Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik

SS 2009

Seminararbeit

Laptop
a location aware peer-to-peer overlay network

Chi-Jen Wu, De-Kai Liu and Ren-Hung Hwang

Eric Lacher

3. August 2009

Inhaltsverzeichnis

1 Einleitung

Overlay Netzwerke sind für peer-to-peer Infrastrukturen von grosser Bedeutung. Sie implementieren auf Anwenderschicht ein logisches Netz oberhalb einer existierenden Infrastruktur und erlauben es so, eigene, auf den Einsatzzweck hin optimierte Adressräume und Routingverfahren zu verwenden.
Laptop ist eine Implementierung eines solchen Overlay-Netzwerks mit dem Hauptaugenmerk auf Ortsbezogenheit, guter Skalierbarkeit und einem niedrigen Organisationsoverhead.
Die Ortsbezogenheit wird durch geschickte Nutzung des Underlays erreicht. Durch RTT[1]-Messungen beim Einwählen eines neuen Knotens wird ein Bezug zwischen kurzen Wegen im Underlay und Overlay hergestellt, so dass die entstehende Topologie des *Laptop*-Netzes von der des Internets profitiert.
Die Skalierbarkeit und der geringe Organisationsoverhead sind der Tatsache geschuldet, dass die Knoten in den meisten Fällen lediglich mit ihrem Eltern- bzw. Kind-Knoten kommunizieren müssen und der Routingaufwand mittels eines Caches minimiert werden konnte.
In dieser Ausarbeitung wird zunächst auf die Struktur und die Verwaltung des Netzwerks eingegangen. Betrachtet wird dabei der allgemeine Aufbau, das Verhalten bei Knoteneinbzw. austritten, sowie der Routingaufwand. Im nächsten Teil wird dann die theoretische Aufwandsberechnung thematisiert. Im vierten Kapitel werden Testergebnisse aus einer Simulation betrachtet um die Qualität der Skalierung, Ortsbezogenheit und Fehlertoleranz zu bewerten.

[1] round-trip-time

2 Struktur und Verwaltung des Netzwerks

2.1 Struktur

Die Topologie des *Laptop*-Netzwerks ist ein durch den Maximalgrad d und die maximale Tiefe L_{max} beschränkter hierarchischer Baum. Die Parameter d (meist 16 bzw. 32) und L_{max} (hier meist 4) werden beim Aufbau des Netzes einmalig festgelegt und im Betrieb nicht mehr verändert.
Die Knoten des Baumes repräsentieren einen Teilnehmer und sind beschriftet mit einer Adresse und einem Level, der den Abstand[2] des Knotens zu seinem Elternknoten beschreibt.

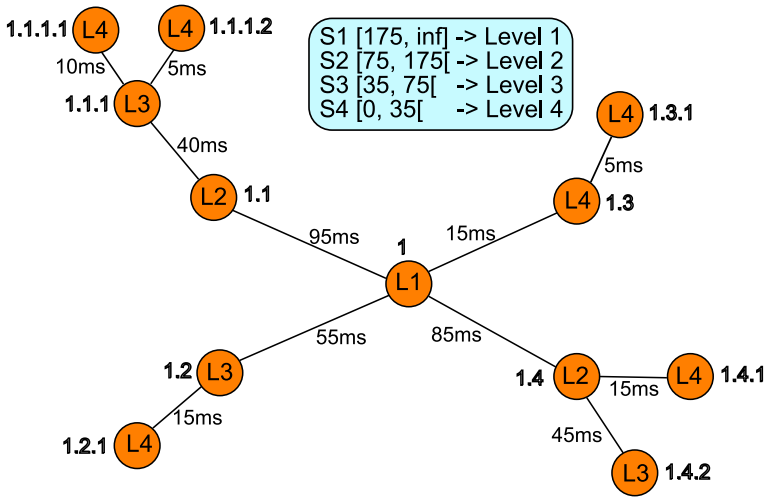

Abbildung 1: Laptop als Baum mit hierarchischer Adressvergabestruktur und Levels

2.1.1 Adresse

Der erste Knoten des Netzwerks bekommt die Adresse 1. Kindknoten erben von ihrem Elternknoten dessen Adresse als Prefix, die dann durch eine iterierende Ziffer als Suffix ergänzt wird. Die Kinder von Knoten 1 sind demnach 1.1 bis 1.d, die Kinder von Knoten 1.3 sind 1.3.1 bis 1.3.d usw. (siehe Abb. 1).

[2] RTT als Metrik

2.1.2 Levels

Um die RTT im Netz zu berücksichtigen werden Intervalle (sog. Segmente) über dieser Metrik gebildet. Je nachdem in welches Segment (S_1 bis S_{max}) die gemessene RTT zum Elternknoten fällt wird dem Knoten ein Level (L_1 bis L_{max}) zugewiesen. Der hellblaue Kasten in Abbildung 1 zeigt eine beispielhafte Unterteilung. Liegt die gemessene RTT eines Knotens zu seinem Elternknoten z.B. bei 40ms (1.1.1), so fällt er in Segment $S_3[35, 75[$ und ihm wird der Level L_3 zugewiesen.

2.2 Verwaltung

2.2.1 Knotenbeitritt

Will ein Knoten C sich als Teilnehmer in das Netzwerk einwählen, so wird eine sog. Join-Prozedur durchlaufen (s. Abb. 2). Dabei wird in einem ersten Schritt der Root-Knoten

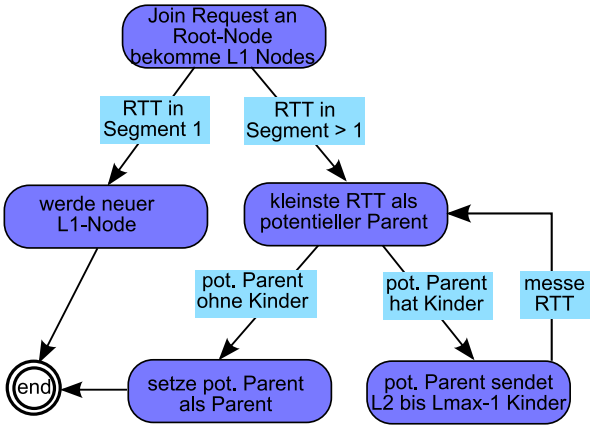

Abbildung 2: Join-Prozedur beim Knotenbeitritt

mit der Adresse 1 kontaktiert, der wiederum alle Level 1 (L_1) Knoten des Netzwerks zurückgibt. Zu diesen misst C nun die RTT um den Nächstliegenden zu ermitteln.
Liegt die kleinste RTT zu einem L_1 Knoten in Segment 1 (S_1) wird der Kandidat C zu einem neuen L_1 Knoten und verbindet sich zu diesem. Die Prozedur ist damit beendet.
Liegt die kleinste RTT jedoch in einem anderen Segment als S_1, so wird der nächstliegende

Knoten P_{pot} als potenzieller Elternknoten ausgewählt. Hat P_{pot} bereits Kinder[3], so gibt dieser alle Kinder-Knoten mit Level L_2 bis L_{max-1} zurück. Aus diesen wählt C nun den Nächstliegenden als neuen potenziellen Elternknoten P_{pot} fest und testet erneut ob diesem bereits Kinder zugewiesen wurden.

2.2.2 Knotenabmeldung und Ausfall

Für den Fall, dass ein Knoten das Netzwerk verlassen möchte oder ausfällt gibt es ebenfalls Standardprozeduren.

Möchte sich ein Knoten abmelden, so wird die *Leave*-Prozedur wie in Abbildung 3 durchlaufen. Hat der austretende Knoten K_{leave} keine Kinder, so meldet er sich lediglich bei seinem Elternknoten P ab, woraufhin dieser ihn aus der Routingtabelle löscht. Die Prozedur ist damit beendet.
Sind ihm jedoch Kinder (c_1 bis c_d) zugewiesen, so misst K_{leave} die RTT zu diesen und ernennt den nächstliegenden Kind-Knoten mit senden der *TAKEOVER*-Nachricht zu seinem Vertreter c_{proxy}. Dieser setzt den Elternknoten von K_{leave} als seinen neuen Elternknoten. Den anderen Kindern wird daraufhin mittels einer *UPDATE_PARENT*-Nachricht die Adresse von c_{proxy} mitgeteilt, den sie als neuen Elternknoten bei sich eintragen.

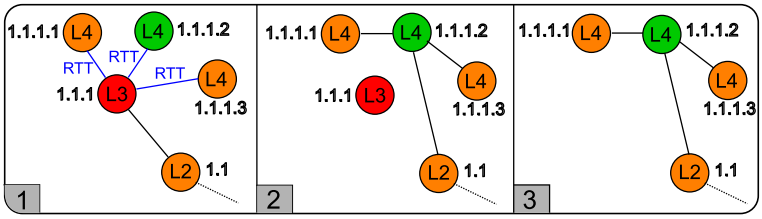

Abbildung 3: Leave-Prozedur beim Knotenaustritt (1.1.1 tritt aus, 1.1.1.2 übernimmt)

Fällt ein Knoten unvorhergesehen aus, so muss dies zunächst einmal festgestellt werden. Kindknoten senden in regelmässigen Abständen *HEARTBEAT*-Nachrichten an ihre Eltern, die wiederum mit einem *ACKNOWLEDGE* antworten sollten. Geschieht dies

[3] wird so im Paper beschrieben. Funktionaler wäre jedoch das Kriterium ob die Anzahl der Kinder bereits den Maximalgrad d erreicht hat.

nicht, wird nach einem Timeout vom Ausfall des Knotens ausgegangen.
Abbildung 4 zeigt beispielhaft wie die nun durchgeführte Prozedur aussehen kann. Bekommen die Kinder c_1 bis c_d keine *ACKNOWLEDGE*-Nachricht mehr von ihrem Elternknoten P, so senden sie an dessen Elternknoten $P_{grandparent}$ eine *CONTENTION*-Nachricht. $P_{granparent}$ misst daraufhin die RTT zu den Waisen und weist ihnen wie bei der Standardprozedur für die Knotenabmeldung den nächstliegenden Kindknoten als Vertreter zu.
Kann $P_{grandparent}$ wegen Ausfall ebenfalls nicht erreicht werden, so führen die Waisen die *JOIN*-Prozedur durch.

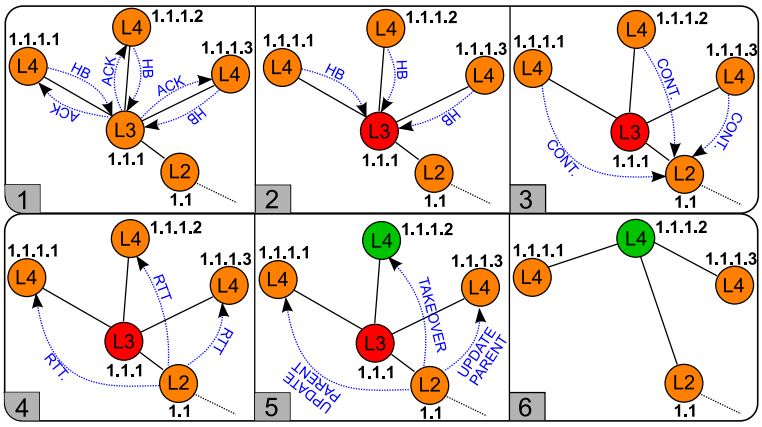

Abbildung 4: Prozedur beim Knotenausfall (1.1.1 fällt aus, 1.1.1.2 übernimmt)

2.3 Routing

Jeder Knoten hat eine Routingtabelle in der seine Kinder, der Elternknoten sowie der Root-Knoten eingetragen sind. Desweiteren gibt es einen Routing-Cache mit fester Länge C_{max} in dem z.B. aus der Join-Prozedur bekannte Knoten dynamisch eingetragen werden (siehe Abb. 5).

Um das Ziel zu finden wird *longest-prefix-matching* angewandt, bei dem die Nachricht an den Knoten der Routing-Tabelle weitergeleitet wird, dessen längstes Prefix mit der Zieladresse übereinstimmt. Will Knoten 1.1.1 zum Beispiel etwas an Knoten 1.2.1 senden, so ist das längste übereinstimmende Prefix laut Routing-Tabelle aus Abbildung 5 die

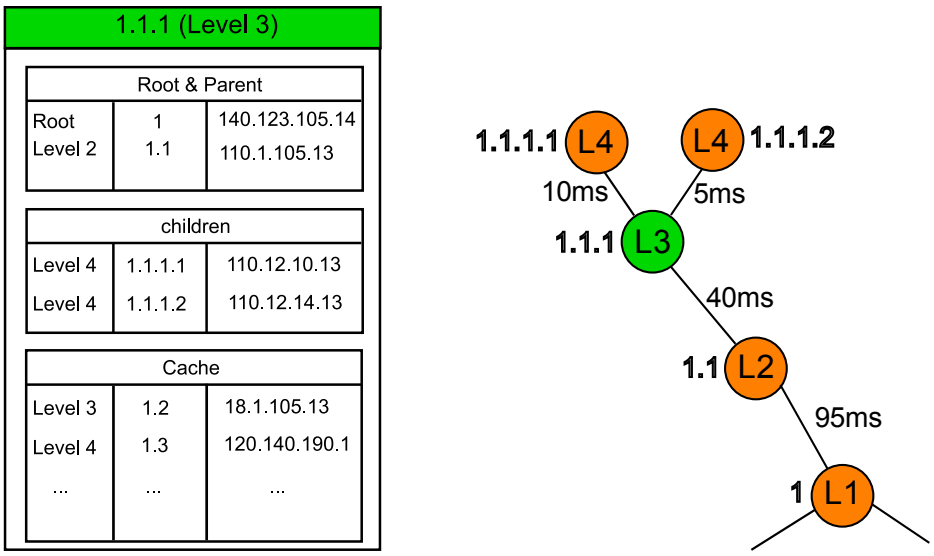

Abbildung 5: Exemplarische Routingtabelle von Knoten 1.1.1

Adresse 1.2 (aus Cache). Dieser Knoten führt die gleiche Prozedur durch und übermittelt die Nachricht direkt an seinen Kindknoten 1.2.1 (nicht dargestellt).

3 Organisationsoverhead

Die Betrachtung des Overheads wird in Beziehung zur Anzahl der in die Kommunikation involvierten Knoten gesetzt.

3.1 Overhead beim Knotenbeitritt

Beim Knotenbeitritt (siehe Kapitel 2.2.1) wird zuerst der Root-Knoten kontaktiert. Im schlimmsten Fall, wenn alle Knoten bereits (zu viele) Kinder haben, wird nun, wie in Abbildung 2 dargestellt, vom Root-Knoten bis nach ganz aussen zu einem Blatt eine Verbindung versucht und damit zu d Kindknoten die RTT gemessen. In die Kommunikation involviert ist also der Root-Knoten und bei Annahme einer balancierten Baumstruktur mit insgesamt N Knoten $\log_d N$ weitere Knoten, mit jeweils d RTT Messungen.
Der Kommunikationsoverhead für den Knotenbeitritt ist also $O(d \log_d N)$.

3.2 Overhead beim Abmelden eines Knotens

Um sich abzumelden kontaktiert ein Knoten seinen Elternknoten sowie die durch d begrenzte Anzahl Kinderknoten.
Der Kommunikationsoverhead beträgt also $O(d)$.

3.3 Overhead beim Ausfall eines Knotens

Fällt ein Knoten aus, so sind im Falle, dass der Elternknoten des ausgefallenen Knotens $P_{grandparent}$ erreichbar ist d Kinder und $P_{grandparent}$ in die Kommunikation involviert. Der Overhead wäre in diesem Fall $O(d)$.
Ist $P_{grandparent}$ jedoch nicht erreichbar, so müssen die d Kinder jeweils die Beitritts-Prozedur durchlaufen, deren Aufwand bei $O(d \log_d N)$ liegt (siehe Kapitel 3.1).
Der Aufwand liegt in diesem Fall also bei $O(d^2 \log_d N)$.

3.4 Overhead beim Routing

Im schlimmsten Fall muss eine Nachricht zuerst an den Root-Knoten und von dort aus Schritt für Schritt bis ganz nach aussen an ein Blatt weitergeleitet werden. Die Anzahl notwendiger Hops sind demnach 1 (Absender→Root-Knoten) + $\log_d N$ (Root-Knoten→Blatt).
Insgesamt liegt der maximale Aufwand beim Routing also in $O(\log_d N)$.

4 Performanz

Um die theoretischen Überlegungen zu untermauern wurden vier Tests mit dem Topologie-Generator BRITE[4] durchgeführt. Es wurde ein Netzwerk mit $1M$ Knoten die 50 AS[5] und dort jeweils 200 LANs zugewiesen wurden, simuliert.

Im ersten Test wurde der sog. *Clustering Effect* betrachtet. Dabei wird gemessen, wie gut die vom Overlay (*Laptop*) abgebildete Struktur dem des Underlay (Internet) entspricht.
Bei $d = 32$ und $N = 1M$ haben 80% der Knoten einen Abstand $<= 100ms$ (Abb. 6 links) und sind im Underlay maximal 3 Hops voneinander entfernt (Abb. 6 rechts).

[4] BRITE: Boston university Representative Internet Topology gEnerator (www.cs.bu.edu/brite)
[5] autonome Systeme

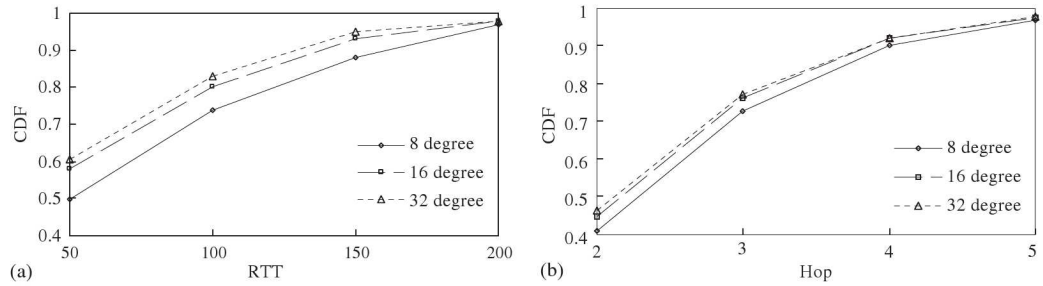

Abbildung 6: Ergebnis von Test 1: Clustering Effect
Quelle: Paper[1]

Im zweiten Test war die Weglänge beim Routing von Interesse. Theoretisch vorhergesagt war, wie in Kapitel 3.4 ersichtlich, eine maximale Pfadlänge von $\log_d N$. Gemessen wurde, wie in Abbildung 7 beschrieben, bei einem Grad d von 32 und $1M$ Knoten eine maximale Pfadlänge von 5, was ein Hop mehr als $\log_{32} 1M$ ist.

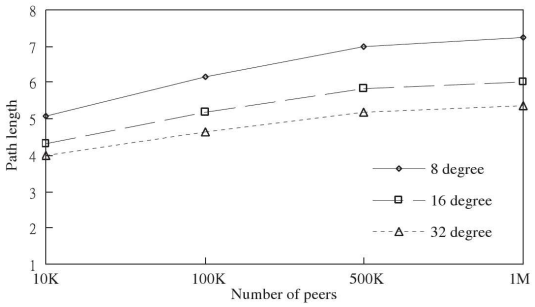

Abbildung 7: Ergebnis von Test 2: Routingpfadlänge
Quelle: Paper[1]

Im dritten Test wurde die Ortsbezogenheit gemessen. Der wichtige Faktor für diese Angabe ist der sog. *Routing-Stretch*, der die Anzahl der benötigten Hops in *Laptop* mit denen im Internet vergleicht. Ein Hop-Stretch S_{hops} bedeutet also, dass für einen Weg, der in *Laptop* 3 Hops benötigt, im Internet $S_{hops} * 3$ hops benötigt werden. Ein niedriger Wert dient also als Indikator für eine gute ortsbezogene Topologie.
Abbildung 8 zeigt nun, dass bei $d = 32$ und $N = 1M$ Knoten S_{hops} bei $4,6$ liegt. Im

Durchschnitt benötigt also eine Nachricht im Underlay 4, 6 mal mehr Hops als im Overlay.

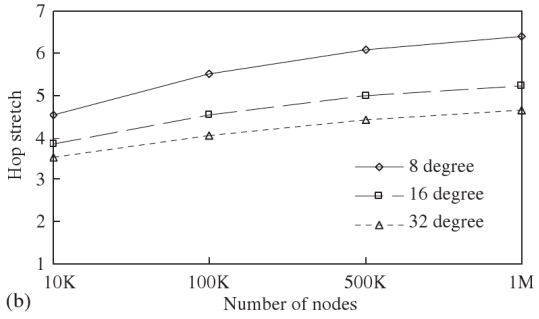

Abbildung 8: Ergebnis von Test 3: Hop-Stretch
Quelle: Paper[1]

Im vierten Test wurde schlussendlich die Fehlertoleranz geprüft. Dazu wurden schrittweise Knoten ausser Betrieb genommen um zu testen, wie sich daraufhin die Pfadlänge entwickelt.
In Abbildung 9 kann man sehen, dass selbst bei 50% an ausgefallenen Knoten die Pfadlänge bei 96% der Knoten kleiner als 10 bleibt. Bei 0% ausgefallenen Knoten lag dieser Wert bei ca. 6.

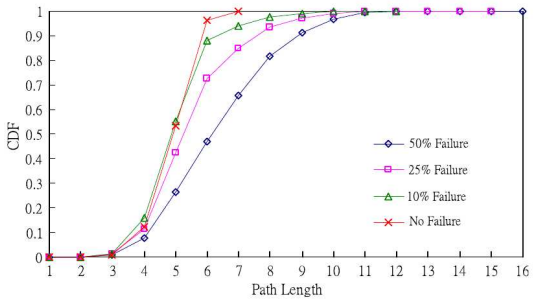

Abbildung 9: Ergebnis von Test 4: Fehlertoleranz
Quelle: Paper[1]

5 Fazit

Laptop erweist sich trotz seines niedrigen Overheads als robustes System. Die von den Entwicklern gesteckten Ziele, wie Selbstorganisation, Ortsbezogenheit, Fehlertoleranz und gute Skalierung wurden in den (simulierten) Tests erreicht. Allerdings bleibt als Wermutstropfen die manchmal doppeldeutige Formulierung im Paper, die sich um haarspalterische Randfälle, zum Beispiel für den Ausfall des Root-Knotens und die damit verbundene evtl. notwendige Neuadressierung von Teilen des Netzwerks (auf Kosten des Overheads), nicht im Detail auseinandersetzt. Bei der Implementierung für die Tests wurden solche Fälle sicher ausgespart, was für einen ersten Testlauf praktikabel ist. In einem zweiten Paper hätte man sich nun an den Ausbau des Potenzials dieser Architektur machen können, was, bei Fertigstellung dieser Seminararbeit, leider noch nicht geschehen war.
Lässt man die Ausnahmefälle bei der Betrachtung allerdings aussen vor, so kann ein durchweg positives Résumé gezogen werden. Trotz hohen Anforderungen an die Fehlertoleranz und Skalierbarkeit hängt der Aufwand für Aufbau, Instandhaltung und Routing im Netzwerk nur logarithmisch von der Anzahl der Knoten ab. Es ist erstaunlich, dass trotz äusserst sparsamer Kommunikation innerhalb des Netzwerks ein hoher Grad an Stabilität und nicht zuletzt Geschwindigkeit erreicht werden konnte.
Blendet man die Frage aus, ob eine statisch-ortsbezogene Architektur überhaupt sinnvoll ist, so könnte sich ein Blick auf das Paper von Chi-Jen Wu, Liu De-Kai und Ren-Hung Hwang durchaus lohnen.

Literatur

[1] Chi-Jen Wu, De-Kai Liu, and Ren-Hung Hwang. A location-aware peer-to-peer overlay network: Research articles. *Int. J. Commun. Syst.*, 20(1):83–102, 2007.