Diplomarbeit

Internet Coordinates

Daniel Bruggesser

Betreuerin: Regina O’Dell
Professor: Roger Wattenhofer

Inhaltsverzeichnis

1 Einleitung

Synthetische Koordinatensysteme bieten eine Möglichkeit die Verzögerungszeit oder Round-Trip-Time (RTT) zu anderen Rechnern vorherzusagen. In verschiedenen Arbeiten[11][5][9][6][14] werden gleiche oder ähnliche Systeme bereits vorgestellt. In diesem System bestimmen die Rechner ihre synthetischen Koordinaten in einem virtuellen Koordinatenraum, so dass die Distanz von zwei Rechnern im Raum proportional zur RTT zwischen den beiden Rechnern ist.

Die Arbeiten lassen sich in zwei Gruppen einteilen: die erste Gruppe schlägt ein zentralisiertes System vor, bei welchem eine Infrastruktur in Form einer kleinen Anzahl von spezialisierten Rechnern (sogenannte Landmarks in [11], [12] und [5], Tracers in [8] und 'Lighthouses' in [6]) benötigt wird. Die zweite Gruppe arbeitet mit einem dezentralisierten System, d.h. es wird keine spezielle zusätzliche Infrastruktur benötigt. Beispiele sind PIC[5], Vivaldi[9] oder Big Bang[14].

Mit synthetischen Koordinatensystemen kann ein Rechner die RTT zu einem Ziel vorhersagen, ohne dass er explizite Messungen vornehmen muss. Dazu erfährt er während einer Verbindung zu einem Rechner dessen Koordinate und kann so die RTT approximieren, oder er kann die Koordinate eines anderen Rechners beispielsweise durch eine DNS-Abfrage in Erfahrung bringen.

Die Möglichkeit die RTT vorherzusagen – ohne vorherige Messungen durchzuführen – erlaubt eine bessere Performance bei verteilten Systemen. So kann ein System den nächsten Replica-Server aufgrund der kleinsten RTT auswählen um Daten zu empfangen. Ein virtuelles Koordinatensystem kann benutzt werden, wenn eine grosse Menge von möglichen Servern zur Verfügung steht oder wenn die Datenmenge, die ausgetauscht werden soll, sehr klein ist. Im zweiten Fall ist der Overhead, um den nächsten Rechner zu finden, nicht mehr vernachlässigbar. Filesharingsysteme wie KaZaA, BitTorrent und CoDeeN sind Beispiele für Systeme, die eine grosse Anzahl Server verwalten. Ein Beispiel für ein System, das eine kleine Menge von Replica-Servern besitzt und bei dem jedes Datenpaket sehr klein ist, wäre der Domain Name Service (DNS).

Die Präzision solcher Vorhersagen einer Distanz hängt stark mit dem Aufbau des Internet zusammen. Eine sehr hohe Genauigkeit würde dann erreicht, wenn die einzelnen Rechner im Internet jeweils durch eine Punkt-zu-Punkt Verbindung miteinander vernetzt wären. So wäre die Verzögerungszeit einer Verbindung vor allem von der Lichtgeschwindigkeit, mit der sich das Signal im Netzwerk ausbreitet, abhängig. In der Realität sind die einzelnen Rechner nicht direkt miteinander verbunden, sondern die Daten werden über ISPs geroutet. Die Verzögerungszeit ist deshalb auch abhängig von der Anzahl Router auf dem Weg und deren jeweiligen Verarbeitungsgeschwindigkeit.

Die Vorhersagen der RTT und das virtuelle Koordinatensystem können auch dazu verwendet werden, einen Rechner in einem realen Koordinatensystem zu platzieren. Jede virtuelle Koordinate könnte auf eine globale Koordinate abgebildet werden, woraus sich der geographische Standort eines Rechners ablesen liesse.

Die Bestimmung eines Rechnerstandorts auf der Erde wäre hilfreich bei der Lokalisierung oder Personalisierung von Applikationen. Man könnte so die Daten schon im Voraus den geographischen Gegebenheiten anpassen. Möglich wären ein lokaler Wetterbericht oder Informationen zu Flügen in der gleichen geographischen Region.

Diese Arbeit versucht die virtuellen Koordinaten auf die Erde abzubilden (reale geographische Koordinaten). Dazu wird der Vivaldi-Algorithmus[9] verwendet. Das System ist dezentralisiert, d.h. es benötigt keine spezielle Infrastruktur - alle Rechner im System führen den gleichen Algorithmus aus. Hierbei werden alle Rechner, die den Vivaldi-Algorithmus ausführen, als ein Vivaldi-Netz zusammengefasst.

Das Ziel ist ein System, das für jeden beliebigen Rechner die geographische Länge und Breite ermittelt. Der Rechner kalkuliert seine geographische Koordinate, indem er die RTTs zu mehreren Rechnern in einem Vivaldi-Netz misst und dann selbst denselben Algorithmus anwendet, bzw. der Rechner dem Vivaldi-Netz hinzugefügt wird.

Experimente werden mit einem Vivaldi-Netz durchgeführt. Als Daten für den Algorithmus werden zuerst jene aus einem künstlichen Netzwerk benutzt. Später werden Messungen vom NLANR Active Measurement Project[1] verwendet. Dabei sind von diesen Rechnern auch die geographischen Standorte bekannt, was zur Kontrolle der Netzwerke hilfreich ist.

2 Virtuelle Koordinaten

Die Standorte der einzelnen Rechner werden zuerst in einem virtuellen Koordinatensystem bestimmt. Diese virtuellen Koordinaten sollen dann auf die Erdoberfläche projiziert werden, um die geographischen Standorte der Rechner zu bestimmen.

Aus diesem Grund beschränkt sich diese Arbeit auf 2D- und 3D-Koordinatensysteme. Andere Arbeiten[11][12][5] schlagen mehrdimensionale Koordinatensystem vor (n-Dimensionen, $n = 5 \ldots 8$). Zusätzliche Dimensionen bezwecken, die Genauigkeit der Standortbestimmung im virtuellen Koordinatensystem zu verbessern.

2.1 Vivaldi Algorithmus

Für die Simulationen in dieser Arbeit verwenden die Rechner den Vivaldi-Algorithmus[9][2]. Dabei werden alle Rechner, die den Vivaldi-Algorithmus anwenden, zusammengefasst und als Vivaldi-Netz bezeichnet.

Nun soll genauer auf den Vivaldi-Algorithmus, der jeweils von den einzelnen Rechnern in einem Vivaldi-Netz ausgeführt wird, eingegangen werden.

2.1.1 Physikalisches Modell

Die Rechner und deren Verbindungen untereinander werden simuliert durch ein Netz aus Federn. Jede Feder simuliert die Latenzzeit zwischen zwei Rechnern. Die Federlänge im Ruhezustand ist dabei gleich der Latenzzeit zwischen den beiden Rechnern.

Für die Lösung des Problems, nämlich die Bestimmung der Rechnerstandorte, müssen die potentiellen Energien aller Federn minimiert werden, wobei die potentielle Energie einer Feder quadratisch zu ihrer Verschiebung aus dem Ruhezustand ist.

Der Algorithmus berechnet aufgrund der Summe aller Kräfte, die auf den Punkt einwirken, in kleinen Zeitschritten die Verschiebung eines Punktes in diesem Federnetzwerk. Dies entspricht der Minimierung einer quadratischen Fehlerfunktion E mit $E = \sum_{i,j}(L_{ij} - dist(i,j))^2$, wobei L_{ij} die reale Latenzzeit und $dist(i,j)$ die Distanz zwischen den Rechnern i und j ist.

2.1.2 Zeitschritt

Der Algorithmus berechnet die Verschiebung des Ortes eines Rechner in einzelnen Zeitschritten. Sind die Zeitschritte gross, so nähert sich der Standort in grösseren Schritten seinem Ziel. Es besteht dann aber die Gefahr, dass der Algorithmus um eine richtige Lösung herum zu oszillieren beginnt. Werden die Zeitschritte kleiner gewählt, so verschiebt sich der Standort langsamer und die Gefahr des Oszillierens wird vermindert.

2.1.3 Lokale Fehler

Ein Rechner, dessen Standort schon nahe an seiner Zielkoordinate ist, soll nur einen kleinen Standortfehler besitzen. Für diesen Rechner ist es nicht sinnvoll, seinen Standort weiterhin in grösseren Schritten zu verschieben. Der Vivaldi-Algorithmus gewichtet den Standortfehler eines Rechners im Vergleich zum Fehler des benachbarten Rechners. Ist der lokale Fehler e_i klein und

```
// Vivaldi Algorithmus für ein Sample:
//
// benachbarter Rechner j: Distanz rtt,
// Koordinate x_j und Fehler e_j
//
// lokaler Rechner i mit Koordinate x_i und Fehler e_i
//
// Konstanten c_i und c_c sind Tuningparameter
proc vivaldi(rtt, x_j, e_j){
   // berechnet Gewicht w für diese Probe
   w = e_i/(e_i + e_j)
   // relativer Fehler dieser Probe
   e_s = |dist(x_i, x_j) − rtt|/rtt
   // lokalen Fehler nachführen
   e_i = e_s · c_e · w + e_i · (1 − c_e · w)
   // lokale Koordinate nachführen
   // (mit variablen Zeitschritt)
   t = c_c · w
   x_i = x_i + t · (rtt − dist(x_i, x_j)) · u(x_i − x_j)
}
```

Abbildung 1: Vivaldi Algorithmus

der andere Rechner hat einen grösseren Fehler e_j, so sollte sich sein Rechnerstandort weniger stark verschieben als der des benachbarten Rechners. Dies wird erreicht mit einer Gewichtung $w = e_i/(e_i + e_j)$.

2.1.4 Algorithmus

Im Codefragment in Abbildung 1 wird der Vivaldi-Algorithmus skizziert. Zuerst wird das Gewicht w berechnet, das für den variablen Zeitschritt und den lokalen Fehler genutzt wird. Der lokale Fehler wird als ein gleitender Durchschnitt berechnet und die Geschwindigkeit der Anpassung mit der Konstanten c_e gesteuert. Dieser Durchschnitt ist von den relativen Fehlern aller Verbindungen eines Rechners zu dessen Nachbarn abhängig.

Der Standort des Rechners wird in den letzten zwei Zeilen neu berechnet. Die Verschiebung an den neuen Standort kann mit der Konstanten c_c beeinflusst werden, was einer physikalischen Reibung der Federn im Modell entspricht. Die Grösse des Zeitschrittes wird in der zweitletzten Zeile berechnet in Abhängigkeit vom Gewicht des eigenen Fehlers, d.h. ist der Fehler kleiner als der Fehler des benachbarten Rechners, so wird auch der Zeitschritt kleiner ausfallen. In der letzten Zeile wird die neue Koordinate berechnet. Dabei ist $u(x_i - x_j)$ der Einheitsvektor der Richtung, in welche sich der Rechner verschieben soll. Diese Richtung wird multipliziert mit der Grösse der Veränderung $(rtt - dist(x_i, x_j)$. Falls die beiden Koordinaten x_i und x_j an der gleichen Stelle liegen, nimmt u eine zufällige Richtung ein. Dies wird v.a. nach dem Start des Algorithmus benötigt, da ein neuer Host als Initialwert für seinen Standort im Ursprung des Koordinatensystems platziert wird.

3 Koordinatensysteme

Um das virtuelle Koordinatensystem später in Form geographischer Koordinaten auf die Erdkugel abbilden zu können, sollen einfache Koordinatensysteme verwendet werden. (Nur 2D- und 3D-Koordinatensysteme).

3.1 2D-Koordinaten

Der Simulator (Systembeschreibung siehe Appendix A) benutzt für die meisten Beispiele ein zwei dimensionales Koordinatensystem. Dabei entspricht die Distanz von zwei Rechnern der Distanz derer Koordinaten im virtuellen Koordinatensystem. In einem euclidischen 2D-Koordinatensystem gilt für die Distanz d zwischen zwei Rechnern i und j: $d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Das 2D-Koordinatensystem eignet sich, weil die Simulation das synthetische Grid-Datenset (siehe Abschnitt 4.1), welches Koordinaten im 2D-Raum zur Verfügung stellt, benutzt. Für die Projektion der virtuellen Koordinaten auf die Kugeloberfläche ist dieses Koordinatensystem ebenfalls geeignet. Die Standorte auf der Kugel können mit einer kleinen Verzerrung auf die Ebene projiziert werden, wenn die abzubildende Region nicht allzu gross ist. Weniger geeignet ist dieses Koordinatensystem, wenn die ganze Erdoberfläche auf eine Ebene abgebildet werden soll, da dann die Projektionsfehler zu gross werden.

3.2 3D-Koordinaten

Dieses Koordinatensystem wurde anfänglich zur Projektion der Rechner vom NLANR-Datenset (siehe Abschnitt 4.2) benutzt. Dabei werden die Koordinaten der Rechner in ein euclidischen 3D-Koordinatensystem umgerechnet. Das ergibt Fehler im Modell, weil die kürzeste Distanz zwischen zwei Standorten duch die Erdkugel hindurch gemessen wird.

Die Einführung einer zusätzlichen Dimension sollte helfen, die Positionsfehler der einzelnen Rechner im Koordinatensystem zu verringern. Die Resultate liessen sich dadurch jedoch nur unmerklich verbessern, so dass dieses Koordinatensystem nur kurz zum Einsatz kam.

3.3 geographische Koordinaten

Mit diesem Koordinatensystem soll der Modellfehler vom 3D-Koordinatensystem behoben werden. Dabei sollen die Rechner in geographischer Länge und Breite auf einer Kugeloberfläche positioniert werden. Die kürzeste Distanz zwischen zwei Rechnern wird mittels Grosskreisen auf der Erdkugel ermittelt. Leider ergaben die Simulationen mit diesem Koordinatensystem keine verwertbaren Resultate.

4 Daten

Der Vivaldi-Algorithmus berechnet auf nur einem Computer die Koordinaten der Rechner, da keine realen Rechner im Internet verwendet werden. Dazu wird ein Simulator gebraucht, der dem Vivaldi-Algorithmus die Latenzzeiten und Verbindungen zu den benachbarten Rechnern einspeist. Zur Evaluation der virtuellen Koordinaten werden vom Simulator zwei verschiedene Datensets benutzt. Das erste Set simuliert ein synthetischen Netzwerk, beim zweiten Datenset werden real gemessene Latenzzeiten aus dem Internet verwendet.

4.1 Synthetisches Grid-Datenset

In diesem Datenset setzt der Simulator die einzelnen Rechner auf die Knoten eines gleichmässig rechtwinkligen Gitternetzes. Die Distanzen zwischen zwei benachbarten Rechnern sind immer gleich gross.

Damit ein Rechner nicht mit allen anderen Rechner verbunden ist, beschränkt der Simulator die Sichtweite eines Rechners auf eine bestimmte Distanz r. Ein Rechner unterhält somit nur Verbindungen zu Rechnern, die im Koordinatensystem innerhalb der Distanz r liegen. So kann auch getestet werden, wie gross die Distanz r sein muss, damit der Algorithmus noch sinnvoll arbeiten kann.

Aus den regelmässig verteilten Knoten in diesem synthetischen Gitternetz werden zusätzlich einzelne Rechner entfernt. Damit wird die Anzahl der Nachbarn eines Rechners reduziert, so dass die einzelnen Rechner eine unterschiedlich grosse Anzahl an benachbarten Rechnern besitzen.

Abbildung 2 auf Seite 10 zeigt ein solches Grid-Datenset.

4.2 Reales Datenset

Hier werden die Messwerte vom LANR Active Measurement Project[1] benutzt. Wie bereits erwähnt, ist bei diesem Projekt der Standort jedes Rechners bekannt. Die IP-Adresse eines Rechners lässt sich aus den Messungen herauslesen. Somit erfüllen diese Daten die Anforderungen, von einem Rechner dessen IP-Adresse, Ort und die verschiedenen Verbindungen zu den anderen Rechnern zu kennen.

Das NLANR Netzwerk besteht aus ca. 140 Rechner weltweit, wobei die grösste Anzahl Rechner in den USA liegen. Die Rechner pingen sich jede Minute gegenseitig an und führen alle paar Minuten ein Traceroute zu den anderen Rechnern durch.

Alle Resultate der Messungen können über ein WWW-Interface, mittels WebService oder als Rohdaten abfragt werden. Auf der Website findet man auch eine Weltkarte, auf der die Standorte der Rechner eingezeichnet sind.

Damit der Simulator nicht ständig Daten über das Internet abfragen muss, wird ein Snapshot der Daten lokal zwischengespeichert. Dazu werden von den Informationsseiten zu den einzelnen Rechnern die Koordinaten ausgelesen und für jede Paarbeziehung die Daten aller Traceroute eines Tages abgefragt. Daraus werden die IP-Adresse und die RTT zum anderen Rechner extrahiert. Um die Auswirkung von Paketverlusten und Verzögerungen im Internet so gering wie möglich zu halten, wird jeweils der kleinste Wert eines ganzen Tages für die RTT genommen.

5 Simulation

Der Vivaldi-Algorithmus wird auf einem einzelnen Rechner durch einen Simulator ausgeführt. Grund dafür ist die einfache und schnelle Installation auf einem einzelnen Rechner. Somit werden keine weiteren Rechner im Internet gebraucht, auf denen der Algorithmus ausgeführt werden müsste.

Der Simulator hält alle Daten (wie die Position oder den relativen Fehler für einen Rechner) im Speicher vor, jeweils als ein Objekt für jeden einzelnen Rechner.

Ausgeführt wird der Algorithmus sequentiell, d.h. die Simulation berechnet die virtuellen Positionen für die einzelnen Rechner nacheinander neu. Da der Vivaldi-Algorithmus aber ein verteilter Algorithmus ist, könnte er auch parallel ausgeführt werden. Dies wäre notwendig, wenn der Algorithmus auf realen Rechnern im Internet liefe, da diese Rechner nicht synchronisiert wären.

Der Simulator führt den Vivaldi-Algorithmus in mehreren Iterationen aus, wobei jeweils über alle Rechner iteriert wird. Bei einer Iteration wird jedes Rechnerobjekte aufgefordert, seinen Teil des Algorithmus auszuführen. Jeder Rechner verbessert dann seine virtuelle Position durch eine neue Berechnung. Dabei bestimmt er mit Hilfe des Simulators die Latenzzeiten zu all seinen Nachbarn und führt für jede Verbindung zu einem benachbarten Rechner den Vivaldi-Algorithmus (aus Abbildung 1) aus.

Der Simulator kennt eine obere Grenze für die Anzahl der Iterationen. Damit wird der Algorithmus abgebrochen, wenn er nicht zu einer genauen Lösung konvergiert. Vorzeitig beendet sich der Algorithmus nur, wenn eine zufriedenstellende Lösung gefunden wird. Dies wird anhand der relativen Fehler der einzelnen Rechner bestimmt. Liegen die Fehler aller Rechner unterhalb einer gewissen Schwelle, hat der Algorithmus eine gute Lösung gefunden und bricht ab.

Ein Rechner kann schon während der Laufzeit des Algorithmus seine virtuelle Position finden. Dieser Umstand drückt sich durch einen kleinen Positionsfehler des betreffenden Rechners aus. Damit dieser Rechner nicht bei jeder Iteration seine Position neu berechnen muss, ermöglicht die Simulation, solchen Rechnern bei einer Iteration pausieren zu können. Wie oft ein Rechner hintereinander pausieren darf lässt sich über einen Parameter bei der Simulation einstellen. Es besteht jedoch die Möglichkeit, dass sich die Positionen seiner Nachbarn während der Pausen signifikant geändert haben. Als Folge davon sollte ein pausierender Rechner regelmässig seine Position neu kalkulieren.

Dem Simulator wird beim Start mitgeteilt, welches Datenset er benutzen soll. So kann die Simulation während der Laufzeit die korrekten Werte für eine Latenzzeit einer Verzögerung in Erfahrung bringen. Wenn nun ein Rechner seinen Teil des Vivaldi-Algorithmus ausführt, benötigt er die Latenzeit für eine Verbindung zu einem benachbarten Rechner. Dieser Parameter wird nun dem Rechner vom Simulator übergeben, wonach das Rechnerobjekt seine Berechnungen autonom ausführen kann.

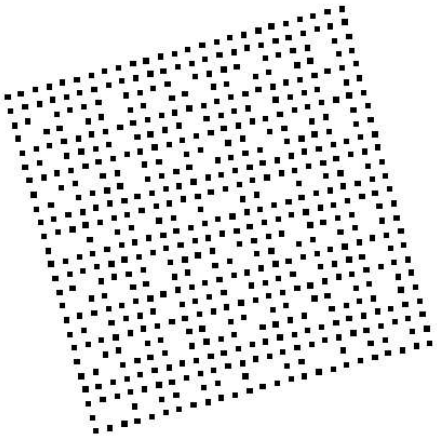

Abbildung 2: 25×25-Grid

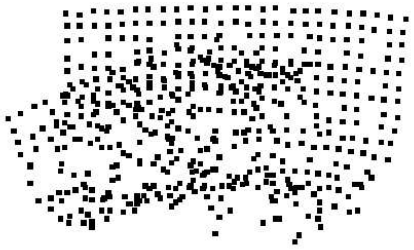

Abbildung 3: Grid Faltung

6 Resultate

6.1 Verschiedene Parameter

Die Simulation beendet sich, wenn eine bestimmte maximale Anzahl Iterationen duchlaufen wurde, oder wenn die lokalen Fehler der Rechner unterhalb einer bestimmten Schwelle liegen. Dabei werden in den Simulationen Werte im Bereich von 0.05 bis 10^{-6} benutzt. Es zeigte sich, dass aber diese Werte für das NLANR-Datenset zu klein sind – bei der Simulation bricht der Algorithmus nie vorzeitig ab.

Damit der lokale Fehler eines Rechner schnell konvergiert, wurden für den Vivaldi-Algorithmus (Abbildung 1) verschiedene Werte für die Konstante c_c ausprobiert. Wie gut der Wert von c_c wirklich ist, kann anhand der Gesamtanzahl der Iterationen beurteilt werden. Für kleine Werte ($c_c < 0.01$) konvergiert der Algorithmus sehr langsam. Für zu grosse Werte ($c_c > 0.5$) konvergiert er schnell, doch ist es in diesem Fall möglich, dass der Algorithmus um eine gute Lösung herum oszilliert. Ein Wert für c_c von 0.25 erwies sich als guter Kompromiss. Teilweise hängt der Wert noch von der Netzwerkgrösse und der Anzahl Verbindungen eines Rechners ab – weshalb c_c eventuell noch ein wenig kleiner gewählt werden sollte.

Im Simulator kann ein Rechner eine gewisse Anzahl Iterationen pausieren, wenn er seine virtuelle Endposition schon gefunden hat. Für eine bessere Performance ist es sinnvoll, wenn dieser Wert gross ist, andererseits soll der pausierende Rechner seine Position von Zeit zu Zeit wieder neu berechnen. Darum werden in der Simulation Werte im Bereich von 1 bis 5 Pausen verwendet.

6.2 Evaluation des Algorithmus

Grid Resultate

Für die ersten Beispiele benutzt die Simulation das Grid-Dataset und der Algorithmus bricht vorzeitig ab, wenn die Summe der relativen Fehler kleiner als 10^{-6} ist.

Für die Simulation mit den synthetischen Daten werden für ein 5×5-Netzwerk ca. 20 Iterationen durchlaufen. Für ein 25×25 grossen Netzwerk (Abbildung 2) werden ca. 100 Iterationen benötigt, wobei in beiden Fällen die Sichtweite eines Rechners 10 beträgt. D.h. er hat Verbindungen zu benachbarten Rechnern, die innerhalb dieses Umkreises vom Rechner entfernt sind. Diese Zahlen variieren ein wenig, da der Vivaldi-Algorithmus die Richtung für die Verschiebung zufällig wählt. Der Fall, dass die beiden Rechner den gleichen Standort haben, tritt hauptsächlich beim Starten des Vivaldi-Algorithmus auf.

Bei den beiden vorherigen Beispielen wurden alle Rechner zum gleichen Zeitpunkt dem Netzwerk hinzugefügt. Es besteht aber auch die Möglichkeit, Rechner erst hinzuzufügen, nachdem der Algorithmus die Standorte für alle Rechner im Vivaldi-Netz berechnet hat. Es lässt sich erkennen, wie die schon gefunden virtuellen Koordinaten fest bleiben und sich die neu hinzugekommenen Rechner ihre Koordinate neu suchen.

Wählt man beim Grid-Datenset die Sichtweite eines Rechners zu klein, so kann es vorkommen, dass die Rechner an den Rändern des Netzwerks ihre Koordinaten schon gefunden haben, die Rechner innerhalb aber noch nicht an der richtigen Position angelangt sind. Es erscheint, als ob das Gitternetz gefaltet ist, ähnlich einem Blatt Papier, dass man an seinen gegenüberliegenden Rändern zusammen hält (Abbildung 3). Im Fall einer zu kleinen Sichtweite kommt es dann meist vor, dass sich das Vivaldi-Netz nicht mehr entfalten kann. So bleiben die relativen Fehler der Rechner im Falt gross und die Simulation wird abgebrochen, weil die maximale Anzahl Iterationen erreicht ist.

Bei einem Grid-Netzwerk, bei dem alle Rechner gleichzeitig hinzugefügt werden, sollte die Sichtweite grösser sein als der halbe Durchmesser des Netzwerks, damit der Vivaldi-Algorithmus zu einer Lösung konvergiert. Je weiter dann die Sichtweite ausgedehnt wird, um so schneller wird der Algorithmus eine Lösung finden.

Für Grid-Netzwerke, bei welchen eine Gruppe von Rechnern erst nach dem Start des Algorithmus dem Netzwerk hinzugefügt werden, sollte der Radius der Sichtweite grösser sein als der der Durchmesser der zuletzt hinzugefügten Rechner. Wird z.B. einem 15×10-Grid auf einer Seite noch eine Gruppe von 15×5 Rechnern hinzugefügt, so sollte die Sichtweite grösser als 5 sein. Sonst ergeben sich Phänomene ähnlich der oben erwähnten Faltung.

Resultate NLANR

Für die Simulationen mit dem NLANR-Datenset wurde die Anzahl der Rechner auf alle Rechner mit Standort USA reduziert (Abbildung 4). Ein Grund dafür ist, dass sich nur eine handvoll Rechner ausserhalb der Staaten befindet. Weiter war die Absicht, ein möglichst einfaches Koordinatensystem zu benutzen, damit die Kugeloberfläche ohne grosse Verzerrungen auf eine Ebene abgebildet werden kann.

Bei der Simulation mit dem NLANR-Datenset bleiben die lokalen Fehler im Durchschnitt über 20%. Es ist anzunehmen, dass einige Rechner nicht optimale Daten liefern. Schliesst man die Rechner mit den höchsten relativen Fehlern aus, so reduzieren sich die durchschnittlichen Fehler auf ca. 10-15%. Andere Arbeiten [5][9][4][2] kommen zu ähnlich hohen Ergebnissen bei den Fehlern.

Diese hohe Fehlerrate ist auch Ursache für die spontanen Verschiebungen einzelner Rechner. Wenn die Koordinaten visuell betrachtet werden (siehe Appendix A), während der Algorithmus

Abbildung 4: NLANR: Dataset USA

noch läuft, verschieben sich immer wieder spontan einige Punkte. Die Ursache dafür ist, dass die Simulation die Reihenfolge, wie über die einzelnen Rechner und deren Verbindungen zu den anderen Rechnern iteriert wird, zufällig wählt. Wenn aber der Simulator in immer der gleichen Reihenfolge über die einzelnen Rechner iteriert, so bleiben diese spontanen Verschiebungen mehrheitlich aus. Der Algorithmus findet auch in diesen Fällen kein vorzeitiges Ende, da die relativen Fehler der Rechner ebenfalls über 20% liegen.

Dieses letzte Szenario ist leider realitätsfern, da die Iterationsreihenfolge nur in einer Simulation geordnet werden kann. Der Vivaldi-Algorithmus ist aber auch für verteilte Systeme geeignet. Wenn der Algorithmus auf Rechnern im Internet liefe, wären das voneinander unabhängig ablaufende Prozesse. In diesem Fall würde die Ordnung in der Reihenfolge der Rechner fehlen.

Im Hinblick auf die Abbildung der virtuellen Koordinaten auf die Erdkugel wird nun das Resultat noch visuell betrachtet. Dabei werden die virtuellen Koordinaten der Rechner (Abbildung 4) mit den realen Standorten in den USA (Abbildung 5) visuell verglichen. Es ist durchaus möglich, dass das Abbild der virtuellen Koordinaten gedreht oder gespiegelt ist. Beim Vergleich der beiden Karten kann man vereinzelte Häufungen von Rechnerpositionen nur mit viel Fantasie einer Region auf der Erdkugel zuordnen. In den virtuellen Koordinaten in Abbildung 4 scheint es, dass die dunklere Region rechts oben der Region um New York und die Anhäufung in der Bildmitte der Region San Francisco entsprechen könnte.

Diese ungenauen Zuordnungen wiederholen sich auch dann, wenn nur die Rechner berücksichtigt werden, die sich geographisch im Westen oder im Osten der USA plazieren lassen. Diese Tatsache war der Grund, dass alle Versuche scheiterten, die virtuellen Koordinaten der Rechner auf die Erdkugel abzubilden.

Abbildung 5: NLANR: geographische Karte USA

7 Diskussion

Leider wurde kein Weg gefunden, die virtuellen Koordinaten sinnvoll auf die Erdkugel zu projizieren. Die Hauptursache waren sicherlich die hohen relativen Fehler der Rechner im NLANR-Datenset. Dafür scheint es folgende Ursachen zu geben:

- das 2D-Koordinatensystem
- Fehler in den NLANR-Daten
- Aufbau des Internets
- Fehler im Simulator

Koordinatensystem

Bei den Simulationen wurden oft alle Rechner mit einbezogen, welche in den USA liegen. Vereinzelt wurde auch nur mit den Rechnern im Osten oder Westen der USA gearbeitet. Die Verzerrung durch das Abbilden der Kugeloberfläche auf eine zwei dimensionale Oberfläche ist sicherlich vorhanden, hält sich aber in Grenzen, wenn man nur mit den Rechnern aus Teilregionen der USA arbeitet.

Dennoch vermute ich, dass dieser Fehler keine grossen Auswirkungen hat. Die Ergebnisse werden nicht besser, wenn man die Anzahl der Rechner durch das Verkleinern der Region reduziert.

NLANR Daten

Für die Simulation wird der kleinste Wert der RTT innerhalb 24 Stunden zwischen zwei Rechnern genommen. Dies ist sicher besser, als den Durchschnitt der RTTs zu nehmen.

Auffallend war, dass einzelne Rechner nicht auf die Messung eines anderen Rechners reagiert haben. Es gibt sogar Rechner, die zu einzelnen anderen Rechnern innerhalb der letzten Stunden oder Tage gar keine Verbindung hatten. Hier stellt sich die Frage, ob die Qualität der Daten genügend sei.

Andererseits müssen aber die geographischen Koordinaten der Rechner vorhanden sein, damit man später die virtuellen Koordinaten auf die Erde abbilden kann. Das NLANR-Datenset war die einzige Quelle, die im Internet gefunden wurde und welche diese Anforderungen erfüllt.

Internet

Der Aufbau des Internets ist oft noch wie eine Blackbox. Niemand kennt die genaue Struktur des Internets. Die Datenpakete kommen über mehrere Zwischenstationen zum Ziel. In jedem Router wird noch eine zusätzliche Verarbeitungszeit benötigt. Auch sind die Bandbreiten und Distanzen der Teilstrecken, die ein Paket zurücklegen muss, nicht alle bekannt. So kann eine Teilstrecke überlastet sein und es kommt zum Stau, oder im schlimmsten Fall zu Paketverlusten. Und schliesslich weiss niemand, ob ein Datenpaket auf dem geographisch direkten Weg zum Ziel weitergeleitet wird oder ob es einen Umweg über eine grosse Metropole nimmt.

Hier spielen die ISPs eine wichtige Rolle. Aufgrund welcher Policies werden die Daten überhaupt geroutet? Es ist doch immerhin möglich, dass ein ISP ein Paket mit grösserer Wahrscheinlichkeit an einen Partner weiterleitet, statt dieses an einen direkter ans Ziel angebundenen ISP zu schicken. Hier spielen wirtschaftliche und politische Faktoren eine Rolle, die meistens über der logisch kürzesten Verbindung stehen.

Bekannt ist auch nicht, wie die einzelnen Rechner des NLANR Projekts miteinander verbunden sind. Hängen alle an einem ISP (z.B. Internet2 in den USA) oder sind einzelne Rechner über kleine ISP, deren Verbindungen einen Engpass darstellen, angeschlossen?

Alle diese Punkte können einen Einfluss auf die Verzögerungszeit eines Datenpaketes haben. Darum kann es möglich sein, dass der Vivaldi-Algorithmus nur Positionen mit grossen relativen Fehlern berechnet, obwohl die Messungen des NLANR-Datasets korrekt die schnellste Verbindung ergeben haben.

Fehler im Simulator

Natürlich besteht die Möglichkeit, dass das Simulatorprogramm nicht fehlerfrei ist. Programmierfehler kommen in allen möglichen Programmen vor. Weiter wäre es natürlich möglich, dass der Simulator von Design her fehlerhaft ist – zu 100% kann das nicht ausgeschlossen werden.

8 Related Work

Eine der ersten Arbeiten war Global Network Postions (GNP)[11]. Die Autoren zeigten, dass eine Positionierung der Rechner in einem virtuellen Koordinatensystem möglich ist. Das System unterteilt dabei die Rechner in Landmarks und normale Hosts. Ein Rechner, der neu in dieses System hinzukommt, kann jetzt seine Position zu diesem fixen Set von Landmarks bestimmen. Im Unterschied zum Vivaldi-Algorithmus wird bei diesen System eine spezielle Infrastruktur aufgebaut.

Ebenfalls eine spezielle Infrastruktur baut IDMaps[8][7] auf. Dieses System besteht aus einigen hundert Tracern. Diese messen jeweils die Verzögerungen zu jedem anderen Tracer. Sie messen auch die Verzögerung zu jedem CIDR Adress Präfix. Die Distanz zwischen zwei Rechnern wird nun aus folgenden Verzögerungszeiten summiert: vom Präfix zu dessen zuständigen Tracer, von einem zweiten Tracer zum Präfix des Zielrechners und vom ersten zum zweiten Tracer.

Eine Erweiterung zu GNP ist Lighthouse[6]. Auch hier gibt es spezielle 'Landmark'-Rechner. Wenn ein Rechner neu hinzukommt, kann er sich aber im Gegensatz zu GNP ein beliebiges Set von Landmarks aussuchen und seine Position relativ zu dieser Basis bestimmen. Danach transformiert er die Koordinate in ein globales Koordinatensystem.

Viele Arbeiten beziehen sich auf GNP[11]. Unter anderem die Arbeiten [12] und [10]. Es wird dann gezeigt, wie man mit kleinen Veränderungen an GNP die Qualität der Lösung verbessern kann. Ein Beispiel ist das Ersetzten der Fehlerfunktion durch eine verbesserte Funktion.

PIC[5] ist ein dezentralisiertes System. Jeder neue Rechner kann ein beliebiges Set von Rechnern auswählen, die ihre Position schon bestimmt haben. Im Gegensatz zu Vivaldi berechnet PIC die Positionen der Rechner nicht fortlaufend, sondern sobald er die Verzögerungen zu den Landmarks gemessen hat. Zusätzlich kann sich PIC vor feindlichen Rechnern (malicious nodes) schützen, indem mit der Dreiecksungleichung die Zuverlässigkeit einer Messung getestet wird.

Ähnlich wie bei Vivaldi liegt dem Big Bang System[14] auch ein physikalisches System zugrunde. Dabei wird die Position der Rechner mit Hilfe eines Potentialfeldes ermittelt. Es wird jeder Rechner als ein Partikel im Potentialfeld simuliert und das Moment jedes Partikels berechnet. Zusätzlich wird eine Reibung eingeführt, damit die Partikel in einen stabilen Zustand konvergieren.

Ein Sytem, das die Rechner nicht in ein virtuelles Koordinatensystem einbettet, sondern lediglich die Verzögerungszeit zwischen zwei Rechnern bestimmen kann, ist King[3]. King geht davon aus, dass sich in der Nähe jedes Rechners ein DNS Server befindet, der auch für die Auflösung des Rechnernamens zuständig ist. Die Latenzzeit zwischen den beiden DNS Servern wird wie folgt bestimmt: zuerst wird die Verzögerung zum ersten DNS Server gemessen, indem man den ersten Rechnernamen auflöst und dabei die Zeit misst. Danach wird der zweite Rechnername mittels einer rekursiven DNS Abfrage über den ersten DNS Server zum Zweiten aufgelöst und wiederum die Zeit gemessen. Die Differenz der beiden gemessen Zeiten approximiert den Abstand zwischen den beiden Rechnern, da ja die Verzögerung von einem Rechner zum DNS Server vernachlässigt wird.

Und schliesslich sei noch eine weitere Arbeit erwähnt, die ohne virtuelles Koordinatensystem auskommt: die Geographic Mapping Techniques[13] können Rechner einzelnen Regionen

oder Orten zuordnen. GeoTrack benutzt den DNS Namen des Rechners und durchsucht ihn nach irgendwelchen geographischen Hinweisen wie Städenamen, Länder- oder Staatenkürzel. GeoCluster durchsucht die BGP Prefix Informationen nach Kürzel, Namen oder anderen Hinweise, um so auf den geographischen Ort des Rechners schliessen zu können.

A System Übersicht

Da alle Experimente mit einem Simulator ausgeführt wurden, folgt hier eine kurze Beschreibung des Simulators.

Der Simulator ist in Java geschrieben und wird mit Packages in seiner Funktionalität unterteilt. Die wichtigsten Teile sind: Vivaldi, Net, Geom, Simulator und Visual. Es folgt für jeden Punkt eine kurze Beschreibung.

Vivaldi

Die Klassen in diesem Package kapseln den ganzen Vivaldi-Algorithmus. Die Klasse Vivaldi selbst wird von einem Anwendungsprogramm benutzt und stellt diesem u.a. eine Methode zur Verfügung, um den Vivaldi-Algorithmus für eine Iteration auszuführen. Vivaldi macht intensiven Gebrauch von der Klasse Net und benutzt den Simulator, um die Verzögerungszeiten zwischen zwei Rechnern abzufragen.

Die Klasse Vivaldi ist auch eine Fassade für die Klasse Net, damit der interne Umgang mit dem Netz nicht sichtbar wird. Sie stellt ebenfalls Hilfsfunktionen bereit, damit u.a. alle Rechner im Netzwerk abgefragt werden können.

Im Package ist auch die Klasse Host enthalten. Darin werden alle Daten gespeichert und auch der Algorithmus wird hier in einer Methode bereitgestellt.

Net

Net bleibt hinter Vivaldi vor dem Anwender verborgen. Es verwaltet die Struktur des Netzwerks, d.h. es werden alle Rechner und deren Verbindungen zu anderen Rechnern verwaltet.

Geom

Dieses Paket stellt die verschiedenen Koordinatensysteme bereit. Mit einer Abstrakten Fabrikmethode kann der Algorithmus neue Positionsvektoren erzeugen. So ist sichergestellt, dass der Vivaldi-Algorithmus unabhängig vom Koordinatensystem arbeiten kann.

Simulator

Die Schnittstelle, wie ein Benutzerprogramm oder Vivaldi selbst mit dem Dataset interagieren können, wird in einem Interface festgelegt. Züsätzlich hat es für jedes Dataset eine Impementation. Speziell an der Variante für die NLANR Daten ist, dass es die heruntergeladenen Daten zwischenspeichert. Diesen Cache könnte man auch ausschalten, damit bei jeder Anfrage die aktuellen Daten aus dem Internet herunter geladen werden.

Visual

Dieses Paket stellt ein einfaches visuelles Interface bereit, mit welchem die Vivaldi-Netzwerke angeschaut werden können. Die Daten können auch fortlaufend aktualisiert werden, damit der Verlauf des Algorithmus besser beobacht werden kann. Ohne dieses visuelle Interface wären die Benutzerprogramme darauf beschränkt, lediglich über die Konsole mit dem User zu interagieren.

Literatur

[1] The NLANR active measurement project. http://amp.nlanr.net/active/.

[2] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proceedings of ACM SIGCOMM 2004*, Portland, Oregon, USA, August 2004.

[3] Krishna P. Gummandi, Stefan Saroiu, and Steven D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proceedings of 2nd ACM SIGCOMM Internet Measurement Workshop (IMW'02)*, pages 5–18, Marseille, France, November 2002.

[4] Liying Tang and Mark Crovella. Virtual Landmarks for the Internet. In *Proceedings of Internet Measurement Conference (IMC'03)*, pages 143–152, Miami (FL), USA, October 2003.

[5] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet Coordinates for Distance Estimation. Technical Report MSR-TR-2003-53, Microsoft Research, Cambridge, CB3 0FB, UK, September 2003.

[6] Marcelo Pias, Jon Crowcroft, Steve Wilbur, Tim Harris, and Saleem Bhatti. Lighthouses for Scalable Distributed Location. In *Proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, February 2003.

[7] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A Global Internet Host Distance Estimation Service. In *IEEE/ACM Transactions on Networking*, October 2001.

[8] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel F. Gryniewicz, and Yixin Jin. An Architecture for a Global Internet Host Distance Estimation Service. In *Proceedings of IEEE INFOCOM 1999*, pages 210–217, New York (NY), USA, March 1999.

[9] Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, Distributed Network Coordinates. In *Proceedings of 2nd Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge (MA) USA, November 2003.

[10] T. S. Eugene Ng and Hui Zhang. Towards Global Network Positioning.

[11] T. S. Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinate-Based Approaches. In *Proceedings of IEEE INFOCOM 2002*, pages 170–179, New York (NY) USA, June 2002.

[12] T. S. Eugene Ng and Hui Zhang. A Network Positioning System for the Internet. In *Proceedings of USENIX Conference*, June 2004.

[13] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *Proceedings of ACM SIGCOMM 2001*, pages 173–185, San Diego (CA), USA, August 2001.

[14] Yuval Shavitt and Tomer Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. In *Proceedings of IEEE INFOCOM 2003*, San Francisco (CA), USA, April 2003.