Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik
Prof. Dr.Prof. Dr. Christian Schindelhauer

WS 09/10

Seminararbeit

Routing in a Delay Tolerant Network

S. Jain, K. Fall, R. Patra

Achille Nana Tchayep

8. Februar 2010

Abstract

Delay Tolerant Network abgekürzt **DTN** beschreiben Netzwerke, wo die End-to-End-Verbindung zwichen Knoten eine sehr große Verzögerungszeit annehmen kann. Ausserdem in einem **DTN** gibt es keine dauerhaften verbindungen zwischen Knoten. Die Anzahl von Nachrichten im Netz soll also möglichst geringt gehalten werden, um den Datenverlust zu minimieren. In einem **DTN** verfügt jeder Knoten über einen endlichen Buffer, wo die ankommenden Daten gespeichert werden. Diese Daten sollen, solange gespeichert werden, bis einen Weg gefunden wird, sie zum Zielknoten weiterzuleiten, falls die Daten für den Knoten selbst nicht bestimmt sind. Wir sehen also, dass die Routingstrategie in einem **DTN** anders als bei einem gewöhnlichen Netz abgewicklet wird. Genau dieser Punkt wollen wir hier näher betrachten.
Ein Framework wurde speziell zum Performancecheck entwickelt. Mit Hilfe dieses Framework wollen wir verschiedene Routing-Alogoritgmen in einer **DTN**-Umgebung untersuchen.

Inhaltsverzeichnis

1 Einführung

In diesem Paper wollen wir die Probleme beim Routing in einem **DTN**-Netz unter die Lupe nehmen. Für das Internet-Netzwerk muss gesichert werden, dass es einen stabilen Weg zwischen den Knoten gibt, die Knoten an Energie versogt wird und die Bandbreite genügend ist. In manchen Umgebungen sind diese Voraussetzen schwer auszufüllen, wie z.B: Verbindung Satellite-Erde, Verbindung zwischen Planeten oder auch im Ad Hoc Netzwerke, wo die Anfrage- und Antwortzeit sehr groß sein können, oder auch die lückenhafte Verteilung der Knoten die Verbindung zwischen denen erschwert. Die **DTN**-Prinzip wäre ein Alternativ die Verbindung in solcher Umgebung zu ermöglichen.

2 Die Strategie

Im Allgemein beim Routing handelt es sich um eine Weiterleitung von Paketten von einer Quelle bis zu einem Ziel. Dabei soll es gesichert werden, dass so wenig Daten wie möglich verloren gehen. Ausserdem sollen die Daten in dem richtigen Zeitpunkt eintreffen, denn ein Paket, dass zu spät ankommt, ist einfach unbrauchbar.
In diesem Teil wollen wir eine Reihe von Routing-Techniken vorstellen und ihre Wirkungen in einem **DTN**-Netz näher betrachten.

2.0.1 Die Ziele

Im Gegensatz zum traditionellen Netz ist die Verbindung zwischen Knoten im **DTN**-Netz sehr zeitweilig, denn die Knoten unstabil sind. Die traditionellen Routing-Algorithmen können in dieser Umgebung nicht angewandt werden. Im **DTN**-Netz ist die Verbindung zwischen Knoten unzuverlässig und die Bandbreite ist generell sehr schwach.
Ein **DTN**-Netz soll mehere Eigentschaften aufweisen. Eine wichtige davon ist die Sparsamkeit. Die Knoten müssen energiesparend sein. Also die Anzahl der Datenübertragung und Datenkopie sollen minimisiert werden. Wir können noch weiteren Kriterien ernennen:

- Der Datenverlust soll während der Datenübertragung möglichst geringt gehalten werden.
- Die Minimierung der Verzögerungszeit während der Datenübertragung.

Da jetz die Ziele festgelegt sind wollen wir nun die unterschiedlichen Routing-Strategien vorstellen. Gewöhnlich kann man 2 Type unterscheiden: das *Proactive Routing* und das *Reactive Routing*.

2.1 Proactive Routing

In diesem Fall wird die Route unabhängig der Datenverkehr berechenet. Im **DTN**-Netzwerk wird diese Berechnung ausscließlich für die aktuellen verbundenen Knoten durchgeführt, denn die Knoten hier nicht immer verfügbar sind. Falls ein Knoten nicht Online ist, fehlt dem Algorithm schlagt. Die Nachrichten zwischen Router werden schwer weitergeleitet. Also wird die Termierung des Algorthmus nicht immer gewährleistet. Durch die stetige Trennug der Knoten vom Netz werden viele *update*-Nachricht verschickt, die zu einer Überlastung des Netzes führen kann. Allerdings macht der **DTN**-Routing Algorithmus von dieser Lücke gebraucht. Das Wiessen über die nicht erreichbaren Knoten ist nützlich die optimale Route unter den aktuellen erreichbaren Knoten zu berechnen.

2.2 Reactive Routing

Die Route wird nur per Bedarf berechnet und dies für jeden neuen erreichbaren Knoten. Also wenn eine Route zu einem bestimmten zielknoten benötigt wird, wird eine globale Route-Entdeckungsprozedure aufgerufen, um eine bestimmte Information zu bekommen, die vorher nicht bekannt war. Im **DTN**-Netz die Anwendung dieses Algorithmus verlangt, dass die Clients verbunden sind. Der *reactive Routing*-Algorithmus kann für Antwortmangel (während der Routeberechnung) fehlschlagen, im Gegensatz zum *practive Routing*, der in einer vergleichbaren Situation viel früher fehlschlagen kann., denn er wird festellen, dass der Knoten nicht erreichbar ist.

Die Abbildung 1 zeigt wie dynamisch die Topologe in einem Ad Hoc Netz sein kann. D.h der Weg zum Ziel variiert sehr stark mit der Zeit. Also in einem *DTN*-Netz soll es ein Alogorithmus entworfen werden, der genügend Kenntnisse über den Dynamimus des Netzes verfügt. Der sogennante *route stability* wäre ein guter Kandidat zur Lösung dieses Problem, denn hier wird gemessen wie lang die aktuelle berechnete Route gultig sein wird. Für stabile Wege wäre schon denkbar ein *Route-Caching* anzuwenden, denn somit wird die Anzahl von Datenübertragen (für die Berechnung einer Route) im Netz minimiert. Dieser Faktor kann sich in einem *DTN*-Netz nutzlich erweisen. Ausserdem mit zusätzlichen Kenntnissen über den Dynamismus der Netztopologie wäre es einfacher veraltete Einträge im Cache zu löschen. Die Kenntnisse über das Netz ist also ein grosßer Vorteil die Route zu berechnen. Aus dieser Grundlage haben wir ein Routing Evaluation Framework entworfen, um die Performance des Routing in einem **DTN**-Umgebung zu untersuchen. Mehr darüber später.

3 Routing-Evaluierung Framework

Die Routeberechnung in einem **DTN** ist ein sehr komplexe Aufgabe, denn die Lösung hängt von mehreren Faktoren ab. Um dieses Prinzip besser zu nachvollziehen, haben wir verschiedene abstrakte Elemente definiert, sogennannte *knowledge oracles*, die Schlüsselinformation über das Netz enthalten. Unsere Aufgabe besteht hier darin, eine Art von Zusammensetzung zwischen der Performance der Algorithmen und die *knowledge oracles* zu finden.
Die Abbildung 2 zeigt einen Überblick über die Algorithmen und die *knowlege oracles*, die wir hier vorsellen werden.

3.1 Knowlege Oracles und die Algorithmen

Wir wollen in diesem Teil zuerst vershiedene *knowlege oracles* vorstellen. Sie verfügen über Informationen über das gesamte Netz und die verfügbaren Ressourcen im Netz. Sie sind somit für die Routing-Entscheidung sehr wichtig. Wir werden danach Algorithmen vorstellen, die diese Informaionen benutzen, um den Weg eines Datenpakets zu bestimmen.

- *Contacts Summary Oracle*: Für jeden Knoten gibt er Information über die Wartezeit bis zur nächsten Nachbarnknoten zurück.
- *Contact Oracle*: Für zwei Knoten gibt er Information über die dazwischen liegenden Knoten zurüvk. Diese Informationen werden von *Contact Summary Oracle* benötigt.
- *Queuing Oracle*: Verfügt jede Zeit über Informationen der Speicherbelegung des Buffer für jeden Knoten. Dieses *Oracle* könnte sehr aufwändig zu realisieren sein, denn er benötigt Information sowohl über die im System neuen eingehenden Nachrichten als auch die vom Routing-Algorithmus betroffene Wahl.
- *Traffic Demand Oracle*: Die Information über die Aktuelle und die zukunftige Datenverkehr im Netz kann von diesem *Oracle* gegeben werden.

In der Tabelle unter der Abbildung 2 haben wir eine Liste von Algorithmen gegeben und die dazu passenden *Oracles*. Wie wir es oben definiert haben, verfügt jedes *Oracle* über eine bestimmte Information über das gesamte Netz. Die hier zu vorstellenden Algorithmen werden Gebrauch von diesen Informationen machen um ihre Routing-Strategie

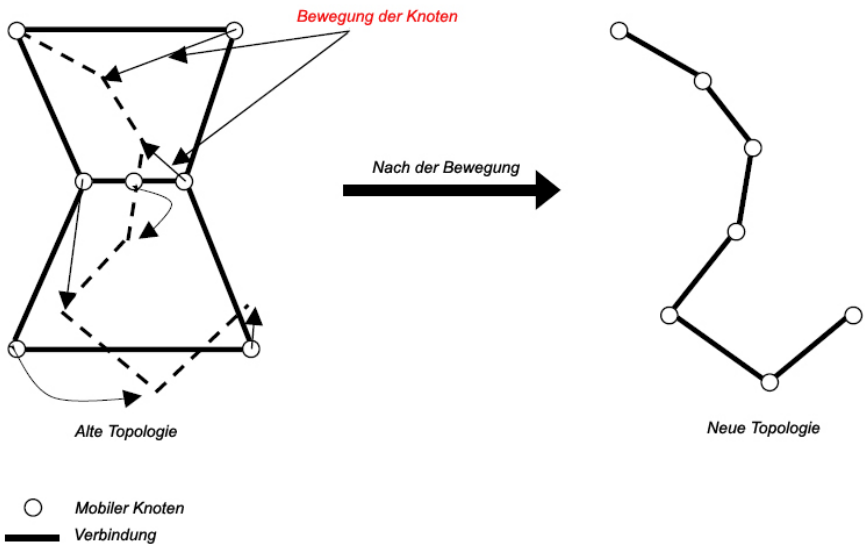

Abbildung 1: Dynamische Topologie im Ad Hoc Netzwerk

Abbr.	Name	Description	Oracles Used
FC	First Contact	Use any available contact	None
MED	Minimum Expected Delay	Dijkstra with time-invariant edge costs based on average edge waiting time	Contacts Summary
ED	Earliest Delivery	Modified Dijkstra with time-varying cost function based on waiting time	Contacts
EDLQ	Earliest Delivery with Local Queue	ED with cost function incorporating local queuing	Contacts
EDAQ	Earliest Delivery with All Queue	ED with cost function incorporating queuing information at all nodes and using reservations	Contacts and Queuing
LP	Linear Program	-	Contacts, Queuing and Traffic

Abbildung 2: Übreblick über die Routing-Alogorithmen und *knowlege oracles*

festzulegen. Die von den *Oracles* zur Verfügung stellenden Informationen werden von diesen Algorithmen unterschiedlicher Weise benutzt. Wir werden im Grunde 3 Algorithmenklassen unterscheiden.

- *Algoritihmen mit Zero Knowledge:* Sie benutzen nicht die von *Oracles* zur Verfügung gestellten Informationen. Es wird also erwartet, dass diese Algorihmentype unsere Tests schlecht schneidet.
- *Complete knowlege*: Sie benutzen Information der sämtlichen oben genannten *Oracles* (*contact*, *traffic* and *queuing*).
- *Partial knowledge*: Er berechnet die Route ohne den *Traffic Demand Oracle* zu benutzen. Er benutzt aber alle anderen *Oracles*.

Anhand dieser Kenntnisse werden wir nun die Routing-Technik vorstellen, die wir betrachten werden.

3.2 Routing mit *zero knowledge*

: Wie oben beschrieben wurde, benutzen die Algorithmen dieser Klasse keine *Oracles*-Informationen. Wie man es aus der Abbildung 2 entnehmen kann, ist der Alogorithm *First Contac*t (**FC**) ein Beispiel aus dieser Klasse. Der **FC** wählt zufällig unter den von ihm erreichbaren Knoten einen aus und leitet dann zu diesem die Pakette weiter. Falls keinen Knoten erreichbar ist, wird er solange warten, bis einen davon erreichbar wird.
Die Anwendung des **FC**-Alogorithmus garantiert nicht eine erfolgreiche Weiterleiung der Datenpakette bis zum Ziel, denn der nächste Knoten wird zufällig ausgeählt, also ohne Rückicht auf die Topologie und den aktuellen Zustand des Netzes. Der ausgewählte Knoten könnte also kein Pfad bis zum Zielknoten haben oder ein toter Knoten sein (Knoten, die keinen Bezug mit anderen Knoten im Netz haben). Der **FC** benötigt aber nur Information über das lokale Netz und ist ganz trivial zu implementieren.

3.3 Routing mit *Partial Knowledge*

Wie man aus dem Name entnehmen kann, benutzt die Strategie nicht alle verfügbaren *Oracles* bzw. der *Traffic Demand Oracles* wird in dieser Klasse nicht verwendet. Also die Datenpakette werden hier ungeachtet der Datenverkehr im Netz weitergeleitet. Je nach verfügbaren *Oracles* wird jedem Knoten bestimmte Kosten zugewiesen, um die Verzögerungszeit in diesem Knoten einzuschätzen. Die Kostenzuweisung erfolgt über eine Art von *minimum-cost* Pfad. Diese Kosten kann aus unterschiedlichen Faktoren wie Zeit oder die länge des Wegs berechnet werden. Ein Nachteil ist die Tatsache, dass nur ein einziger

Weg berechnet wird, dies ist nicht unbedingt optimal in einem **DTN**, denn in einem **DTN**-Netz je mehr es Wege gibt desto großer ist die Chance, dass die Datenpakette ans Ziel kommen. Da die Länge der Route ein ziemlich reletive Angabe ist, wird nur Kosten in Betracht ziehen, die zeitabhängig sind. Ausserdem lässt sich die Performance eines Routing-Algorithmus besser über die Zeit als über die zurückgelegte Route der Datenpakette im Netz messen. Die Hauptaufgabe ist also der beste Weg zu finden, um diese Zeit möglich geringt zu halten.

Nicht zeitgebundene Algorithmen:
In der Graph-Theorie ist der Dijkstra's Algorithmus das bekannteste Verfahren der kürzeste Weg zwischen 2 Punkte zu bestimmen. Wir werden in den nächsten Punkte sehen dass, eine blinde Anwendung dises Algorithmus keine effektive Route berechnen kann. Als Beispiel für die Anwendung von Algorithmen mit zeitinvariant haben wir den *Minimum Expected Delay* (*MED*), der der *Contacts Summary Oracles* verwendet, wie man es auch in der Tabelle 2 ablesen kann. Die Kosten hier setzen sich aus der durchschnittlichen Wartezeit eines Knoten. Der *MED* versucht stets diese Zeit zu minimieren.

Input: $G = (V, E)$, s, T, $w(e, t)$
Output: L
1: $Q \leftarrow \{V\}$
2: $\mathrm{L}[s] \leftarrow 0$, $\mathrm{L}[v] \leftarrow \infty \; \forall \; v \in V \; s.t \; v \neq s.$
3: **while** $Q \neq \{\}$ **do**
4: Let $u \in Q$ be the node s.t $\mathrm{L}[u] = \min_{x \in Q} \mathrm{L}[x]$
5: $Q = Q \leftarrow \{u\}$
6: **for** each edge $e \in E$, s.t. $e = (u, v)$ **do**
7: **if** $\mathrm{L}[v] > (\mathrm{L}[u] + w(e, \mathrm{L}[u] + T))$ **then**
8: $\mathrm{L}[v] \leftarrow \mathrm{L}[u] + w(e, \mathrm{L}[u] + T)$
9: **end if**
10: **end for**
11: **end while**

Abbildung 3: Modifizierter Dijkstra's Algorithmus

Zeitgebundene Algorithmen:
Im gegensatz zu dem oben genannten Algorithmus, wollen wir in diesem Teil die Kosten von Algorithmen berechnen, die zeitgbunden sind. Der Dijkstra's Algorithmus wäre auch hier eingebracht. Allerdings müssen wir in diesem Fall die Faktorzeit in Betracht ziehen. Also werden wir aus diesem Basis und die Kenntnisse über das Netz einen modifizierten Dijkstra's Algorithmus einführen, der in einem **DTN**-Netz eingesetzt werden kann. Die Pseudo-Kode dieser modifizierten Version ist in der Abbildung 3 illstruiert. In den Eingabeparametern haben wir hier zusätzlich zu einem normalen Dijkstra's Algorith-

mus die Funktion $\mathtt{w}(\mathtt{e},\mathtt{t})$ hinzugenommen. Die Funkion $\mathtt{w}$ zieht also in Betracht die Zei ($\mathtt{t}$), bis die Nachricht beim dem betroffenen Knoten eintrifft. Für nicht zeitgebundene Algorithmen muss man lediglich die Funktion $\mathtt{w}(\mathtt{e},\mathtt{t})$ durch $\mathtt{w}(\mathtt{e})$ ersetzen. Also die Funktion $\mathtt{w}(\mathtt{e},\mathtt{t})$ berechnet die Verzögerungszeit ($\mathtt{t}$) beim Senden einer Nachricht von einem Knoten $\mathtt{a}$ zu einem Knoten $\mathtt{b}$ und ($\mathtt{e}$) ist die gebildete Kante. Die $\mathtt{w}$ Funktion hängt also von den Knoten $\mathtt{a}$ und $\mathtt{b}$ und auch von der Zeit ab. Ausserdem ist die große der zu versendeten Nachricht zu beachten, denn die Verzögerungszeit (t) in der Funktion $\mathtt{w}$ davon abhängt. Also die Funktion $\mathtt{w}(\mathtt{e},\mathtt{t})$ könnte noch wie folgt umgeschrieben werden: $\mathtt{w}(\mathtt{e},\mathtt{t}) = \mathtt{w}'(\mathtt{e},\mathtt{t},\mathtt{m},\mathtt{s})$, wobei $\mathtt{e}$ die Kante von $\mathtt{a}$ nach $\mathtt{b}$ ist, $\mathtt{t}$ die Berechnungszeit, $\mathtt{m}$ die Große der Nachricht und $\mathtt{s}$ der Knoten ist, der den Dijkstra's Algorithmus aufruft. Wir defnieren nun die Funtion $\mathtt{w}'$ folgender Maße $\mathtt{w}'(\mathtt{e},\mathtt{t},\mathtt{m},\mathtt{s}) = \mathtt{t}'(\mathtt{e},\mathtt{t},\mathtt{m},\mathtt{s}) - \mathtt{t} + \mathtt{d}(\mathtt{e},\mathtt{t}')$ wobei $\mathtt{t}'(\mathtt{e},\mathtt{t},\mathtt{m},\mathtt{s}) = \mathtt{min}\{\mathtt{t}'' | \int_{\mathtt{x}=\mathtt{t}}^{\mathtt{t}''} \mathtt{c}(\mathtt{e},\mathtt{x})\mathtt{dx} \geq (\mathtt{m} + \mathtt{Q}(\mathtt{e},\mathtt{t},\mathtt{s}))\}$. Die Funktionen $\mathtt{c}(\mathtt{e},\mathtt{t})$ bzw. $\mathtt{d}(\mathtt{e},\mathtt{t})$ sind Eingabe von dem *Contact Oracles*. Sie bezeichnen die Kapazität bzw. die Verzögerungszeit für die **DTN** Topologie. Anderseits bezeichnet die Funktion $\mathtt{Q}(\mathtt{e},\mathtt{t},\mathtt{s})$ die Größe des Stacks an der Quelle der Kante ($\mathtt{e}$) in der Zeit ($\mathtt{t}$) vorgesehen vom Knoten ($\mathtt{s}$).

Aus der Tabelle in der Abbildung 2 können wir folgende Algorithmen als Beispiel für diese Klasse entnehmen:

- *Early Delivry*(**ED**):Dieser Algortihmus funktionniert wie einen *source routing* Algorithmus. Also die Route wird an der Quelleknoten berechnet und festgelegt. Bei der Routeberchnung wird die Stackkapazität der dazwischen liegenden Knoten missgeachtet. Also wird die Funktion $\mathtt{Q}(\mathtt{e},\mathtt{t},\mathtt{s}) = \mathtt{0}$. Dieses Verfahren kann natürlich dazu führen, dass Daten aufgrund der limitierten Speicherkapazität der Zwischenknoten verworfen werden (*buffer overflow*). Der **ED**-Algorihmus ist aber optimal, wenn die selektierten Knoten auf dem Pfad genügt Speicherkapizität zur Verfügung haben, um die gesamte Nachricht(Daten) aufzunehmen.
- *Early Delivry with Local Queuing* (**EDLQ**): Das Verfahren ist fast identisch wie beim **ED**, aber hier wird die Speicherkapazität der dazwinchen liegenden Knoten in Betracht gezogen. Ausserdem wird die Route für jeden Knoten auf dem Pfad berechnet. Also statt der *source routing* wie beim **ED** wird hier den *per hop-routing* verwendet. Also hier wird der *Contacts Oracles* benutzt.
- *Earliest Delivry with All Queues* (**EDAQ**): Wie beim **ED** wird hier den *source routing*. Die Funktion $\mathtt{Q}$ zieht in Betracht den Stackzuszand auf jedem Knoten. Nach jeder Routeberechnung soll die Kantekpazität der laufigen Kanten auf jedem Knoten reserviert werden.

3.4 Routing mit *Complete Knowledge*

Bishr haben wir Algorithmen betrachten, die gar keine bzw. partielle Kenntnisse über das gesamte Netz Gebrauch gamacht haben. Wir wollen in diesem Teil Algerithmen betrachten, die die sämtlichen oben definierten *Oracles* benutzen. Also wir werden den *Traffic Demand Oracle* verwendet, damit werdenden Überblick die Datenverkehr im Netz übersichtlicher. Wir werden hier eine *Linear Programming*(**LP**) Formullierung vorstellen, die eben alle von *Oracles* zur Verfügung gestellte Informationen benutzt, um eine optimale Route zu berechnen. Unter „optimale Route", soll man verstehen der beste Weg um die maximale Miniminierung der Verzögerungszeit im Netz zu erreichen.

4 Performenzevaluierung

Ein Simulator wurde implementiert [5] um eine **DTN**-umgebung zu simulieren, dies um die bisherigen vorgestellten Algorithmen auf Performance zu prüfen. Wir betrachten hier das *remote village* Szenario wie es in der Abbildung 4 dargestellt wird.

Das *remote village* ist ein Projekt, dass unter dem Name *Wizzy Digital Courier* in

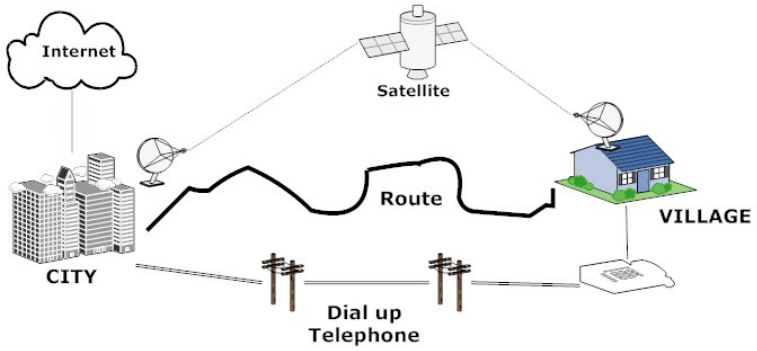

Abbildung 4: Remote village

Süd-Afrika durchgeführt wurde. Das Internet ist in dieser Umgebung nur zeitlich verfügbar. Das Szenario besteht aus 3 Darsteller4. Es gibt einen Motorradfahrer (Nr. 1) (einen Kurier), der ein paar Mal pro Tag von der Stadt ins Dorf fährt (5min. Fahrzeit), er besitzt einen USB-Massenspeicher(128MB), der permanent über eine *high-speed* Wireless-Internetverbindung mit etwa 1Mbps verfügt. Das Dorf ist ausserdem über ein Dialup (Nr. 2) und ein Satellit (Nr. 3) mit Internet verbunden. Der Satellite hat eine

niedriege Bandbreite (etwa 10Kbps) und fliegt ungefähr 4 bis 5 Male pro Tag über das Dorf. Für unsere Tests können wir der Motofahrer als ein high-bandwidth, der Dialup-Verbindung als low-bandwidth und der Satellite als ein moderate-bandwidth darstellen.

- Daten vom Dorf zur Stadt etwa 1KB
- Daten von der Stadt zum Dorf etwa 10KB
- 2 Datenverkehrrate: 200/Tag (*low load*) und 1000/Tag (*high load*). Diese Daten werden in unterschiedlichen Zeitpunkt ins Netz eingeführt. Das Experiment wird als beendet erklärt, wenn alle diese Daten an ihren Ziele ankommen werden.

Jetzt wollen wir untersuchen, wie die einzelnen Routing-Algorithmen Gebrauch von der 3 verfügbaren Klassen (Dialup, Satellit, Motofahrer) machen werden, um die eingeührten Datenpakette an das richtige Ziele weiterzuleiten. Wirhaben in diesem Paper nur ein Teil der Ergebnisse berichtet, allerdings die detaillierten Resultate können hier [5] entnehmen werden.
Die Ergebnisse davon wird in der Abbildung 5 und 6. Unter anderen können wir leicht entnehmen, dass der Algorithm **MED** ausschließlich die Dialup-Verbindung in der *low*- und *high*-Phase benutzt, um die Daten weiterzuleiten. Denn der *MED* benutzt ein *source routing* Algorithmus. Also die Route wird schon an der Quelleknoten berchnet und festgelegt und angesichts der Tatsache, dass die Dialup-Verbindung mehr verfügbar als die anderen Verbindungsmöglichkeiten ist, wird diese in der einmaligen Routeberechnung einbezogen.

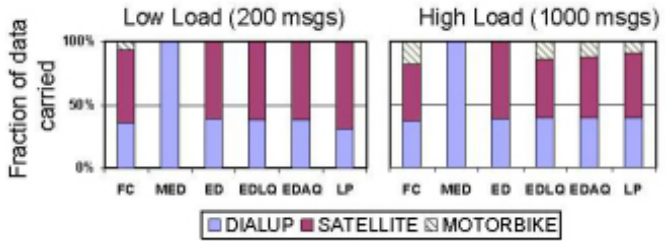

Abbildung 5: Datenverkehr bei unterschiedlichen Verbindungstyp

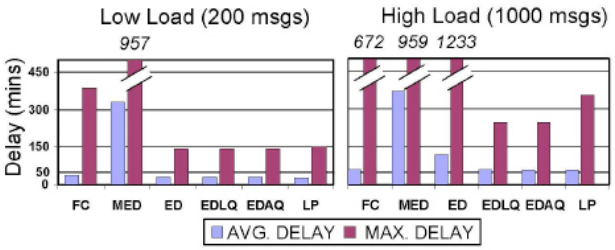

Abbildung 6: Vergleich der Verzögerungszeiten

5 Fazit

DTN sind Netwerke die eine große Verzögerungszeit annehmen. Diese Zeit kann sich auf Stunden, sogar Tagen erstrecken. Die üblichen Netzwerkprotokolle, die wir in gewöhnlichen Netwerke kennen, können in dieser Umgebung nicht angewandt werden.
In den letzten Jahren ist das **DTN**-Problem immer wichtiger geworden. In Umgebung mit ständigen getrennten Netzverbindungen können die Verbindungsmöglichkeiten vorhergesagt werden. Diese Idee war der Ausgangpunkt von zahlreichen Projekten, wie z.B das *Wizzy Digital Courier*. Wir haben in diesem Paper verschiedene Algorithmen eingeführt, die auf diese Idee basiert sind und haben ein Framework für die Evaluierung diese Routing-Algorithmen in einer **DTN**-Umgebung implementiert. Das Experiment wurde in einem Dorf in Süd-Afrika im Rahmen des Projekts *Wizzy Digital Courier* durchgeführt. Wir haben festgestellt, dass die globale Kenntnisse über das Netz keine Voraussetzung für die eine effiziente Routeberechnung ist. Ausserdem kamm auch heraus, dass die Algorithmen wie **EDLQ**, **EDAQ** und **LP** gute Kandidate sind, um mit den Netzstörungen und der Ressourcenknappheit umzugehen. Aus den Ergebnissen könnten wir auch merken, dass für Netze mit zahlreichen Verbindungen (die Ressourcen sind vorhanden) ein triviale Routing-Technik ausreichend ist, um die Datenpakette weiterzuleiten.

Literatur

[1] Jean Patrick Gelas. Reseaux Interplanetaires et Reseaux tolérants au delais, 2003.

[2] http://www.dtnrg.org/. Delay Tolerant Networking Research Group.

[3] http://www.wizzy.org.za. Wizzy Digital Courier, 2003.

[4] Tayeb LEMLOUMA. Routage dans les reseaux mobile Ad Hoc.

[5] R. Patra S. Jain, K. Fall. Routing in a Delay Tolerant Network, 2004.

[6] Forrest Warthman. Delay Tolerant Notworks (DTNs), 2003.

[7] Wikipedia. http://de.wikipedia.org.