Kanten des Graphen. Zu Anfang sind ursprüngliche Knoten- und Kantenanzahl festgehalten.

3.8 Ergebnisse

Anhand der normalisierten Darstellungen der *joint degree distributions* (siehe Abb. 3.8 bis 3.14) ist der Erfolg der IP Alias Resolution gut absehbar. Hier ist mit zunehmender Aliaserkennung ein Verschwinden der schwarzen Fragmente für den Bereich von Knoten mit einer Kantenzahl über vierzig zu beobachten. Das Fehlen dieser Fragmente im DIMES Graph, lässt die Vermutung zu, dass dort bei der Erstellung der Topologiedatensätze auch mit einem oder mehreren Verfahren zur Alias Resolution gearbeitet wird. Weiterhin ist festzstellen, dass die Skitter Graphen durch die Alias Resolution einen leicht disassortativen Charakter erhalten. Da bei allen untersuchten Graphen der Assortationskoeffizient aber nahe bei Null liegt, was für eine zufällige Durchmengung von Verbindungen von Knoten mit geringer und hoher Kantenzahl spricht, ist dem aber keine grosse Bedeutung beizumessen. Auffällig war, dass auch in den Datensätzen des ARK-Projekts, trotz schon erfolgter Alias Resolution noch Aliase zu finden waren. Hierbei muss festgehalten werden, dass die aktiven Tests fünf bis sechs Monate nach Erstellung der Datensätze vorgenommen wurde, und sich in dieser Zeit die untersuchten Topologien im Bereich der IPv4 Adressen geändert haben können. Abschließend ist zu sagen, dass sorgfältige Alias Resolution ein zeitintensives Verfahren darstellt, in dessen Kontext der Aufbau und die Pflege einer Datenbank mit gefundenen Aliasen sinnvoll erscheint. Um dabei ein aktuelles Abbild tatsächlicher Routertopologien zu erhalten müssen die gefundenen Aliase kontinuierlich überprüft werden.

Datensatz			$\bar{x}$	σ_x^2	σ_x	r
Skitter		12/07	4.31012	60.69756	7.79086	-0.005898
		01/08	4.33434	62.09481	7.88002	-0.007880
Ark	Team 1	12/07	3.26841	31.68879	5.62928	0.021083
		01/08	3.30428	32.46484	5.69779	0.019359
	Team 2	01/08	2.85658	20.14108	4.48788	-0.025714
		02/08	3.02436	22.54271	4.74791	-0.016712
		05/08	3.18702	27.81054	5.27356	-0.001895
Dimes		12/07	6.53087	356.77677	18.88853	0.015118
		01/08	6.36281	311.77256	17.65708	0.057638

Tabelle 3.4: Die Eigenschaften der ungefilterten Topologien. Mit $\bar{x}$ = mittlere Kantenzahl (*mean degree*), σ_x^2 = Varianz, σ_x = Standardabweichung und r = Assortativitätskoeffizient.

Datensatz			$\bar{x}$	σ_x^2	σ_x	r
Skitter		12/07	4.27158	61.58812	7.84780	-0.006806
		01/08	4.29698	62.93979	7.93346	-0.009031
Ark	Team 1	12/07	–	–	–	–
		01/08	–	–	–	–
	Team 2	01/08	–	–	–	–
		02/08	–	–	–	–
		05/08	–	–	–	–

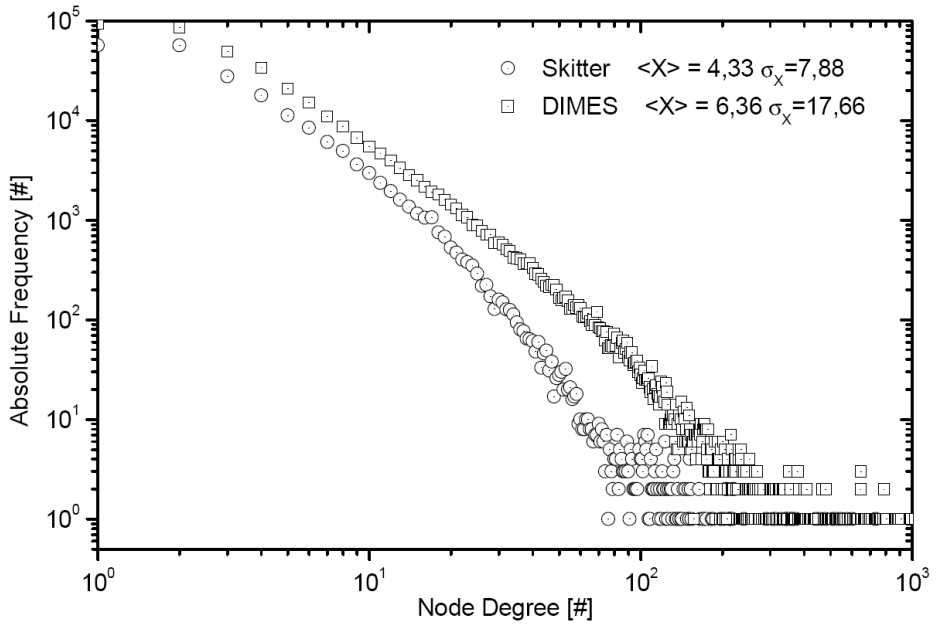
Tabelle 3.5: Die Eigenschaften der Topologien nach *iffinder*-basierter Alias Resolution mit Kandidatenliste. Mit $\bar{x}$ = mittlere Kantenzahl (*mean degree*), σ_x^2 = Varianz, σ_x = Standardabweichung und r = Assortativitätskoeffizient.

Datensatz			$\bar{x}$	σ_x^2	σ_x	r
Skitter		12/07	4.25042	61.83424	7.86347	-0.007365
		01/08	4.27402	63.20868	7.95038	-0.009489
Ark	Team 1	12/07	3.25603	32.39900	5.69201	0.019542
		01/08	3.277955	32.96763	5.74174	0.016246
	Team 2	01/08	2.84926	20.39292	4.51585	-0.026447
		02/08	3.01159	22.88291	4.78360	-0.017928
		05/08	3.17647	28.23682	5.31383	-0.002791

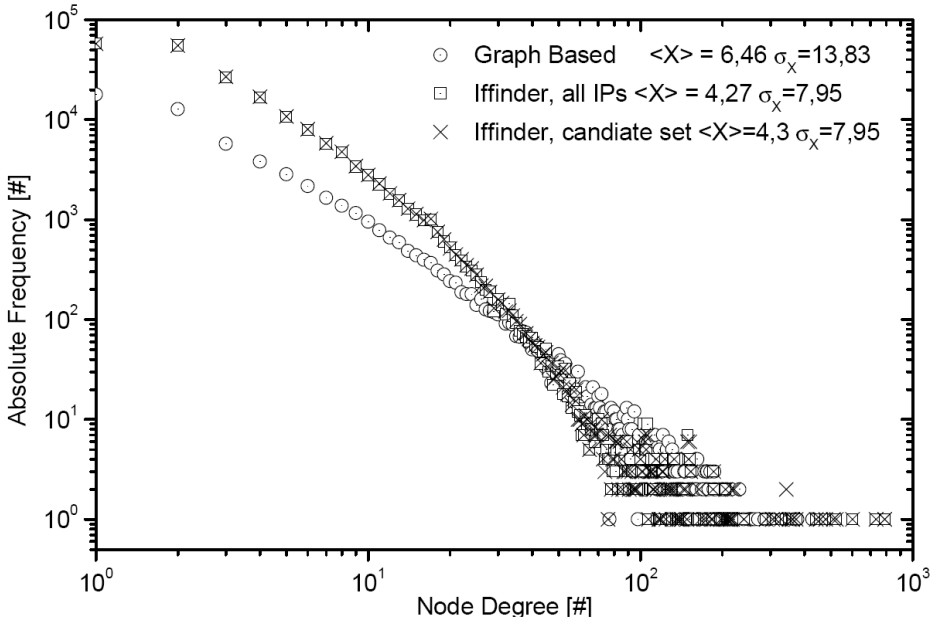
Tabelle 3.6: Die Eigenschaften der Topologien nach *iffinder*-basierter Alias Resolution mit allen IPv4 Adressen. Mit $\bar{x}$ = mittlere Kantenzahl (*mean degree*), σ_x^2 = Varianz, σ_x = Standardabweichung und r = Assortativitätskoeffizient.

Datensatz			$\bar{x}$	σ_x^2	σ_x	r
Skitter		12/07	3.65656	30.63431	5.53482	0.132519
		01/08	3.62757	30.17408	5.49309	0.121465
Ark	Team 1	12/07	2.85540	32.87927	5.73404	-0.013473
		01/08	2.85980	34.99400	5.91557	-0.016287
	Team 2	01/08	2.58782	23.76459	4.87489	-0.080559
		02/08	2.74666	29.68341	5.44824	-0.042837
		05/08	2.90948	34.17430	5.84587	-0.020608

Tabelle 3.7: Die Eigenschaften der Topologien nach *graph based* Alias Resolution. Mit $\bar{x}$ = mittlere Kantenzahl (*mean degree*), σ_x^2 = Varianz, σ_x = Standardabweichung und r = Assortativitätskoeffizient.

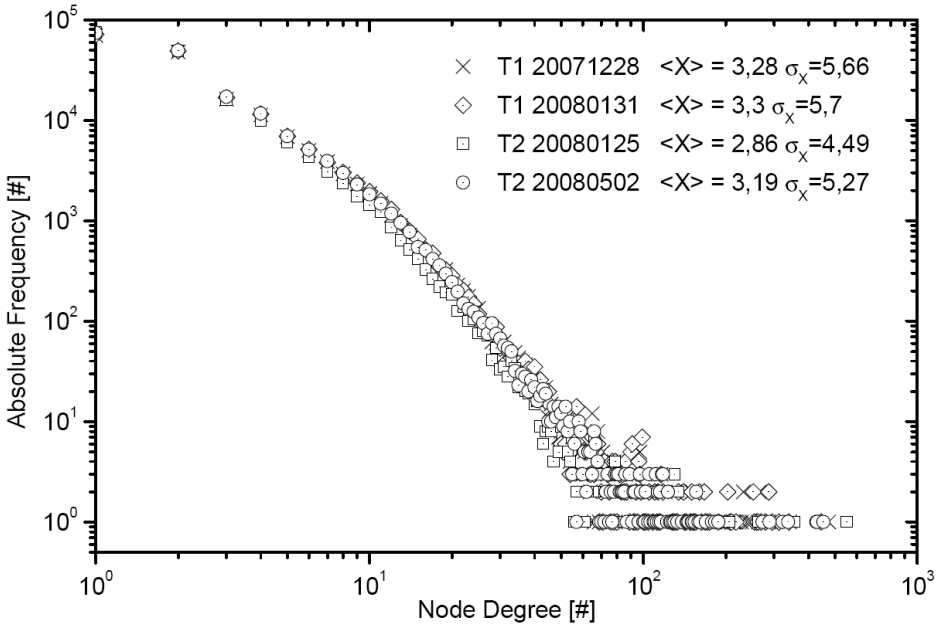


(a) DIMES und Skitter Graphen ungefiltert

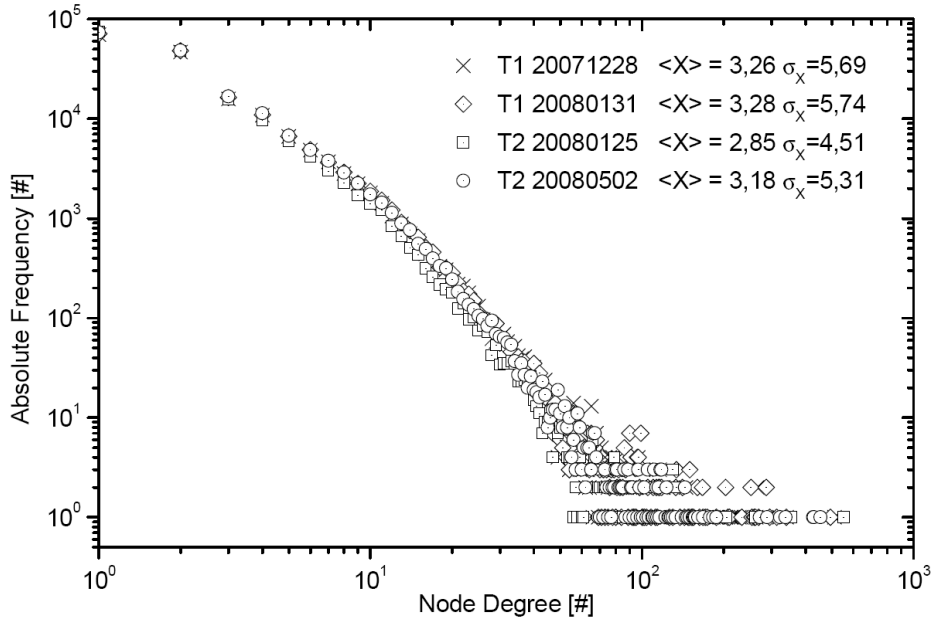


(b) DIMES ungefiltert und Skitter nach Alias Resolution

Abbildung 3.6: Die Degree Distributions als log-log-plot.



(a) ARK Graphen ungefiltert



(b) ARK Graphen nach Alias Resolution

Abbildung 3.7: Degree Distributions der ungefilterten und gefilterten Ark Graphen aus Team 1 und 2.

Joint Degree Distributions

Nachfolgend sind die Joint Degree Distributions der verschiedenen Datensätze abgebildet. Hierbei ist besonders interessant die Auswirkung der Alias Resolution nach der graphbasierten Methode auf den Skitter Graph im Vergleich zum DIMES Graph (siehe Abbildung 3.10). Zu den ARK Graphen aus Team 1 und Team 2 ist anzumerken, dass für den *degree* Wert zwischen 2 und 3 eine Auftretenshäufigkeit über den bisherigen Maximalwerten von 0.020 fest gestellt wurde, die in den Grafiken deshalb als weisser Fleck erscheint.

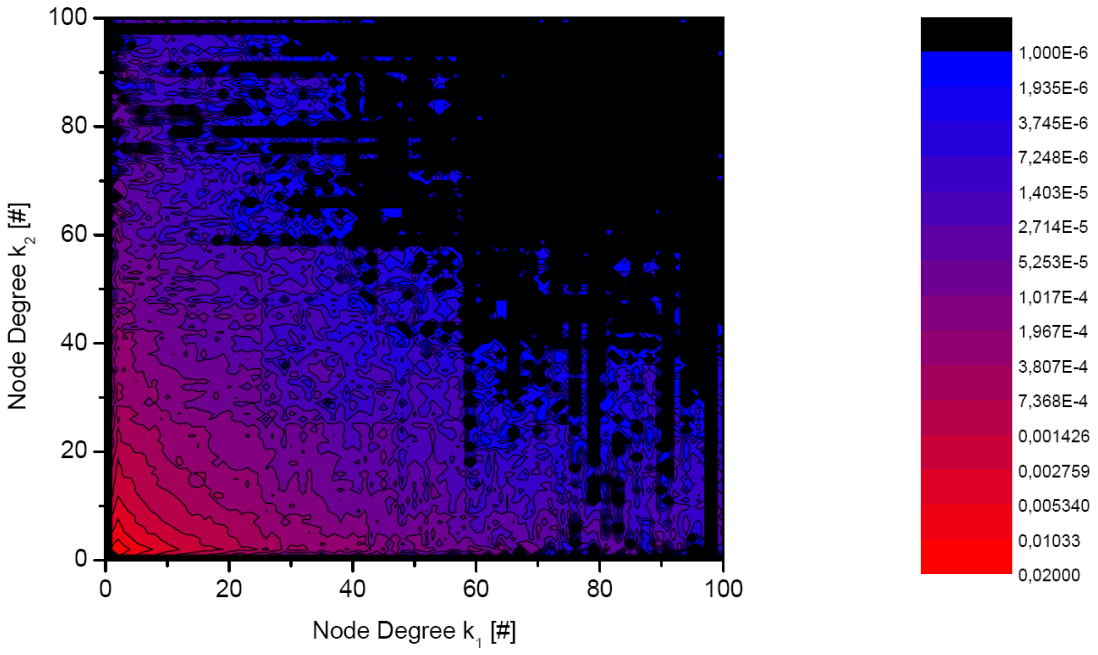
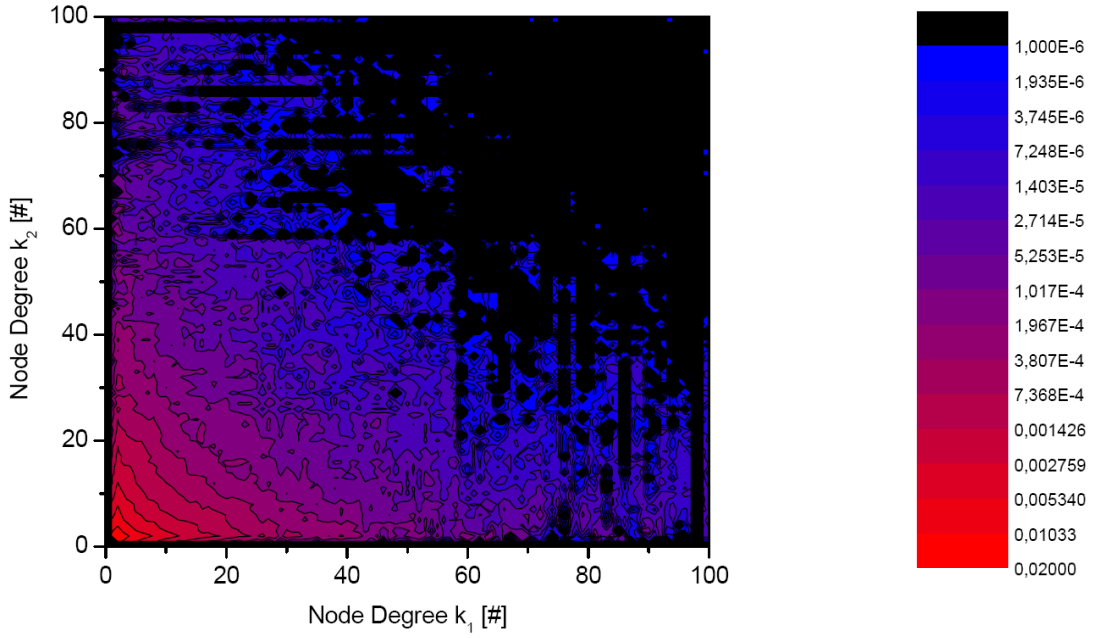
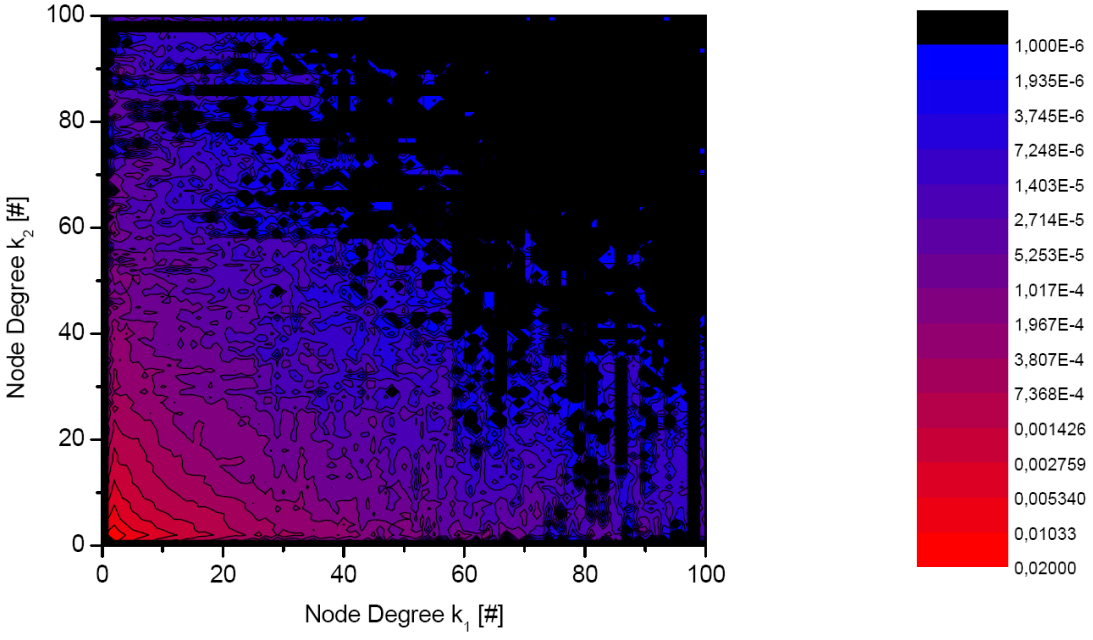


Abbildung 3.8: Die Joint Degree Distribution des ungefilterten Skitter Graphen

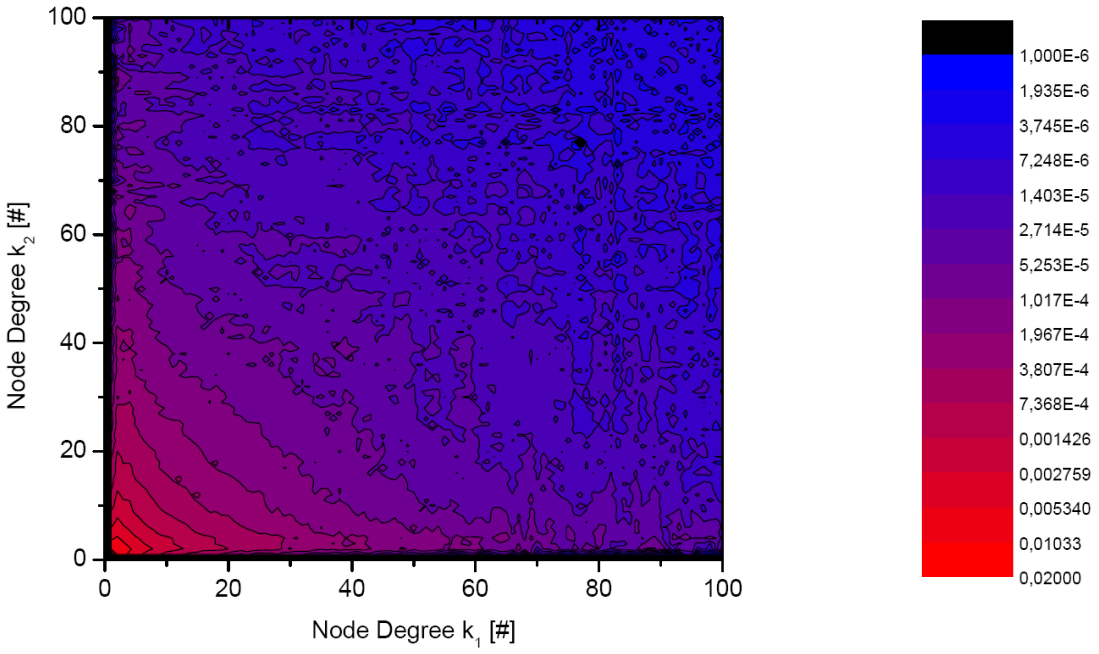


(a) Alias Resolution auf Basis der ermittelten Kandidatenliste des Datensatzes

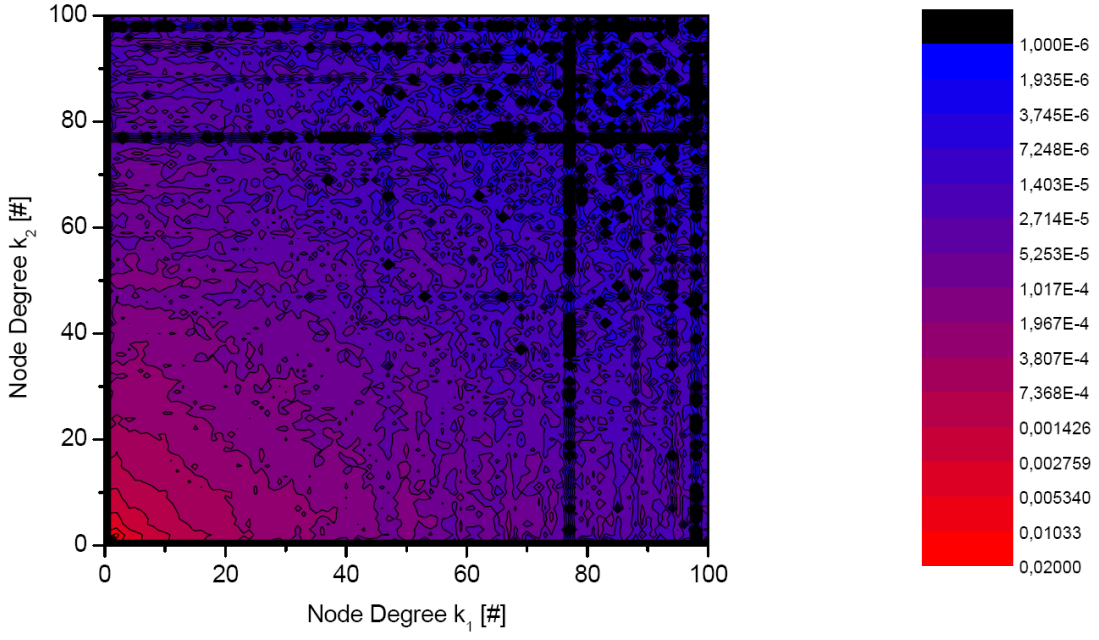


(b) Alias Resolution auf Basis aller innerhalb einer Woche ermittelten Aliase

Abbildung 3.9: Auswirkung der Alias Resolution mit iffinder auf die Joint Degree Distribution des Skitter Graphen.

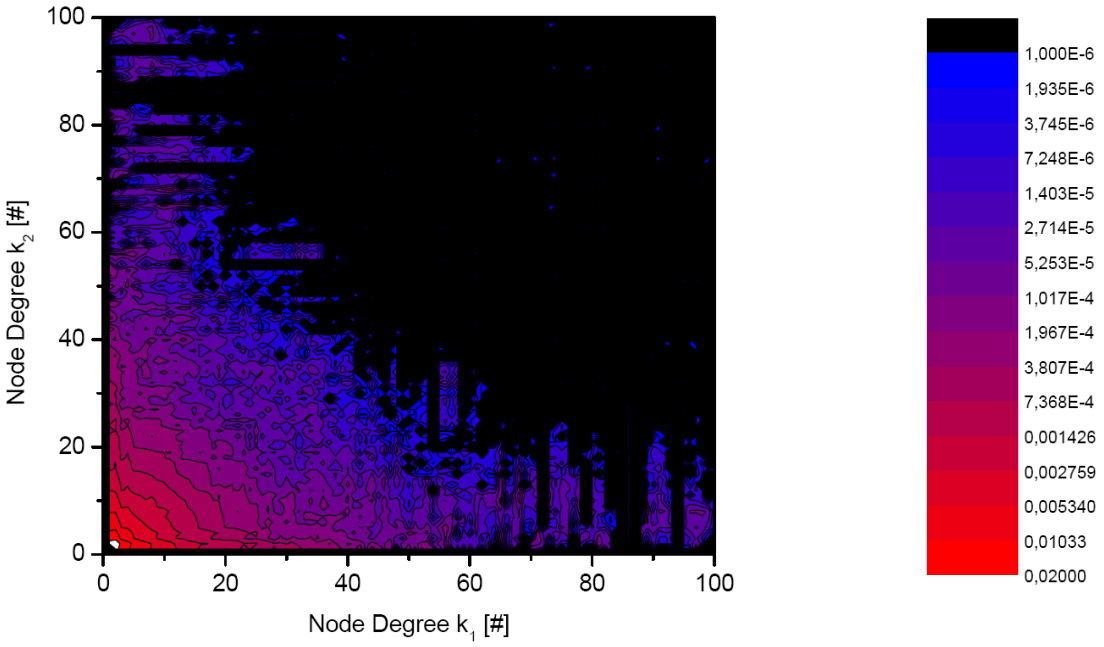


(a) DIMES ungefiltert

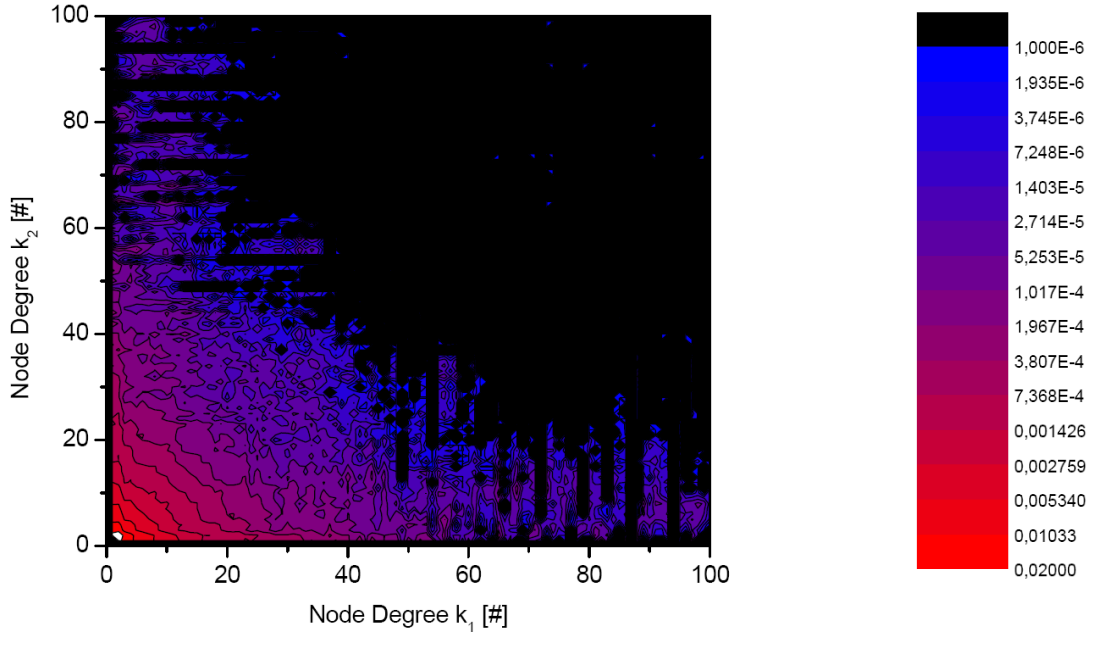


(b) Skitter nach Alias Resolution

Abbildung 3.10: Vergleich der Joint Degree Distributions des ungefilterten DIMES Graphen und des mittels graphbasierter Methode gefilterten Skitter Graphen

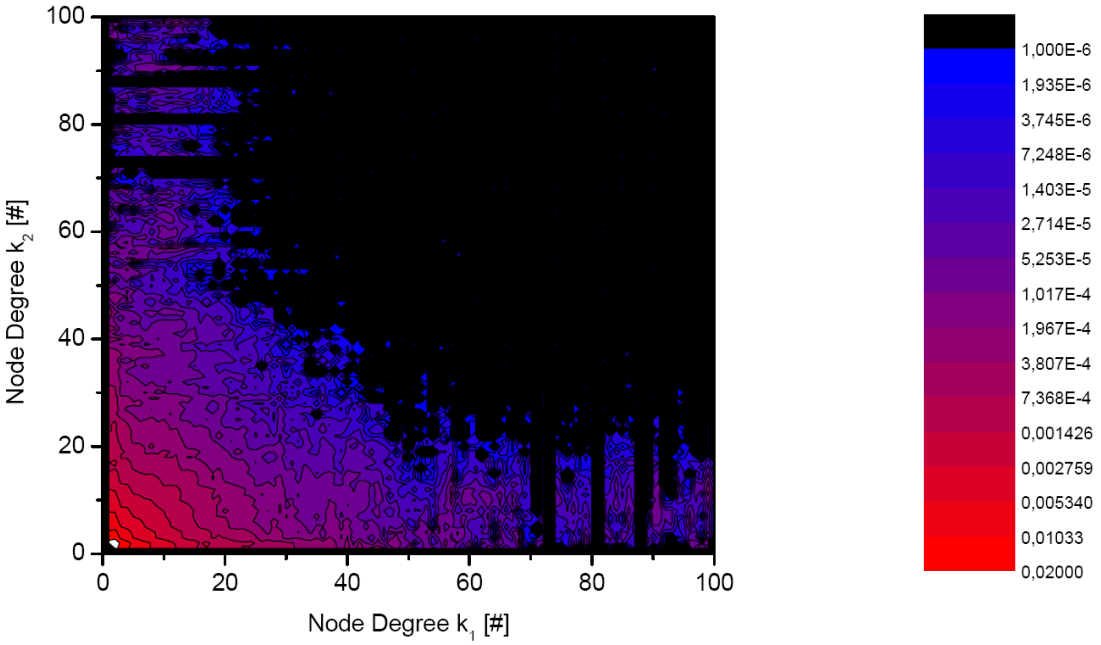


(a) ungefiltert

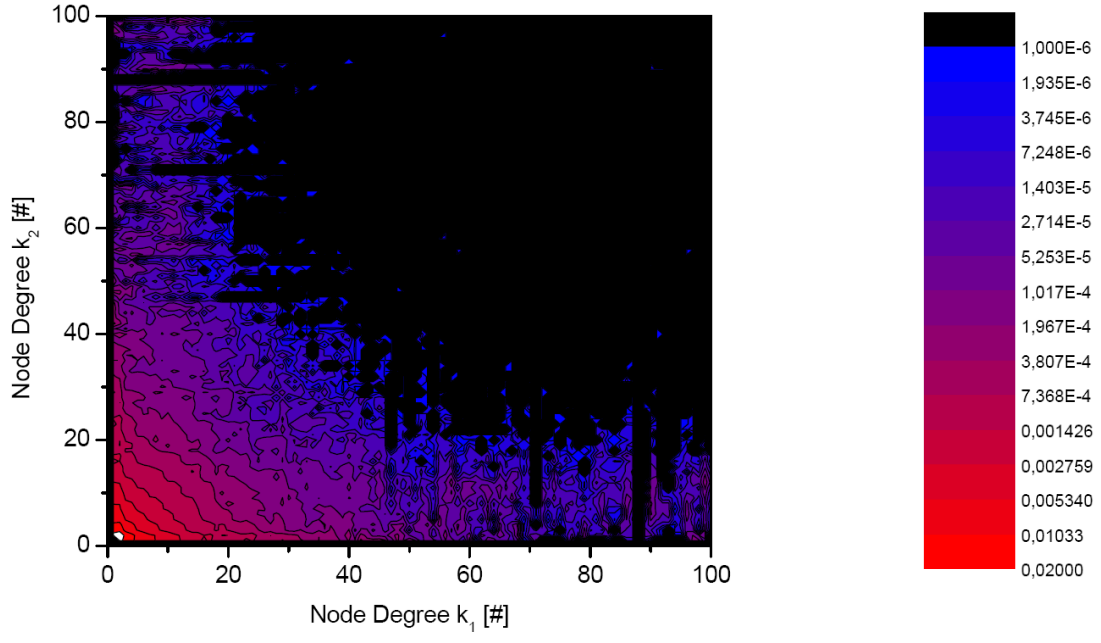


(b) nach Alias Resolution

Abbildung 3.11: Vergleich der Joint Degree Distributions der Graphen des Archipelago Projekts Team 1 vom Dezember 2007. Alias Resolution auf Basis aller innerhalb einer Woche ermittelten Aliase.

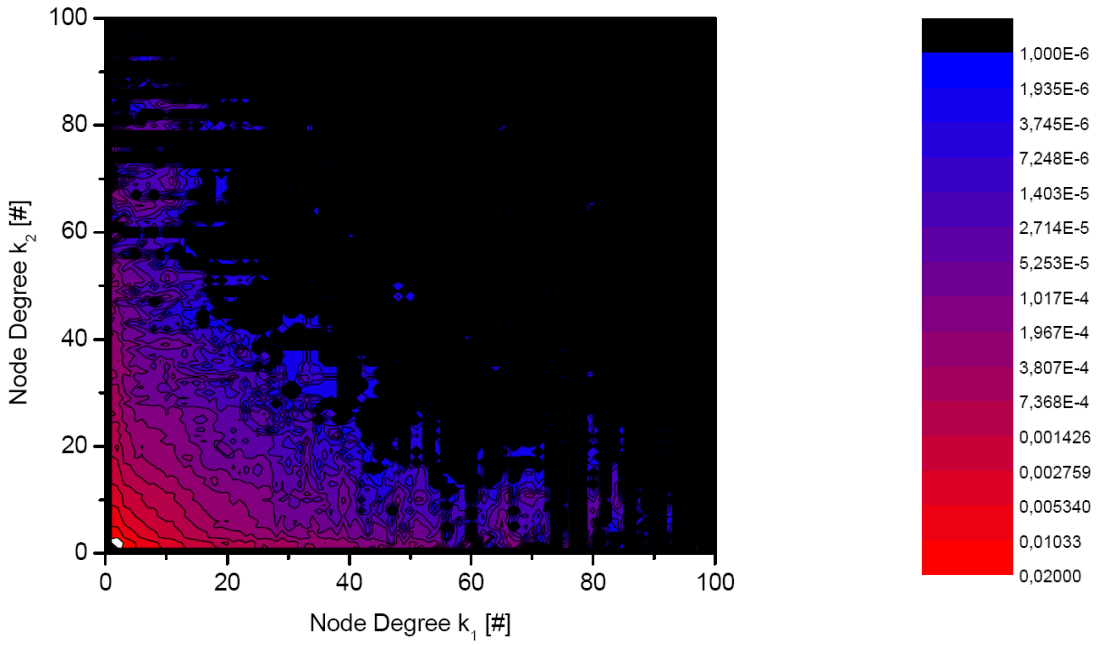


(a) ungefiltert

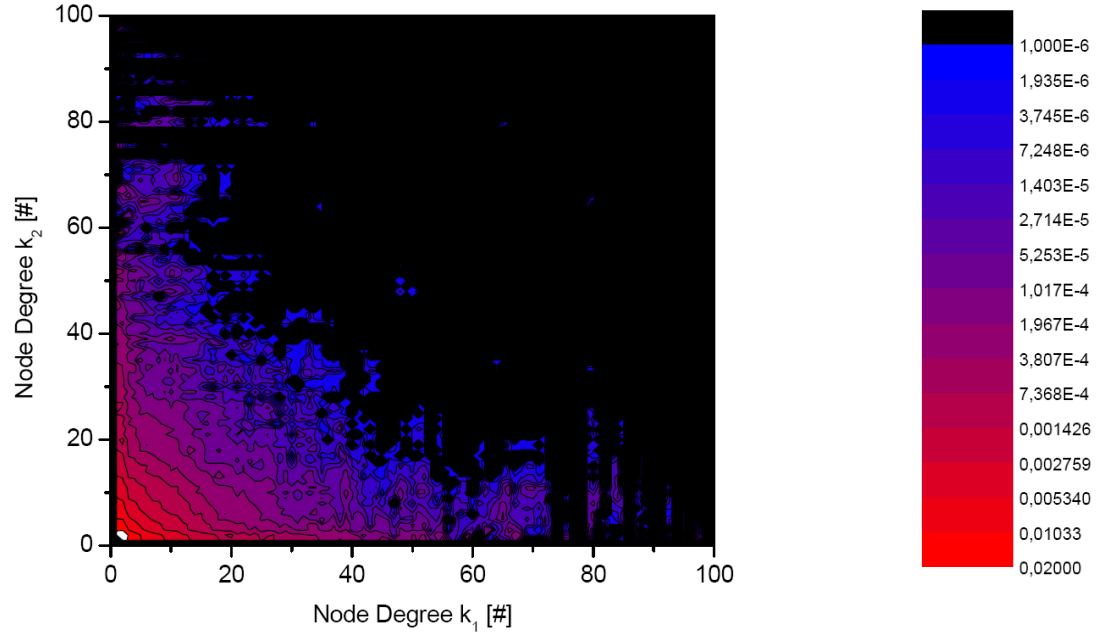


(b) nach Alias Resolution

Abbildung 3.12: Vergleich der Joint Degree Distributions der Graphen des Archipelago Projekts Team 1 vom Januar 2008. Alias Resolution auf Basis aller innerhalb einer Woche ermittelten Aliase.

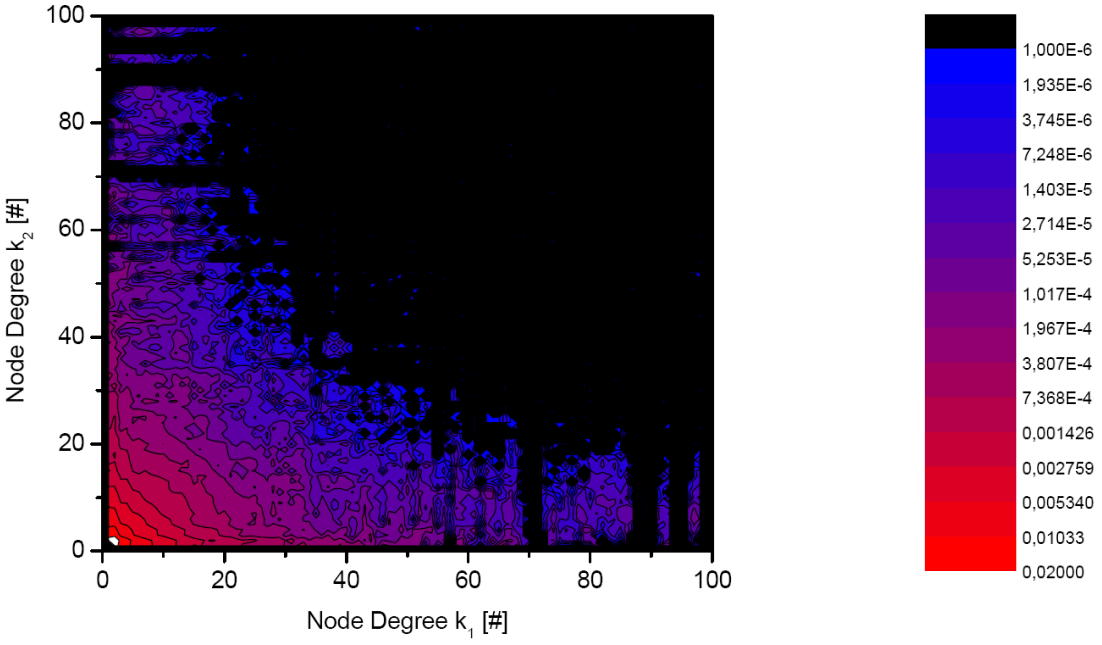


(a) ungefiltert

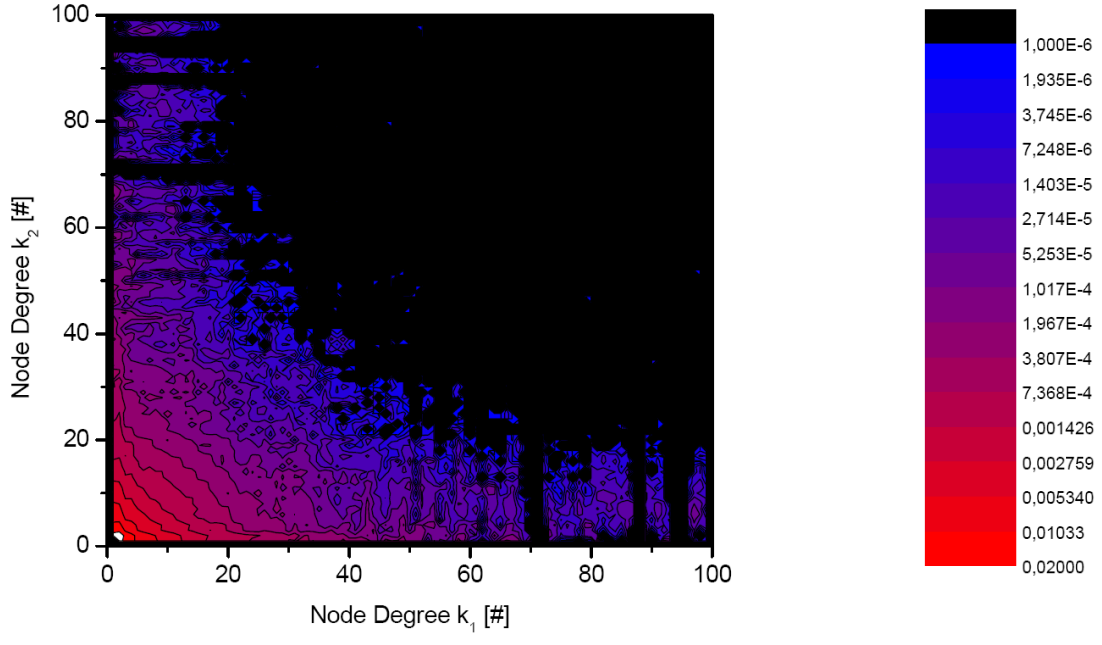


(b) nach Alias Resolution

Abbildung 3.13: Vergleich der Joint Degree Distributions der Graphen des Archipelago Projekts Team 2 vom Januar 2008. Alias Resolution auf Basis aller innerhalb einer Woche ermittelten Aliase.



(a) ungefiltert



(b) nach Alias Resolution

Abbildung 3.14: Vergleich der Joint Degree Distributions der Graphen des Archipelago Projekts Team 2 vom Januar 2008. Alias Resolution auf Basis aller innerhalb einer Woche ermittelten Aliase.

4 Strukturierte Overlay Netzwerke

Im Gegensatz zu unstrukturierten Overlay Netzwerken, die die zum Aufbau und zur Verwaltung des Netzes nötigen Informationen, entweder über eine zentrale Serverinstanz, über mehrere lokale Serverinstanzen, sogenannten *super peers* oder über *flooding search* von den anderen Peers beziehen, legen strukturierte Overlays diese Informationen in einer verteilten und geordneten Datenstruktur ab, der sogenannten *Distributed Hash Table (DHT)*. Dies bietet im Hinblick auf den unstrukturierten Ansatz Vorteile bezüglich Skalierbarkeit oder Robustheit des Netzwerkes, je nachdem ob mit zentralem Serveransatz, der einen *single point of failure* darstellt, dafür aber gut skaliert oder der *flooding search*, die zwar ein sehr fehlertolerantes Netzwerk aufbaut, aber nicht skaliert, verglichen wird (hierzu s. a. Tabelle 4.1). Im folgenden wird nun der Aufbau und die Funktionsweise einer DHT erläutert und das auf einer DHT basierende Overlay Pastry vorgestellt.

Verfahren	Routinginformationen per Knoten	lookup	Robustheit
zentraler Server	$\mathcal{O}(N)$	$\mathcal{O}(1)$	nein
flooding search	$\mathcal{O}(1)$	$\geq \mathcal{O}(N^2)$	ja
DHT	$\mathcal{O}(\log N)$	$\mathcal{O}(\log N)$	ja

Tabelle 4.1: Vergleich zwischen den Ansätzen zentraler Server, flooding search und distributed indexing, Quelle: [32].

4.1 DHT

Distributed Hash Tables stellen eine Umsetzung eines verteilten Indizierungssystems dar. Die verteilte Indizierung sowohl von Knoten- als Dateninformationen bietet gegenüber den unstrukturierten Peer-to-Peer Systemen eine Reihe von Vorteilen. Dies ist zum einen der durch die Einführung der DHT geschaffene einheitliche Adressraum (s. a. Abschnitt 4.1.1), der das Routing im Gegensatz zur *flooding search* effizienter gestaltet. Weiterhin ensteht innerhalb des Systems mehr Widerstandfähigkeit gegen Ausfälle einzelner Knoten dadurch, dass jeder DHT Knoten nur einen Teil der Informationen des Gesamtsystems verwaltet. Hierbei kann zusätzliche Robustheit erreicht werden, indem Repliken der DHT auf anderen Knoten gehalten werden und bei Verlust von Indexinformationen herangezogen werden.

Im Optimalfall liegen die Daten annähernd gleichverteilt im System vor und führen so zu einem *load balancing*. Nachfolgend werden die wesentlichen Eigenschaften einer DHT kurz beleuchtet.

4.1.1 Adressierung in DHT's

DHT's erzeugen einen Adressraum in den Daten und Knoten abgebildet werden. Als Adressen kommen typischerweise lange Ganzzahlen, z. B. im Bereich von 0 bis $2^{160} - 1$ zum Einsatz. Jedem Datum das in die DHT eingetragen wird, wird eine ID zugewiesen, die einer Adresse des Adressraums entspricht. Prinzipiell ist die Anwendung frei wie sie diese Adressen erzeugt, in tatsächlichen Realisierungen hat sich zu diesem Zweck aber die Verwendung von kollisionsfreien,sicheren Hashfunktionen[1] durchgesetzt. Dabei kann die Hashfunktion bei Daten auf den Dateinamen oder den Inhalt angewendet werden, und bei der Generierung von Knotenadressen auf die IP-Adresse (empfiehlt sich nicht, wenn mobile Knoten Zutritt zum System haben[2]) oder auf ausgesuchte Hardwareparameter (z. B. MAC-Adresse). Zusätzlich können zu den Daten noch Hashwerte von Schlüsselbegriffen gehalten werden, um bei der Suche nach einem Datum nicht den genauen Dateinamen kenne zu müssen.

4.1.2 DHT Interface

Nach Aussen stellt die DHT eine einfache Schnittstelle zur Verfügung, die sich im wesentlichen auf zwei Funktionen beschränkt. Diese entsprechen der Funktionalität einer herkömmlichen Hashtable, wie sie in den meisten Programmierspachen vorhanden ist. Einem *put*–Aufruf, der als Argumente einen Schlüssel und einen Wert übergeben bekommt und ein *true* oder *false* zurückliefert, je nachdem ob die Operation erfolgreich war und einem *get*–Aufruf der als Parameter einen Schlüssel übergeben bekommt und, falls vorhanden, den Wert dazu zurückliefert.

4.1.3 DHT Realisierung

Durch die einfache Schnittstelle lässt sich eine DHT auf verschiedene Arten realisieren. Einmal kann jeder Knoten innerhalb des Overlay ein Teil der DHT halten, hierbei bleibt

[1] Hierzu findet sich in der Literatur zum Thema oft der Hinweis auf den *Secure Hash Algorithm 1 (SHA-1)*, dieser gilt heute aber nicht mehr als sicher. Stattdessen sollte eine Funktion aus dem SHA-2 Paket oder RIPEMD-160 benutzt werden. Für weitergehende Informationen zum Thema siehe RFC 4270.

[2] In diesem Fall würde sich der Schlüssel des Knotens bei jedem Wechsel des Access Points durch die damit verbundene Änderung der IP Adresse ändern und alle über diesen Knoten im Overlay vorhandenen Routinginformationen würden ungültig.

die Kommunikation mit der DHT für den Overlay Knoten lokal. Ist die gesuchte Information lokal nicht vorhanden, kontaktiert der Overlay Knoten, je nach Algorithmus den für ihn nächsten Knoten mit einer Anfrage. Ein DHT Knoten kann seine Netzwerkschnittstelle aber auch über das Netzwerk publizieren und somit selbst für Rechner zugänglich machen, die nicht Teil des Overlay sind, dies ist z. B. sinnvoll, wenn das Overlay einen Infrastrukturdienst implementiert. Dann kann die DHT auch vollständig losgelöst von Overlay Netzwerken als Dienst z. B. auf besonders zuverlässigen Rechnern implementiert werden, seine Schnittstelle über eine Vielzahl von Protokollen implementieren und von beliebigen Rechnern genutzt werden. Ein Beispiel hierfür ist die auf einigen PlanetLab[3] Knoten laufende DHT-Implementierung OpenDHT[4], die einerseits von PlanetLab als Infrastrukturdienst genutzt wird, aber auch für Nutzung durch andere Projekte zur Verfügung steht.

4.2 Pastry

Pastry wurde 2001 von Rowstron und Druschel publiziert [25]. Ziel war der Entwurf eines vollständig dezentralen Peer-to-Peer Systems, mit effizientem Routing unter Einbeziehung von Netzwerklokalität. Im folgenden wird ein Überblick über die wichtigsten Eigenschaften des Pastry-Algorithmus gegeben, der sich an den Darstellungen des Originalpapiers [25] und der Zusammenfassung in [14] orientiert.

4.2.1 Adressraum

In Pastry erhalten Daten und Knoten Adressen mit einer Länge von l Bit, in [25] wählen die Autoren hier einen Wert von $l = 128$. Die Adressen sind Ganzahlen im Bereich 0 bis $2^l - 1$, werden aber für das Präfixbasierte Routing als Zeichenketten von Zahlen zur Basis 2^b, typischerweise mit einem Wert $b = 4$, interpretiert. Ein Knoten ist verantwortlich für alle Schlüssel zu denen er numerisch der nächste Knoten ist. Liegt die ID eines Datums exakt in der Mitte zwischen zwei Knoten, so sind beide für den Schlüssel verantwortlich. Ein Pastry Knoten unterhält und pflegt zur Gewährleistung seiner Funktionalität drei Datenstrukturen mit Adressen. Dies sind die *routing table*, das *leaf set*, sowie das *neighborhood set*. Die ersten beiden Datenstrukturen sind direkt für das Routing und die letztgenannte für die Auswahl der Nachbarschaft, indirekt aber auch für das Routing, von Bedeutung.

[3] siehe unter: `http://www.planet-lab.org/`
[4] siehe unter: `http://www.opendht.org/`

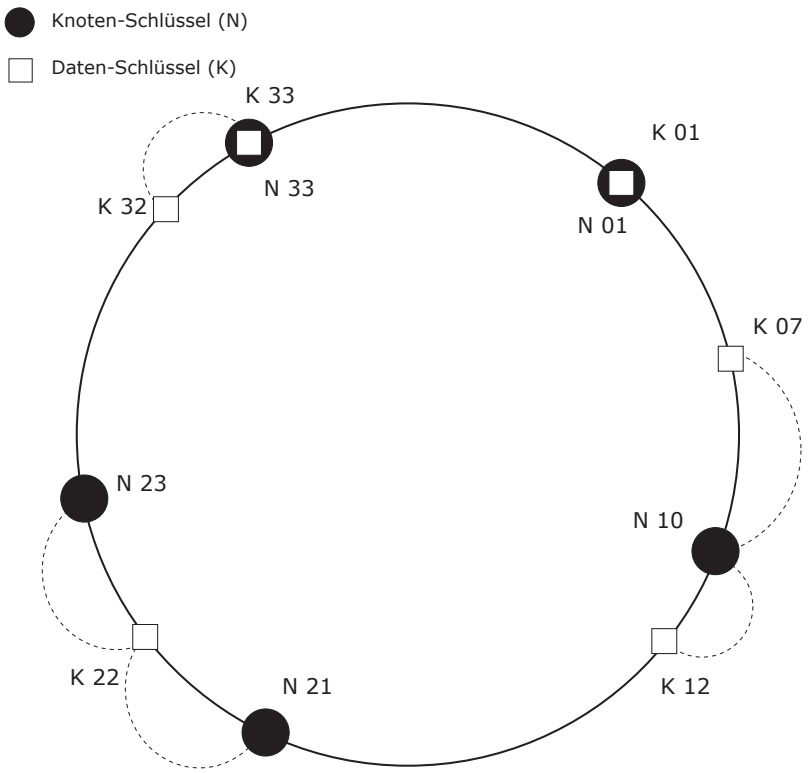


Abbildung 4.1: Schematische Darstellung der Zuständigkeiten von Knoten für die Verwaltung von Schlüsselinformationen in einem 4-Bit Adressraum. Quelle [14].

0	031120	1	201303	312201
1	0	110003	120132	132012
2	100221	101203	102303	3
3	0	103112	2	103302
4	0	103210	2	3
5	0	1	2	3

Tabelle 4.2: Routing Tabelle für einen angenommenen Knoten mit der ID 103220 innerhalb eines 12-Bit Adressraums zur Basis 4 ($l = 12,\ b = 2$). Quelle [14].

103123	103210	103302	103330

Tabelle 4.3: Leaf Set zum Knoten mit der ID 103220. Quelle [14].

031120	312201	120132	101203

Tabelle 4.4: Neighborhood Set zum Knoten mit der ID 103220. Quelle [14].

4.2.2 Routing Table

Die Routing Tabelle setzt sich aus $\frac{l}{b}$ Reihen und $2^b - 1$ Spalten zusammen. Hierbei enthält die *i-te* Reihe der Routing Tabelle auf dem Knoten *n* Knoten-ID's, die ein *i*–stelliges Präfix mit *n* gemeinsam haben. Somit stehen in Reihe 0 Knoten die kein Präfix mit *n* gemeinsam haben, und in der letzen Reihe Knoten deren ID sich von der des Knoten *n* nur in der letzten Stelle unterscheiden. Innerhalb einer Reihe sind die Knotenschlüssel numerisch aufsteigend angeordnet. Zu jedem Knotenschlüssel in der Routing Tabelle wird eine IP-Adresse gehalten. Findet sich für eine Reihe kein passender Eintrag bleibt sie leer.

4.2.3 Leaf Set

Um das *lookup* effizienter zu gestalten hält das Leaf Set die $|L|$ numerisch nächsten Schlüssel aus der Routing Tabelle, mit $\frac{|L|}{2}$ Werten $< n$ und $\frac{|L|}{2}$ Werten $> n$. Als typischer Wert wird von Rowstron und Druschel in [25] $|L| = 2^b$ oder $|L| = 2 \times 2^b$ genannt.

4.2.4 Neighborhood Set

Das Neighborhood Set hält die Schlüssel von Knoten, samt ihrer IP-Adresse, die sich an einer gegebenen Netzwerkmetrik gemessen am nächsten zum Knoten *n* befinden. Diese Metrik kann z. B. die Anzahl an hops (mit traceroute gemessen) oder der Wert für ein round trip delay sein. Auch besteht hier die Möglichkeit räumliche Nähe über einen geographischen Datensatz herzustellen. Die Knoten im Neighborhood Set werden in fixen Intervallen auf ihr vorhandensein im Overlay überprüft. Durch neu ins Overlay eintretende Knoten kann sich das Neighborhood Set verändern. Taucht dann im Neighborhood Set eine Knoten ID mit einer Präfixübereinstimmung von *i* Stellen zum Knoten *n* auf, so kann diese ID einen Eintrag in der Routing Tabelle ersetzen, wenn die Metrik für diese ID günstigere Werte liefert als für den Eintrag in der Routing Tabelle.

4.2.5 Routing

Das Routingverfahren ist mehrstufig organisiert. Zuerst überprüft der Knoten *n*, ob er selbst für den gesuchten Schlüssel *k* zuständig ist. Ist dies der Fall ist der Routing Prozess abgeschlossen. Andernfalls wird überprüft, ob *k* in den numerischen Bereich des Leaf Set fällt. Trifft dies zu wird die Query an den Knoten, der numerisch am nächsten zu *k* steht weitergeleitet. Sollte auch das nicht zutreffen, muss die Anfrage unter Nutzung der Routing Tabelle weitergeleitet werden. Hierbei wird zuerst ein Eintrag gesucht, der ein grösseres Präfix mit *k* gemeinsam hat als der Knoten *n*, an den im Falle eines Treffers die Anfrage weitergeleitet wird. Ist solch ein Eintrag in der Routing Tabelle nicht vorhanden wird die Query an einen Knoten weitergereicht, der die gleiche Präfix Übereinstimmung zu *k* hat wie der Knoten *n* aber numerisch näher an *k* liegt. Dieses Routingverfahren verhindert *routing loops*, da eine Query immer an einen numerisch näheren Knoten weitergeleitet wird und dadurch eine strenge Ordnungsrelation erreicht wird.

4.2.6 Selbstorganisation

Bevor ein Knoten in ein Pastry Overlay Netzwerk eintritt, wählt er sich selbst eine Knoten ID. Im allgemeinen wird hierzu der Hashwert der IP-Adresse oder des öffentlichen Schlüssel des Rechners benutzt. Damit der Vorgang des *bootstrapping*[5] erfolgreich sein kann wird vorausgesetzt, dass der Eintrittskandidat Kenntnis von einem, gemessen an der verwendeten Netzwerkmetrik, nahen Knoten hat. Im folgenden wird der neu hinzukommende Knoten mit n und der bekannte Nachbar mit k bezeichnet. Zuerst muss n nun seine Datenstrukturen Routing Tabelle, Leaf Set und Neighborhood Set initialisieren. Da k als benachbart vorausgesetzt wird, ist dessen Neighborhood Set eine erste gute Wahl für den Knoten n und wird also kopiert. Um nun die Routing Tabelle und das Leaf Set aufzubauen benötigt n Informationen von Knoten, die im Adressraum benachbart sind. Um diese zu erhalten routed n eine spezielle JOIN Message, die als Schlüssel die eigene ID enthält, via k zu dem Knoten, der numerisch am nächsten zu n ist. Dieser Knoten, im folgenden mit c benannt, antwortet auf die Anfrage, da er bisher für den Schlüssel von n zuständig war, mit seinem Leaf Set, welches durch die numerische Nähe im Adressraum wiederum ein gute erste Wahl für den Knoten n ist. Von jedem Knoten, der die JOIN Message auf dem Weg zu c weiterleitet bezieht der Knoten n eine Zeile für den Aufbau seiner Routing Tabelle, und zwar vom i–ten Knoten die i–te Zeile, angefangen bei Knoten k der die Zeile 0 liefert, da k zu n nur im Sinne der Netzwerkmetrik, aber nicht bezüglich des Adressraums nahe ist. Ist diese Prozedur abgeschlossen, sendet n an alle Knoten in der Routing Tabelle, dem Leaf Set und dem Neighborhood Set seine eigene Routing Tabelle. Dies führt bei den Knoten eventuell

[5] *bootstrapping* beschreibt im Kontext von Peer-to-Peer Systemen den Vorgang der Initialisierung eines Knotens. Dieser Vorgang ist abgeschlossen, sobald der Knoten Teil des Overlay geworden ist.

zur Anpassung ihrer eigenen Datenstrukturen. Ab diesem Moment ist ein Pastry Knoten vollständig im Overlay integriert.

5 Mobilität und Proximität

Die meisten Verfahren und Algorithmen, um ein Overlay aufzubauen, sind, ohne dies explizit zu benennen, für eine statische Topologie mit homogenen Endhostsystemen angedacht. Auch unter diesen Bedingungen unterliegen die Overlaynetze bei Aufkommen von *churn* schon starken Belastungen, die zur Fragmentierung oder Zerfall des Netzes führen können, zumindest aber seine Funtionsweise zeitweilig stark einschränken. Mit der Teilnahme von mobilen Knoten am Overlay kommt nun zum weiterhin möglichen herkömmlichen *churn*, eine neue Problemklasse hinzu, der durch die Mobilität verursachte *churn*, der sogenannte *mobility churn* [16]. Während mittlerweile viele Lösungsansätze existieren, um Peer-to-Peer Systeme robuster gegen den herkömmlichen *churn* zu machen, ist das Feld auf dem Gebiet mobiler DHT basierter Overlays eher dünn besiedelt. Hier existieren im wesentlichen zwei Ansätze, die im folgenden Abschnitt besprochen werden. Anschliessend folgt ein Abschnitt über die Proximitätsbeziehungen von Knoten in Overlay Netzwerken.

5.1 Mobilität

Eine Lösungsmöglichkeit der Problematik, die die Nutzung von DHT basierten Overlay Netzwerken in mobilen Umgebungen mit sich bringt, ist der von den Autoren Brampton u. a. zuerst in [3] formulierte Ansatz der StealthDHT. Dieser Ansatz unterscheidet in zuverlässige (*service nodes*) und unzuverlässige, mobile Knoten (*stealth nodes*). Den *service nodes* steht hierbei der volle Funktionsumfang der DHT zur Verfügung, während die *stealth nodes* weder Schlüssel speichern, noch Nachrichten weiterleiten. Somit spielen die *stealth nodes* für das Routing keine Rolle und die einzigen Knoteninformationen, die in der DHT gehalten werden sind die über *service nodes*. Erreicht wird diese Unterscheidung durch einen speziellen leichtgewichtigen Eintrittsmechanismus für die *stealth nodes*, der darauf verzichtet seine Ankunft im Overlay zu propagieren. Dies führt dazu, dass innerhalb des Overlays Nachrichten nur an *service nodes* gesendet und weitergeleitet werden können, also das Ziel jeder Anfrage nach einem Schlüssel einer *stealth node* eine *service node* ist. Somit können *stealth nodes* innerhalb des Overlay nicht direkt miteinander kommunizieren. Und auch *service nodes* kommunizieren mit *stealth nodes* nur zum Zwecke einer direkten Anfrage durch eine andere *stealth node*. Hierdurch wird erreicht, dass *stealth nodes* auf das Routing keinerlei Einfluss nehmen und somit durch auftretenden *mobility churn* keine Veränderungen in der DHT notwendig sind. Allerdings kann es dadurch, dass *stealth nodes*

nicht mit Routing Informationen versorgt werden bei diesen Knoten zu veralteten Routing Tabellen kommen. Für dieses Problem bieten die Autoren in [3] drei Lösungsmöglichkeiten an. Zum einen können Auffrischungen der Routing Informationen quasi huckepack mit den Antwort Nachrichten an die *stealth nodes* weitergereicht werden, diese können periodisch nach Routing Informationen pollen oder innerhalb definierter zeitlicher Intervalle führt jede *stealth node* eine erneute Join-Prozedur aus.
Dieser Ansatz weisst parallelen zu dem aus dem Bereich der unstrukturierten Netzwerke bekannten *super peers* auf und kann somit als hybride Variante angesehen werden. In [19] untersuchen die Autoren MacQuire u. a. die Laufzeiteigenschaften ihres StealthDHT Ansatzes im Vergleich zu einem Pastry Overlay. Hierzu wird ein Netzwerk mit 1000 Knoten simuliert, von denen 1% als *service nodes* vorgesehen sind und der Rest als *stealth nodes*. Im ersten Testlauf werden alle *stealth nodes* als stationäre Knoten betrieben. Der Vergleich zwischen Pastry und StealthDHT kommt hier zu gleichen Laufzeiteigenschaften. Sukzessiv werden nun die *stealth nodes* als mobile Knoten betrieben. Hier ergeben sich innerhalb der StealthDHT kaum Veränderungen, während Pastry mit Zunahme der mobilen Knoten schlechtere Werte vor allem für die Effizienz beim *lookup*, wie auch für die Anzahl der zur Reorganisation benötigten Nachrichten erreicht.

5.2 Proximität

Innerhalb von Overlay Netzwerken erhalten verschiedene Ebenen von Proximität Relevanz für die Effizienz des Routing und die Selbstorganisationsmechanismen des verwendeten Overlay Algorithmus. Hierbei sind zu unterscheiden die Proximität innerhalb des strukturierten Adressraums, also innerhalb des Overlay, sowie die Nähe innerhalb der darunter liegenden Topologie dem Underlay. Die Nutzung der Nachbarschaftzbeziehungen innerhalb der Overlay Topologie kann zu einer Minimierung der benötigten *hops* im Peer-to-Peer System führen, was damit natürlich auch eine Minimierung der *hops* im Routing in der darunterliegenden Netzwerktopologie mit sich bringt. Diese Form der Nachbarschaftsbeziehungen lässt sich wie in Pastry über ein Präfix gesteuertes Routing im Adressraum des Overlay nutzbar machen. Hierbei muss erwähnt werden, dass hierdurch in der Selbstorganisation und im Verwalten der Routing Tabellen jedes Knoten ein Overhead ensteht, da nicht einfach jeder Knoten als Nachbar genommen werden kann, sondern nur Knoten mit möglichst langen Präfixübereinstimmungen und diese erst gefunden werden müssen. Proximitätsbeziehungen in der Topologie des Underlay Netzwerks auszunutzen bringt wiederum eine Reduzierung der benötigten *hops* auf dem Weg zum Ziel einer Nachricht mit sich. Um diese auszunutzen muss eine geeignete Metrik gefunden werden. Hier ist es möglich als Basis die gemittelte *Round Trip Time* als Ausgabe des *ping* Programms, die Anzahl der benötigten *hops* als Ausgabe des *traceroute* Programms zu nutzen, oder Messungen der Bandbreite über das Versenden von Testpaketen anzustellen, eine Metrik die eine Kombination aus allen Werten nutzt und zu minimieren versucht (siehe hierzu [4]). Optimal für das Routing wäre es die

beiden Ansätze zu vereinen, so dass nahe im Schlüsselraum gleichzeitig bedeutet nahe innerhalb der genutzten Netzwerkmetrik. Dieses ließe sich innerhalb von Selbstorganisation nur mit einem Verwaltungsoverhead bewerkstelligen, der einerseits die Vorteile beim Routing wieder zunichte machen und andererseits das System mit einer erhöhten Empfindlichkeit gegenüber *churn* versehen würde. Ausserdem wäre diese Möglichkeit für den mobilen Einsatz nicht geeignet, da sich durch die Mobilität die Nähe innerhalb der Netzwerkmetrik permanent ändert.

6 Simulation

Das folgende Kapitel beschäftigt sich mit der Simulation des Pastry Overlay auf Basis der gewonnenen realen Internettopologien, die in den Kapiteln 2 und 3 beschrieben sind. Da diese Topologien zur Simulation zu umfangreich sind, mussten hieraus kleinere Teilnetze extrahiert werden. Ausserdem musste ein Simulationsmodell entwickelt werden, inklusive eines Modells der Bewegung der mobilen Knoten. Damit befasst sich der Abschnitt 6.2. Anschliessend folgt eine Beschreibung der durchgeführten Simulation. Das Kapitel schliesst mit einer Besprechung der Resultate.

6.1 OMNeT++/OverSim

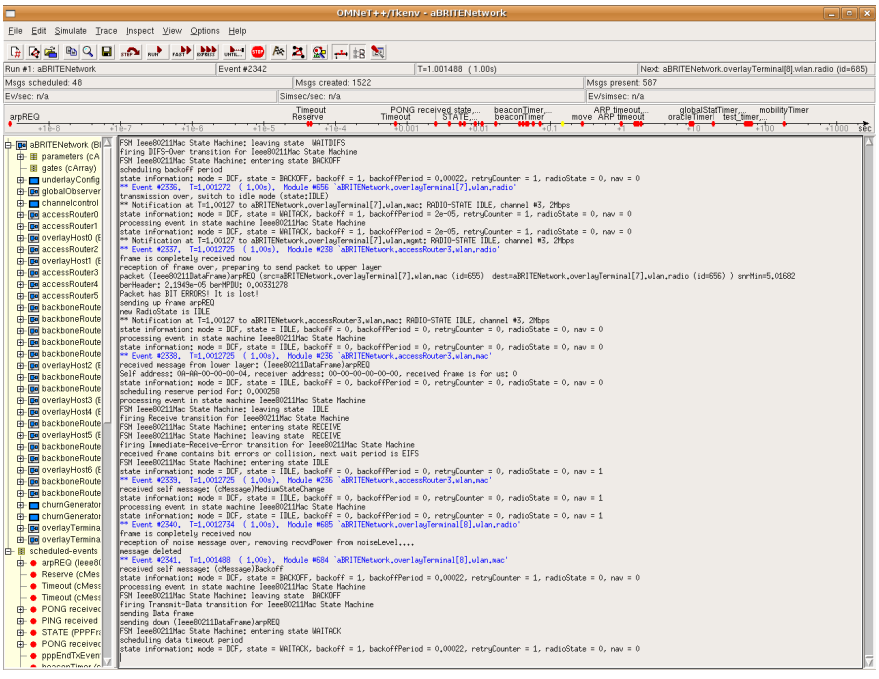


Abbildung 6.1: Darstellung des OMNeT++ Hauptfensters während einer Simulation.

Zur Simulation wurden drei aufeinander aufbauende Frameworks benutzt. OMNeT++ das den Simulationskernel und Basisklassen zum Netzwerkaufbau stellt. Das darauf aufsetzende INET-Framework, dass Implementierungen gängiger Netzwerkprotokolle, wie ARP, den IP-Layer mit den Application-Layern TCP und UDP zur Verfügung stellt, sowie Netzwerkinterfaces für kabelgebundene und kabelose Verbindungen. Dieses Framework wurde in einer vom darauf aufsetzende OverSim gepatchten Version benutzt. Und schliesslich das OverSim Paket mit der Implementierung der gängigsten Overlay Protokolle und darauf operierenden Anwendungen, wie einer DHT oder einer Anwendung zum Testen des Schlüsselbasierten Routing.

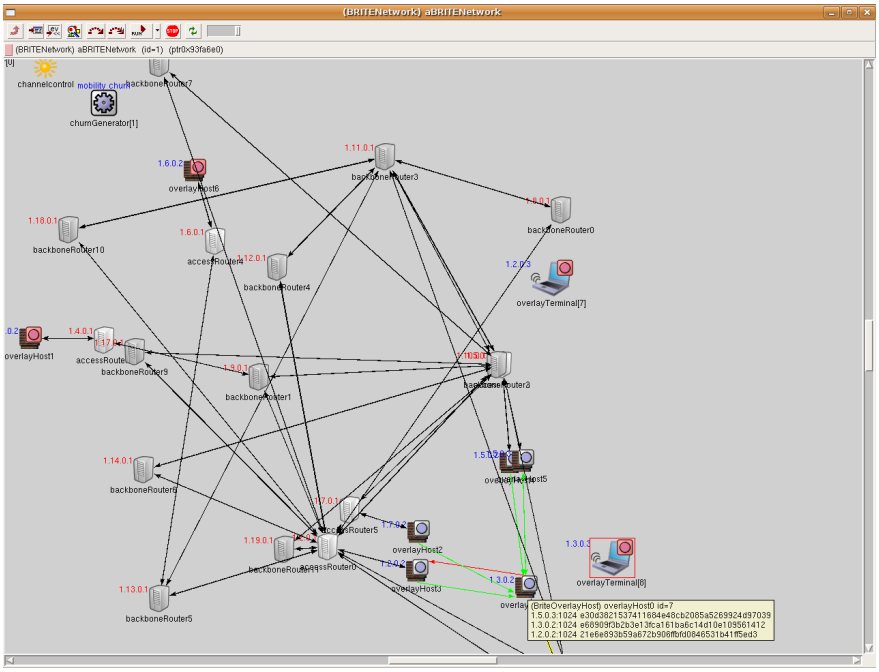


Abbildung 6.2: Darstellung der Simulation des Aufbaus eines kleinen Pastry Overlays auf Basis des mittels des Map Sampling Algorithmus auf 25 Knoten reduzierten Skitter Datensatz 12/07. Dieses Netzwerk diente lediglich als Test Overlay während Programmierphase.

6.2 Simulationsszenario

Auf den erzeugten Topologien soll ein realistisches IP basiertes Netzwerk mit Backbone Routern, Access Routern, sowie stationären und mobilen Endhostsystemen simuliert werden. Dazu wurden die stationären Systeme mit PPP Links verbunden. Die Access Router wurden zusätzlich als Access Points für die mobilen Systeme betrieben. Auf die Einrichtung von LANs zwischen den stationären Endhostsystemen und den Access Routern wurde

auf Grund des dadurch zu hohen Nachrichtenaufkommens in der Simulationsumgebung zu Gunsten einer höheren Effizienz verzichtet. Beim Einsatz von Ieee 802.11 Interfaces für die mobilen Knoten und die Access Points der Access Router ließ sich die Nutzung von Ethernet Frames allerdings nicht vermeiden. Hierdurch und durch die Beacon Frames zur Entdeckung der Nachbarschaft, sowie durch die ebenfalls über Nachrichten gesteuerte Mobilität wurde eine hohe Nachrichtenlast im System erzeugt. Für das in Abbildung 6.2 dargestellte Netz mit 12 Backbone Routern, 7 stationären Hostsystemen und 6 Access Routern ohne Access Points wurden im System durchschnittlich 300 Nachrichten, mit Access Points und 2 mobilen Knoten 1000 Nachrichten und mit 20 mobilen Knoten durchschnittlich 14000 Nachrichten pro simulierter Sekunde erzeugt. Dies dürfte es unmöglich machen realistische Netzwerkstrukturen mit mehr als 1000 Knoten und mehr als 100 mobilen Knoten auf einem Rechner zu simulieren.
Somit wurden aus den vorhandenen Topologien mit Hilfe des in BRITE implementierten Map Sampling Algorithmus nach Magoni und Pansiot in [20] Teilnetzwerke mit 1000 stationären Knoten extrahiert. Diese wurden dann mit einer Neuimplementierung (siehe hierzu Anhang A) des OmnetExport in BRITE in zur Simulation taugliche NED Netzwerkbeschreibungen umgewandelt.
Als Bewegungsmodell wurde das im INET Framework vorhandene *Random Waypoint* Modell ausgewählt, siehe hierzu auch [2]. Für den mobilen Knoten wurde mit Hilfe einer Normalverteilung eine Geschwindigkeit zwischen $1,12\frac{m}{s}$ und $1,67\frac{m}{s}$ bestimmt, dies entspricht mit 4-6$\frac{km}{h}$ der durchschnittlichen Geschwindigkeit eines Fussgängers. Die Rechnerknoten wurden auf einer virtuellen Fläche von $10\,km \times 10\,km$ angeordnet, innerhalb derer das Mobilitätsszenario ablaufen konnte.

6.3 Erweiterung des OverSim Sourcecodes

Dadurch, dass zur Simulation mit OverSim Netzwerke auf der Basis realer Topologien zum Einsatz kommen sollten, ergaben sich unerwartete Probleme in Hinsicht auf die Benutzung des OverSim Frameworks. Wie sich erwies ist OverSim allein für die Simulation mit zufällig generierten Topologien konzipiert, auch das in OverSim vorhandene Paket IPv4Underlay basiert auf zufällig generierten Topologien. Dies führte dazu, dass umfangreiche Anpassungen am OverSim Code notwendig waren und auf Teile des ungepatchten INET Framworks zurückgegriffen werden musste. Eine Auflistung der benötigten Änderungen und Neuimplementierungen findet sich in Anhang B. In diesem Zusammenhang war auch eine Neuimplementierung des OmnetExport aus dem BRITE Topologiegenerator vonnöten, da der vorhandene Export keine Unterscheidung zwischen verschiedenen Knotentypen traf, zum anderen weil er die Knoten fehlerhaft mit asymetrischen Links versah, was zu Fehlermeldungen in verschiedenen Netzwerkmodulen der OverSim Umgebung führte.

6.4 Ergebnisse

Bis zum Ende dieser Arbeit gelang es eine reale Internettopologie als Basis zur Simulation in die OMNeT++/OverSim Umgebung zu laden und zur Simulation zu nutzen. Allerdings nur mit zwei Einschränkungen. Der Handovermechanismus der WLAN Interfaces konnte nicht benutzt werden, um einen IP Adressenwechsel beim betroffenen mobilen Host herbeizuführen, was nach dem Wechsel des Access Points jeweils dazu führte, dass der mobile Host nicht mehr über das IP basierte Routing zu erreichen war und somit auch kein neuerlicher Eintritt in das Overlay möglich war. Ausserdem war es bei dem in 6.2 abgebildeten Testnetzwerk nicht möglich mehr als 4 mobile Knoten an Access Points zu binden. Ursache waren mit steigender Anzahl mobiler Knoten gehäuft auftretende Paketkollisionen der Air Frames. Hier wurde versucht Abhilfe zu schaffen, indem jedem Access Point ein eigener Kanal zugewiesen wurde und die mobilen Clients dann das ganze Spektrum versetzt scannten, dies führte zu keiner Verbesserung. In einem nächsten Schritt wurden die Konfigurationparameter für Sendeleistung, Antennenempfindlichkeit und der Frequenz sukkzessiv auf günstigere Werte gesetzt ohne Wirkung zu zeigen. In einem letzten Versuch wurden die Abstände zwischen den Access Points vergrössert und wieder Anpassungen der genannten Konfigurationparameter vorgenommen, was lediglich dazu führte, dass keiner der mobilen Knoten mehr einen Access Point zugewiesen bekam. Dies und die unerwarteten Probleme die sich durch die Nutzung realer Internettopologien in OverSim ergaben führten dazu, dass das Ziel dieser Arbeit, die Erhebung statistisch relevanter Daten über die Auswirkungen, des durch Mobilität verursachten *churn*, auf das Routing in einem Pastry Overlay nicht erreicht werden konnte.

7 Fazit

Diese Arbeit hat im ersten Teil gezeigt, dass die IP Alias Resolution ein geeignetes Mittel ist, um die durch *traceroute* erzeugten Schnittstellen zentrierten Topologien in Routertopologien umzuwandeln. Hierbei zeigte sich, dass im Hinblick auf die Gewinnung realer Routertopologien besonders die aktiven Testverfahren von Nutzen sind. Doch bleiben auch noch Fragen offen sind, wie die Aufdeckung von geswitchten Routernetzwerken oder die Entdeckung von ISP internen Routernetzwerken, die nicht routbare IP Adressen nutzen. Eine weitere interessante Frage, warum IP Alias Resolution nicht gleich mit in die Topologiedatenerhebung einfliesst, wie es beispielsweise das Programm *iffinder* mit eingebauter *traceroute* Funktionalität ermöglicht, versucht derzeit das von CAIDA initiierte *Archipelago* zu beantworten auf dessen zukünftigen Datensätze man gespannt sein darf.

Im zweiten Teil dieser Arbeit sollten die derart gewonnenen realen Internettopologien genutzt werden um darauf verschiedene Mobilitätsszenarien in einem Pastry Overlay zu testen und zu analysieren, welche Auswirkungen dies auf die DHT hat. Dies war innerhalb des gewählten Simulationsframework OverSim so jedoch nicht möglich und schlug deshalb fehl. Durch umfassendere Sondierung der Grenzen und Möglichkeiten im Vorfeld dieser Arbeit hätte sich dies möglicherweise vermeiden lassen. Denn auf der Grundlage des jetzigen Kenntnisstand erscheint es sinnvoller das angestrebte Simulationsszenario ohne Nutzung des OverSim Frameworks auf Basis des INET Paketes aufzubauen. Dazu wäre zwar eine Neuimplementierung des Pastry Algorithmus notwendig geworden, was aber im Nachhinein der vielversprechendere Ansatz gewesen wäre. Auch hätte auf den Einsatz realer Topologien zur Simulation verzichtet werden können, dies allerdings um den Preis nicht auf die realen Strukturen im Internet übertragbarer Ereignisse. Somit konnte diese Arbeit bis zum Abschluss lediglich den Aufbau der für das angedachte Simulationszenario nötigen Infrastruktur innerhalb der OverSim Umgebung leisten. Die Analyse des Mobilitätsszenario in strukturierten Overlay Netzwerken bleibt somit zukünftigen Arbeiten vorbehalten.

Literaturverzeichnis

[1] ALDERSON, David ; LI, Lun ; WILLINGER, Walter ; DOYLE, John C.: Understanding Internet Topology: Principles, Models, and Validation. In: *IEEE/ACM Trans. Netw.* 13 (2005), Nr. 6, S. 1205–1218

[2] BETTSTETTER, Christian: Mobility modeling in wireless networks: categorization, smooth movement, and border effects. In: *Mobile Computing and Communications Review* 5 (2001), Nr. 3, S. 55–66

[3] BRAMPTON, Andrew ; MACQUIRE, Andrew ; RAI, Idris A. ; RACE, Nicholas J. P. ; MATHY, Laurent: Stealth distributed hash table: unleashing the real potential of peer-to-peer. In: *CoNEXT '05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*. New York, NY, USA : ACM, 2005, S. 230–231. – ISBN 1-59593-197-X

[4] CASTRO, Miguel ; DRUSCHEL, Peter ; HU, Y. C. ; ROWSTRON, Antony: Exploiting network proximity in peer-to-peer overlay networks / Microsoft Research. URL `www.win.tue.nl/ipa/archive/springdays2002/location.pdf`, 2002 (MSR-TR-2002-82). – Forschungsbericht

[5] DONNET, Bennoit ; FRIEDMAN, Timur: Internet Topology Discovery: A Survey. In: *IEEE Communications Surveys and Tutorials* 9 (2007), Nr. 4

[6] DONNET, Benoit ; FRIEDMAN, Timur ; CROVELLA, Mark: Improved Algorithms for Network Topology Discovery. In: DOVROLIS, Constantinos (Hrsg.): *PAM* Bd. 3431, Springer, 2005, S. 149–162. – ISBN 3-540-25520-6

[7] DONNET, Benoit ; RAOULT, Philippe ; FRIEDMAN, Timur ; CROVELLA, Mark: Efficient algorithms for large-scale topology discovery. In: EAGER, Derek L. (Hrsg.) ; WILLIAMSON, Carey L. (Hrsg.) ; BORST, Sem C. (Hrsg.) ; LUI, John C. S. (Hrsg.): *SIGMETRICS*, ACM, 2005, S. 327–338. – ISBN 1-59593-022-1

[8] GOVINDAN, Ramesh ; TANGMUNARUNKIT, Hongsuda: Heuristics for Internet Map Discovery. In: *INFOCOM*, 2000, S. 1371–1380

[9] GUNES, Mehmet H. ; SARAC, Kamil: Analytical IP Alias Resolution. In: *IEEE ICC*, 2006, S. 459–464

[10] GUNES, Mehmet H. ; SARAC, Kamil: Resolving IP Aliases in Building Traceroute-Based Internet Maps. (2006), December. – URL `http://www.utd.edu/~mhg042000/papers/APAR_techRep.pdf`

[11] GUNES, Mehmet H. ; SARAC, Kamil: Importance of IP Alias Resolution in Sampling Internet Topologies. (2007), May. – URL http://www.utd.edu/~mhg042000/papers/GIS07.pdf

[12] GUNES, Mehmet H. ; NIELSEN, Nicolas S. ; SARAC, Kamil: Impact of Alias Resolution on Traceroute-Based Sample Network Topologies. (2007). – URL http://www.utd.edu/~mhg042000/papers/PAM07.pdf

[13] GUNES, Mehmet H. ; SARAÇ, Kamil: Inferring subnets in router-level topology collection studies. In: *Internet Measurement Comference*, 2007, S. 203–208

[14] GÖTZ, Stefan ; RIECHE, Simon ; WEHRLE, Klaus: Selected DHT Algorithms. In: STEINMETZ, Ralf (Hrsg.) ; WEHRLE, Klaus (Hrsg.): *Peer-to-Peer Systems and Applications*. Berlin : Springer-Verlag, June 2005 (LNCS 3485), S. 95–117. – ISBN 978-3-540-29192-3

[15] HADDADI, Hamed ; RIO, Miguel ; IANNACCONE, Gianluca ; MOORE, Andrew ; MORTIER, Richard: Network Topologies: Inference, Modeling, And Generation. In: *IEEE Communications Surveys* 10 (2008), Nr. 2, S. 48–69

[16] HSIAO, Hung-Chang ; KING, Chung-Ta: Mobility Churn in DHTs. In: *ICDCSW '05: Proceedings of the First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW'05)*. Washington, DC, USA : IEEE Computer Society, 2005, S. 799–805. – ISBN 0-7695-2328-5-08

[17] HYUN, Young ; HUFFAKER, Bradley ; ABEN, Emile ; LUCKIE, Matthew: *The CAIDA IPv4 Routed /24 Topology Dataset*. – URL http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml

[18] KARAPURKAR, Alap ; OGALE, Neeti ; SHAHAMAT, Layla: *CMSC711 Project: Alias Resolution*. – Students Project at the Univ. of Maryland, Dept. of Computer Science

[19] MACQUIRE, Andrew ; BRAMPTON, Andrew ; RAI, Idris A. ; MATHY, Laurent: Performance Analysis of Stealth DHT with Mobile Nodes. In: *PERCOMW '06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*. Washington, DC, USA : IEEE Computer Society, 2006, S. 184. – ISBN 0-7695-2520-2

[20] MAGONI, Damien ; PANSIOT, Jean-Jacques: Internet Topology Modeler Based on Map Sampling. In: *ISCC '02: Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*. Washington, DC, USA : IEEE Computer Society, 2002, S. 1021. – ISBN 0-7695-1671-8

[21] NEWMAN, M. E. J.: Assortative Mixing in Networks. In: *Physical Review Letters* 89 (2002), October, Nr. 20. – URL http://dx.doi.org/10.1103/PhysRevLett.89.208701

[22] NEWMAN, M. E. J.: The Structure and Function of Complex Networks. In: *SIAM*

Review 45 (2003), Nr. 2, S. 167–256. – URL `http://link.aip.org/link/?SIR/45/167/1`

[23] PANSIOT, Jean-Jacques ; GRAD, Dominique: On Routes and Multicast Trees in the Internet. In: *SIGCOMM Comput. Commun. Rev.* 28 (1998), Nr. 1, S. 41–50. – ISSN 0146-4833

[24] RATNASAMY, Sylvia ; FRANCIS, Paul ; HANDLEY, Mark ; KARP, Richard ; SCHENKER, Scott: A Scalable Content-Addressable Network. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA : ACM, 2001, S. 161–172

[25] ROWSTRON, Antony I. T. ; DRUSCHEL, Peter: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: *Middleware*, 2001, S. 329–350

[26] SHERWOOD, Rob ; SPRING, Neil: Touring the internet in a TCP sidecar. In: *Internet Measurement Conference*, 2006, S. 339–344

[27] SPRING, N. ; DONTCHEVA, M. ; RODRIG, M. ; WETHERALL, D.: How to resolve IP aliases / Washington Univ. Computer Sci. URL `http://www.cs.washington.edu/homes/rodrig/pubs/alias_res.pdf`, April 2004 (04-05-04). – Forschungsbericht

[28] SPRING, Neil T. ; MAHAJAN, Ratul ; WETHERALL, David: Measuring ISP Topologies with Rocketfuel. In: *SIGCOMM*, 2002, S. 133–145

[29] STOICA, Ion ; MORRIS, Robert ; KARGER, David ; KAASHOEK, M. F. ; BALAKRISHNAN, Hari: Chord: A scalable peer-to-peer lookup service for internet applications. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA : ACM, 2001, S. 149–160. – ISBN 1-58113-411-8

[30] TEIXEIRA, Renata ; MARZULLO, Keith ; SAVAGE, Stefan ; VOELKER, Geoffrey M.: In search of path diversity in ISP networks. In: *Internet Measurement Comference*, 2003, S. 313–318

[31] WATTS, D. J. ; STROGATZ, S. H.: Collective dynamics of 'small-world' networks. In: *Nature* 393 (1998), Juni, S. 440–442

[32] WEHRLE, Klaus ; GÖTZ, Stefan ; RIECHE, Simon: Distributed Hash Tables. In: STEINMETZ, Ralf (Hrsg.) ; WEHRLE, Klaus (Hrsg.): *Peer-to-Peer Systems and Applications*. Berlin : Springer-Verlag, June 2005 (LNCS 3485), S. 79–93. – ISBN 978-3-540-29192-3

A Erweiterungen zum BRITE Topologie Generator

Der Brite Topologie Generator wurde, um die Ergebnisse der IP Alias Resolution in den Netzwerkgraph einfliessen zu lassen um ein Import Modul erweitert. Dieses Modul erwartet als Übergabeparameter eine Textdatei mit den zu nutzenden Aliaspaaren. Die Aliaspaare führen im Importmodul dann zur Zusammenlegung der Aliase zu einem Knoten. Existieren Kanten zu einem dritten Knoten doppelt werden diese ebenfalls zusammengelegt und der *delay* Wert der neuen Kante als Mittelwert der beiden ursprünglichen Kanten berechnet. Anschliessend wird ausgegeben wieviele Aliase gefunden wurden und wieviele Kanten zusammengelegt wurden.

Für das OverSim Framework wurde ein neues Export Modul implementiert, da sich das der vorhandene Export für die OMNeT++ Simulationsumgebung für die gestellten Anforderungen als untauglich erwies. Dies zum einen, da keine Unterscheidung zwischen den benötigten Knotentypen Backbone Router, Access Router und stationärer Host getroffen wurde, zum anderen, da asymmetrische Links erzeugt werden mit denen keines der Netzwerkinterfaces aus dem INET Framework funktionierte. Das neue Export Modul lässt nun über eine *Properties* Datei umfangreiche Konfigurationseinstellungen für das zu generierende NED Netzwerk zu. Zu diesem Zweck ist als Zusatz noch ein Parser implementiert worden. Diese Erweiterungen finden sich auf der beigefügten DVD.

B Erweiterungen zum OverSim Framework

Das OverSim Framework stellt zwei verschiedene Underlay Modelle zur Verfügung. Zum einen ist dies das IPv4Underlay, das einen vereinfachten IP Layer integriert und eine Unterscheidung zwischen den Knotentypen OverlayHost, BackboneRouter, AccessRouter, OverlayBackboneRouter und OverlayAccessRouter vornimmt. Hiermit lassen sich jedoch nur zufällig generierte Topologien nutzen. Das zweite Underlay Modell ist das SimpleUnderlay mit dem sich der Topologieaufbau zwar über eine aus einem CAIDA-Topologiedatensatz gewonnene XML-Datei mit den Koordinaten der ermittelten Router steuern lässt, dass aber keinen IP Layer nutzt und nur einen Knotentyp unterstüzt. Beide Modelle sind für die Nutzung mobiler Knoten mit dem WLAN Interface ungeeignet, da der OverSim Patch für das INET Framework auf die Nutzung von ARP verzichtet, um das durch ARP Requests erhöhte Nachrichtenaufkommen, zu Gunsten der Möglichkeit mit höheren Knotenzahlen zu simulieren, zu minimieren Dies führte dazu, dass an Stelle der gepatchten Versionen des ARP und des IPv4 Paketes und der im Hosts Paket befindlichen NetworkLayer.ned wieder die Originaldateien aus dem INET Framework benutzt wurden. Auf dieser Basis wurde entschieden ausgehend vom IPv4Underlay Paket ein eigenes Model für ein Underlay zu entwickeln. Hierbei wurde auf Routermodule verzichtet, die am Overlay teilnehmen können, so dass an Hostmodulen ein Backbone Router, ein Access Router mit Access Point, ein stationärer Host und ein mobiler Host zu entwickeln waren. Darüber hinaus war für den Access Point eine Neuimplementierung des Ieee 802.11 Interfaces notwendig, da kein LAN verwendet wurde und somit keine Ethernetinterfaces vorhanden waren auf die die Ausgaben des Management Moduls der WLAN Karte mittels Relay Unit gebridget hätten werden können. Weiterhin war das AccessNet aus dem IPv4Underlay Paket neu zu entwickeln, da nun zwei verschiedene Knotentypen Anschluss an den Acces Router finden sollten. Auch der UnderlayConfigurator aus dem IPv4Underlay Paket war neu zu implementieren, da eine reale Topologie geladen werden sollte. Hierbei zeigte sich eine weitere konzeptuelle Einschränkung des OverSim Frameworks. Der eigene UnderlayConfigurator musste vom Basis UnderlayConfigurator im Common Paket abgeleitet werden um innerhalb der OverSim Umgebung lauffähig zu sein. Dies brachte mit sich, dass für jeden Knotentypen der am Overlay teilnehmen können soll ein ChurnGenerator eingebunden werden musste der die Initialisierung der Overlay Knoten übernimmt. Dies ist direkte Folge der Auslegung auf die Simulation mit zufällig generierten Topologien. Da kein herkömmlicher churn getestet

werden sollte konnten die angebotenen Churn Modelle nicht benutzt werden. So wurde für die stationären Overlay Hosts ein DummyChurn Modell entwickelt, das in der Initialisierungsphase lediglich die stationären Knoten durch Aufruf der entsprechenden Methode im UnderlayConfigurator zur Topologie hinzufügt. Da die selbst implementierten Churn Generatoren wiederum von einer Basis Klasse abgeleitet werden müssen, die selbst nur über ein Handle auf *Common/UnderlayConfigurator.h* verfügt müssen alle über diesen Mechanismus gewünschten Methodenaufrufe dort als echt virtuelle Methode deklariert werden. Dies erfordert jeweils wieder ein Neuübersetzen des OverSim Frameworks. Für den mobilen Knoten bot sich diese Form der Implementierung geradezu an und es wurde hierfür ein MobilityChurn Modul entwickelt, das auch die Migration in andere Access Netze unterstüzt. Ein weiteres Problem während der Initialisierungsphase war das Management Modul des Ieee 802.11 Interfaces für den Access Point. Diese unterstützte zwar das Handover eines mobilen WLAN Knotens, aber änderte seine IP Adresse nicht, was dazu führte, dass ein mobiler Host nach einem Wechsel des Access Points über das IP basierte Routing nicht mehr zu erreichen war. Diese Problem konnte in der verbliebenen Zeit nicht mehr vollends behoben werden. Es wurde der Versuch unternommen über das Modul NotificationBoard zur Laufzeit eine Änderung an den Routing Tabellen der am Handover beteiligten Knoten vorzunehmen, was aber misslang. Aussichtsreich erscheint hier eine Neuimplementierung der Managementmodule der Ieee 802.11 Netzwerkinterfaces.

C Die wichtigsten Konfigurationseinstellungen zum BriteUnderlay

```
[Parameters]

network = aBRITENetwork
**.churnGeneratorTypes = "DummyChurn MobilityChurn"
**.terminalTypes = "BriteOverlayHost BriteMobileOverlayHost"
**.channelTypes = "dummyChannel"
# hier die Anzahl mobiler Knoten angeben
**.targetOverlayTerminalNum = 10
**.underlayConfigurator.startIP="1.1.0.1"

**.overlayAppType = "KBRTestApp"
**.tier1Type = "KBRTestAppModules"
**.tier2Type = "TierDummy"
**.tier3Type = "TierDummy"

**.joinOnApplicationRequest=false
**.useCommonAPIforward = true

**.mobility.x = -1
**.mobility.y = -1
**.mobility.updateInterval = 1.0
**.mobility.speed = uniform(1.12, 1.67)
**.mobility.waitTime = 60
**.playgroundSizeX = 10000
**.playgroundSizeY = 10000

**.ppp[*].queueType = "DropTailQueue"
**.ppp[*].queue.frameCapacity = 100

**.wlan.agent.activeScan = true
**.wlan.agent.channelsToScan = ""
**.wlan.agent.probeDelay = 0.1
**.wlan.agent.minChannelTime = 0.15
**.wlan.agent.maxChannelTime = 0.30
**.wlan.agent.authenticationTimeout = 10
**.wlan.agent.associationTimeout = 10
```

```
**.channelcontrol.carrierFrequency = 2.4e+9
**.channelcontrol.pMax  = 2.0  ;[mW]
**.channelcontrol.sat   = -110
**.channelcontrol.alpha = 2
**.channelcontrol.numChannels = 1
**.channelNumber = 0

**.radio.bitrate=2E+6 ;in bits/second
**.radio.transmitterPower=2.0  ;[mW]
**.radio.carrierFrequency=2.4E+9
**.radio.thermalNoise=-110
**.radio.sensitivity=-65;85
**.radio.pathLossAlpha=2
**.radio.snirThreshold = 4  # in dB

**.relayUnitType = "MACRelayUnitNP"
**.relayUnit.addressTableSize = 100
**.relayUnit.agingTime = 120s
**.relayUnit.bufferSize = 1048576  # 1Mb
**.relayUnit.highWatermark = 524288  # 512K
**.relayUnit.pauseUnits = 300  # pause for 300*512 bit (19200 byte) time
**.relayUnit.addressTableFile = ""
**.relayUnit.numCPUs = 2
**.relayUnit.processingTime = 2us
**.relayUnit.writeScalars = false

# ip settings
**.routingFile=""
**.ip.procDelay=10us
**.IPForward=true

# ARP configuration
**.arp.retryTimeout = 1
**.arp.retryCount = 3
**.arp.cacheTimeout = 100
**.networkLayer.proxyARP = true

**.wlan.mac.address = "auto"
**.mgmt.beaconInterval = 0.05
**.mgmt.numAuthSteps = 4
**.mgmt.frameCapacity = 10
**.mgmt.ssid = "111110"
```

D Inhalt der DVD

Die Datensätze auf dieser DVD sind aufgrund ihrer Grösse (von 1-2 GByte per Datensatz) mit dem Textkompressor *bzip2* gepackt. Dieser Kompressor ist standardmässig in jeder Linux Distribution vorhanden. Entpackt werden die Datensätze mit folgendem Aufruf: *user@host:dir$* **bzip2 -d --best DATEI.bz2**.

aliasResolution/ Enthält die vollständige Liste der ermittelten Aliase

datasets/ Hier finden sich sowohl die verwendeten Datensätze, als auch die generierten Topologien

- **BRITE/** Die generierten Topologien im BRITE Format
  - **graphsNotFiltered/** Die Topologien, wie aus den Datensätzen erzeugt
  - **graphsFiltered/** Die nach der Alias Resolution erzeugten Topologien
- **OverSim/** Die aus den BRITE Topologien erzeugten NED Dateien
- **rawData/** Die benutzten Datensätze in einem vorverarbeiteten Textformat

sourcecode/ Alle im Rahmen dieser Arbeit entwickelten Programmerweiterungen und Skripte

- **BRITE/** Enthält entsprechend der Verzeichnisstruktur des Programms alle angefertigten Erweiterungen
- **OverSim/** Enthält entsprechend der Verzeichnisstruktur des Programms alle angefertigten Erweiterungen
- **scripts/** Die während der Arbeit angefertigten Skripte

statistics/ Enthält die Degree und Joint Degree Distributions der erzeugten Topologien

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 14. August 2008	
Ort, Datum	Unterschrift