Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 31

Adaption Algorithms for Mobile Traffic Offloading

Dominik Olp

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	M. Sc. Patrick Baier
begonnen am:	8. Oktober 2012
beendet am:	27. März 2013
CR-Klassifikation:	C.2.1, C.2.4, C.5.3

Kurzfassung

Der durch Smartphones generierte Datenverkehr in Mobilfunknetzen steigt seit Jahren unablässig an. Dazu trägt sowohl die steigende Verbreitung der Smartphones, als auch die immer größer werdenden Bandbreitenanforderungen z.B. durch Audio- oder Videostreaming bei. Um den Zuwachs an nötiger Bandbreite zu reduzieren, soll entstehender Traffic, von dem Mobilfunknetz auf lokale WLAN Kommunikation verschoben werden. Eine besondere Art dieser Auslagerung bezeichnet man als „Opportunistic Traffic Offloading“. Dabei tauschen benachbarte Smartphones beispielsweise per WLAN Daten aus, wenn sie sich begegnen. Im Laufe dieser Arbeit werden Algorithmen vorgestellt, die anhand der Positionsdaten von Smartphones besonders günstige Ziele, das heißt Smartphones mit einer hohen Anzahl an Begegnungen, bestimmen. Diesen Zielen kann per Mobilfunk die entsprechende Nachricht übermittelt werden, woraufhin sie selbstständig die Nachricht per WLAN verbreiten. Eine hohe Zahl an Begegnungen fördert dabei die schnelle Verbreitung von Nachrichten, da nach jeder Begegnung ein weiteres Smartphone in der Lage ist die Nachricht weiter zu verteilen. Weiterhin wird gezeigt, wie mit dem periodischen Versenden einer Nachricht per Mobilfunk und dem Anwenden der beschriebenen Algorithmen, im Vergleich zum reinen Versenden per Mobilfunk, Traffic-Einsparungen von durchschnittlich 40 bis 50% möglich sind.

INHALTSVERZEICHNIS

1 Einleitung **9**
- 1.1 Problembeschreibung . . . 9
- 1.2 Lösung . . . 10
- 1.3 Einsatzmöglichkeiten . . . 12
- 1.4 Gliederung der Arbeit . . . 13

2 Hintergrund und verwandte Arbeiten **14**
- 2.1 Grundlage . . . 14
- 2.2 Weitere . . . 14
  - 2.2.1 Objective Function . . . 14
  - 2.2.2 Mobilitätsvorhersage/Bandbreitenvorhersage . . . 15
  - 2.2.3 Target Set Selection . . . 16
  - 2.2.4 Energieeffizienz . . . 17
  - 2.2.5 Räumliche Verlagerung . . . 17

3 Systemmodell **18**
- 3.1 Definitionen . . . 18
- 3.2 Kommunikation . . . 18
  - 3.2.1 Interfaces . . . 18
  - 3.2.2 Kommunikationspartner . . . 19
  - 3.2.3 Delay Tolerance . . . 20
- 3.3 Clients . . . 20
  - 3.3.1 Bewegung . . . 20
  - 3.3.2 Voraussetzung . . . 20

4 Adaptive Verteilung **22**
- 4.1 Grundlagen . . . 22
  - 4.1.1 Regulärer Ablauf (ohne adaptiver Verteilung) . . . 22
  - 4.1.2 Regulärer Ablauf (mit adaptiver Verteilung) . . . 22
  - 4.1.3 Beispiel . . . 27
  - 4.1.4 Zielclients und fehlende Clients . . . 28
  - 4.1.5 Nächste Clientposition . . . 28

4.2 Nachrichtentypen . . . 29
4.2.1 Allgemeines . . . 29
4.2.2 Serverseitig . . . 29
4.2.2.1 LocationUpdateInquiry . . . 30
4.2.2.2 Request . . . 30
4.2.3 Clientseitig . . . 30
4.2.3.1 Acknowledge (ACK) . . . 30
4.2.3.2 LocationUpdate . . . 30
4.3 Verteilungsstrategien . . . 31
4.3.1 RANDOM . . . 31
4.3.2 DISTANCE . . . 31
4.3.2.1 Funktion . . . 32
4.3.2.2 Problem . . . 32
4.3.3 DISTANCE_ENHANCED . . . 33
4.3.3.1 Funktion . . . 34
4.3.3.2 Problem . . . 35
4.3.4 DISTANCE_MOTION . . . 36
4.3.4.1 Funktion . . . 36
4.3.5 DISTANCE_COMBINED . . . 38
4.3.5.1 Funktion . . . 38
4.3.6 SAME_STREET . . . 39
4.3.6.1 Funktion . . . 39
4.4 Parameter . . . 41
4.4.1 Adaptive message inquiry . . . 41
4.4.2 Reinjection Threshold . . . 42
4.4.3 Radius reduction . . . 43
4.4.4 Vector Extension . . . 44

5 Evaluierung **45**
5.1 ONE Simulator . . . 45
5.2 Simulation . . . 45
5.2.1 Aufbau . . . 45
5.2.2 Parameterkombinationen . . . 46
5.3 Erwartungen . . . 48
5.4 Vergleich . . . 49
5.4.1 RANDOM . . . 49
5.4.2 DISTANCE . . . 51
5.4.3 DISTANCE_ENHANCED . . . 53
5.4.4 DISTANCE_MOTION . . . 55
5.4.5 DISTANCE_COMBINED . . . 57
5.4.6 SAME_STREET . . . 59
5.5 Fazit . . . 61
5.5.1 Ergebnisse . . . 61

6 Ausblick **64**
6.1 Strategien 64
6.1.1 DISTANCE_ENHANCED 64
6.1.2 Panic Zone 65
6.1.3 Einsatzmöglichkeit Verkehrsinformation 65
6.2 Privacy 66
6.3 Datenübertragung 66

Literaturverzeichnis **67**

Abbildungsverzeichnis

1.1 Normale Nachrichtenverteilung 10
1.2 Beispiel einer Ad-Hoc Verbreitung 11

4.1 Regulärer Ablauf 26
4.2 Zeitlicher Ablauf 27
4.3 Gleichung zur Berechnung der Distanzmatrix. 32
4.4 DISTANCE_ENHANCED Beispiel 33
4.5 DISTANCE_ENHANCED Baumdarstellung 35
4.6 DISTANCE_MOTION Schnittpunkte vorhanden 37
4.7 DISTANCE_MOTION Kein Treffer 38
4.8 SAME_STREET Bewegung in entgegengesetzter Richtung 40
4.9 SAME_STREET Bewegung in gleicher Richtung 41

5.1 RANDOM Reinjection Threshold Vergleich 50
5.2 RANDOM Clientanzahl Vergleich 50
5.3 DISTANCE Reinjection Threshold Vergleich 52
5.4 DISTANCE_ENHANCED Radius Reduction Vergleich 54
5.5 DISTANCE_MOTION Adaptive Message Inquiry Vergleich 56
5.6 DISTANCE_MOTION Vector Extension Vergleich 57
5.7 DISTANCE_COMBINED Radius Reduction Vergleich 58
5.8 SAME_STREET Adaptive Message Inquiry Vergleich 60
5.9 SAME_STREET Reinjection Threshold Vergleich 61
5.10 Vergleich der Überdeckungswerte 63
5.11 Vergleich der nötigen Positionsnachrichten 63

KAPITEL 1

EINLEITUNG

Dieses Kapitel beschreibt generell die Problemstellung, sowie den umgesetzten Lösungsansatz und erläutert die Gliederung der weiteren Arbeit.

1.1 Problembeschreibung

Durch die immer größere Verbreitung von Smartphones und anderen internetfähigen Mobilgeräten steigt der Datenumsatz in Mobilfunknetzen unablässig an. Durch die steigende Nachfrage nach datenintensiven Diensten (wie Videos oder Onlinespiele für Mobiltelefone) ist die vorhandene Bandbreite der UMTS-Netze, vor allem in urbanen Regionen, stark ausgelastet. Eine Lösung wäre der Aufbau von neuen UMTS-Sendemasten, um mehr Bandbreite zur Verfügung zu stellen. Außerdem wäre es möglich an stark ausgelasteten Stellen (z.B. in Städten) WLAN Acces Points aufzustellen, um Nutzer dazu zu bewegen statt des UMTS Netzes, das schnellere lokale WLAN zu verwenden (siehe auch: [SK12]). Da beide Ansätze aber mit hohen Kosten einhergehen, suchen Netzbetreiber (siehe auch [HHK+12]) unter anderem nach Möglichkeiten, ohne bestehende Hardware auszubauen, den Datenverkehr zumindest partiell aus den UMTS- bzw. 3G-Netzen auszulagern. Anstatt die Infrastruktur zu optimieren, um mit dem höhreren Datenverkehr zurechtzukommen, soll der Datenverkehr also reduziert werden. Das ist auch die Idee dieser Arbeit.

Beispiel Angenommen es existieren fünf Empfänger *A*-*E* (z.B. Smartphones) in einer UMTS-Zelle. Alle nutzen die gleiche Anwendung um ein Video zu empfangen und anzuschauen. Zeitgleich möchten alle fünf Empfänger dasselbe Video anschauen (beispielsweise eine Übertragung eines Fußballspiels). Ohne Optimierungen müsste der Mobilfunkanbieter allen fünf Empfängern (*A*-*E*) das Video separat zuschicken. Das bedeutet, dass fünf Mal exakt dieselbe Nachricht an fünf verschiedene Empfänger zugestellt werden muss (siehe dazu Abbildung 1.1).

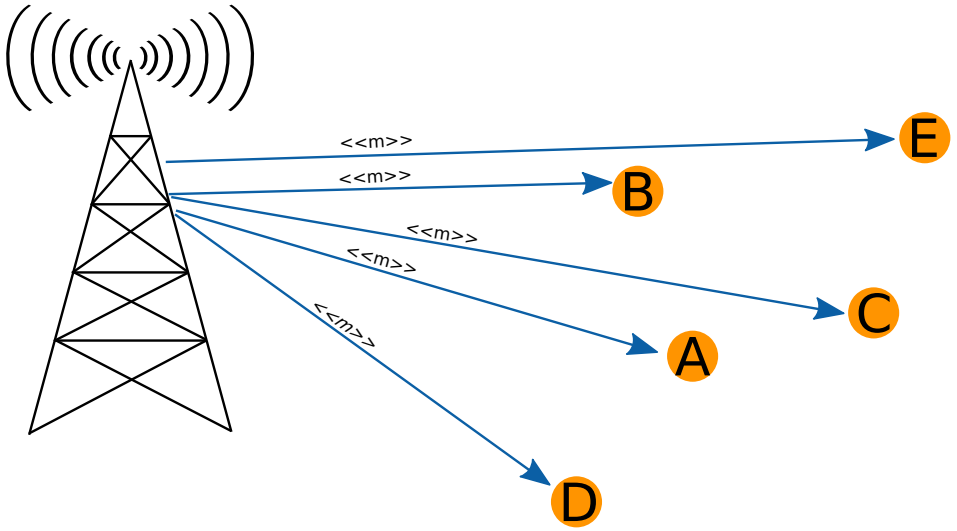

Abbildung 1.1: Beispiel einer normalen Nachrichtenverteilung. Jeder Empfänger erhält die Nachricht $<< m >>$ per Mobilfunk.

1.2 Lösung

Die Idee zur Reduzierung des Datenverkehrs ist, dass Nachrichten die mehrere Empfänger erreichen sollen nicht pro Empfänger ein Mal verschickt werden, sondern optimalerweise nur ein einziges Mal für alle Empfänger. Dies soll dadurch erreicht werden, dass jeder Empfänger weiß, wenn andere Empfänger in seiner Nähe sind und die eingehenden Nachrichten dann an diese weiterleitet. Der hier beschriebene Lösungsansatz basiert auf der Technik des „Opportunistic Traffic Offloading“. Dabei übernehmen die Empfänger die Aufgabe, eingehende Nachrichten weiterzuleiten. Das heißt, falls ein Empfänger die von ihm angefragten Daten gesendet bekommt (beispielsweise den Inhalt einer Nachrichtenwebseite, ein Video oder ähnliches), schickt er diese an umliegende Empfänger weiter, sodass an diese nicht erneut dieselben Daten per Mobilfunk geschickt werden müssen. Voraussetzung für diese Technik ist allerdings, dass in einem bestimmten Gebiet mehrere Geräte dieselben Daten empfangen sollen. Im optimalen Fall sendet der Mobilfunkanbieter also eine Datennachricht nur an einen Empfänger. Der verteilt diese daraufhin per Ad-Hoc Netzwerk (beispielsweise Bluetooth oder WiFi Direct) an alle Geräte in seiner Umgebung. Jeder Empfänger der die Nachricht auf diesem Weg erhält leitet sie ebenso weiter. Nach einer gewissen Zeit besteht so die Möglichkeit, dass auch Geräte die auf die Nachricht warten diese per Ad-Hoc Netzwerk durch ein benachbartes Gerät erhalten. Im Idealfall müsste die Nachricht nur ein einziges Mal von dem Mobilfunkanbieter versandt werden, was dann sowohl Bandbreite als auch Kosten spart. Diese Art der Verteilung funktioniert allerdings nur mit Daten, die mit gewissen Latenzen versandt werden können (auch „delay tolerant“ genannt).

Zur Veranschaulichung siehe Abbildung 1.2. In dieser Abbildung erhält Empfänger A (durchgezogener Übertragungskreis) die Nachricht per Mobilfunk und leitet diese zunächst

an die Geräte weiter, die sich direkt in seinem Übertragungsradius aufhalten (*B*, *C* und *D*). Nachdem Empfänger *C* die Nachricht erhalten hat, wird diese wiederum automatisch an alle überdeckten Geräte weiter versendet (in diesem Fall an *E*).

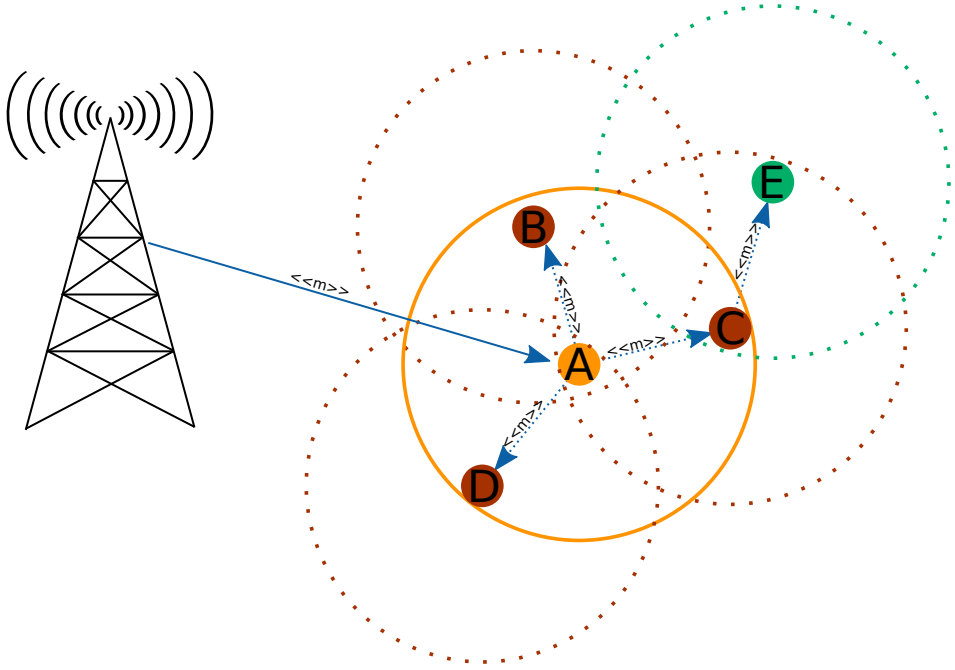

Abbildung 1.2: Beispiel einer Ad-Hoc Verbreitung. Der Kreis um einen Empfänger entspricht seinem Übertragungsradius. Empfänger A erhält Nachricht $<< m >>$ und initiiert die weitere Verbreitung.

Der wichtigste Punkt beim Versenden der Datennachricht zu einem Empfänger ist, den am meisten Erfolg versprechenden Empfänger zu finden. Das heißt, es muss das Gerät gefunden werden, das eine möglichst große Verbreitung der Nachricht initiieren kann. Dazu wurden Algorithmen entwickelt, die mit Hilfe der Positionen und Bewegungsrichtungen zu bestimmen versuchen, welcher Empfänger als aktuelles Ziel am besten geeignet ist (siehe auch 4 Adaptive Verteilung). Zur Evaluation der Algorithmen wurde außerdem eine Zufallsverteilung implementiert, die keine Daten (wie Position oder Bewegungsrichtung) der Geräte nutzt, sondern per Zufall ein neues Ziel bestimmt.

Einschränkung Eine Reduzierung des Datenverkehrs ist also nur dann möglich, wenn mehrere Empfänger (z.B. Smartphones) die gleichen Daten erwarten. Diese werden dann unabhängig vom Mobilfunknetz zwischen den Empfängern mittels Funkverbindungen (wie Bluetooth oder WLAN) ausgetauscht. Da weiterhin jeder Empfänger die von ihm angeforderten Daten erhalten muss, kann kein Datenverkehr eingespart werden, wenn jede angeforderte Nachricht einzigartig ist. Dieser Lösungsansatz zielt nicht darauf ab, dauerhaft den verursachten Traffic im Mobilfunknetz zu senken, sondern in Spitzenlastzeiten

den Datenverkehr etwas zu reduzieren. Eine solche Spitzenlastzeit wäre zum Beispiel der Berufsverkehr morgens und abends, wenn viele Personen auf dem Weg zur bzw. von der Arbeit räumlich relativ eng in Bussen und Bahnen oder auf der Straße zusammen sind. Auch größere Menschenansammlungen beispielsweise bei Großveranstaltungen wären denkbar. Ist die Empfängerdichte (Anzahl Empfänger pro Fläche) zu gering, wie zum Beispiel nachts, kann dieser Lösungsansatz nicht funktionieren, das heißt eine dauerhafte Senkung ist nicht möglich.

1.3 Einsatzmöglichkeiten

Daraus resultiert, dass diese Art der Datenverteilung nur möglich ist, wenn der Ankunftszeitpunkt nicht fix ist, sondern in einem begrenzten Zeitfenster liegt und der Inhalt nicht vertraulich ist. Es gibt eine Reihe von Einsatzmöglichkeiten, die diese Punkte erfüllen.

Zunächst seien dafür Nachrichten-Apps genannt. Viele Smartphonenutzer lesen regelmäßig Nachrichten mit Hilfe ihres Mobiltelefons. Eine typische Verteilung fände dann beispielsweise auf dem Weg zur Arbeit (z.B. in der S-Bahn) statt. Fordert die App eines Smartphones Updates von einer Nachrichtenseite an, werden diese automatisch nach dem Empfang an umliegende Geräte mittels eines Ad-Hoc Netzwerks weitergeleitet. Fordert nun ein zweiter, benachbarter Nutzer mit der gleichen App neue Nachrichten an, so kann dieser auf die zuvor per Ad-Hoc Netzwerk empfangenen Daten zugreifen. Damit wurde das erneute Senden der Daten per Mobilfunk durch die Ad-Hoc Verteilung überflüssig. Hat ein Gerät Daten von einem benachbarten Gerät empfangen, muss daraufhin lediglich überprüft werden, ob serverseitig zwischenzeitlich neue Daten zum Herunterladen bereitstehen. Diese müssen dann unter Umständen per Mobilfunk versendet werden, können aber wiederum per WLAN oder Bluetooth an weitere Empfänger verteilt werden.

Eine weitere Stelle an der man Traffic Offloading verwenden kann, ist die Verbreitung von Verkehrsdaten [1]. So könnten zum Beispiel Autos in einer Stadt untereinander empfangene Verkehrsnachrichten austauschen, um die Fahrer über mögliche Gefahrenquellen oder Staus zu informieren. Die Reichweite und Bandbreite von WiFi Direct würde dabei ausreichen, um auch Fahrzeugen auf anderen Spuren (z.B. auf Schnellstraßen) Nachrichten zu übermitteln. Die relativ hohe Bewegungsgeschwindigkeit fördert dabei auch eine schnellere Verbreitung der Daten. Dabei müssten entsprechende Nachrichten sicherlich relativ klein gehalten werden, um den kurzen Begegnungszeiten Rechnung zu tragen. Um innerstädtische Verkehrsprobleme zu beschreiben, sollten jedoch wenige hundert Kilobyte genügen. Folgende Informationen würden die Meldung zu einem Verkehrsproblem ergeben, wobei mehrere Meldungen in einer Nachricht gesendet werden können (mit Größenabschätzungen):

- Betreffende Straße (ID statt Straßennamen: 8Byte)
- Kilometerangabe der Straße (4Byte)

[1] Idee aus „Relieving the wireless infrastructure“[WAL+11] S. 2 Punkt II (3. Satz)

- Code um Problem zu beschreiben wie Stau, Unfall, Stop-and-go, Eis, etc... (4Byte)
- Zeitstempel zu dem das Problem zum letzten Mal berichtet wurde (8 Byte)

Kombiniert man nun empfangene Verkehrsnachrichten mit einem im Auto verbauten Navigationssystem, kann der Fahrer automatisch auf Staus oder gefährliche Glatteisstellen hingewiesen werden. Zusätzlich ist anzumerken, dass sinnvolle Latenzen bei diesem System zunächst bestimmt werden müssen. Die Gültigkeitsdauer einer Nachricht sollte die in dieser Arbeit verwendete Latenz von wenigen Minuten (z.B. 10 bis 15) vermutlich nicht übersteigen, da bei Verkehrsdaten die Aktualität eine große Rolle spielt.

1.4 Gliederung der Arbeit

Kapitel 1 enthält die Einleitung, die Problembeschreibung sowie die Lösungsansätze und den Überblick über die Gliederung.

Kapitel 2 zeigt welche anderen Arbeiten sich mit diesem oder einem ähnlichen Themengebiet beschäftigen.

Kapitel 3 beschreibt das zugrunde liegende Systemmodell der weiteren Arbeit.

Kapitel 4 beschreibt die entwickelten Verteilungsalgorithmen sowie die dafür notwendigen Nachrichtentypen und Parameter.

Kapitel 5 erläutert die Ergebnisse der durchgeführten Simulationen.

Kapitel 6 zeigt welche weiteren Ideen existieren um beispielsweise die Ad-Hoc Verteilung zu verbessern.

KAPITEL 2

HINTERGRUND UND VERWANDTE ARBEITEN

Dieses Kapitel zeigt, welche anderen Arbeiten sich mit diesem oder einem ähnlichen Themengebiet befassen.

2.1 Grundlage

Grundlage dieser Arbeit ist eine Veröffentlichung von Patrick Baier, Frank Dürr und Kurt Rothermel [BDR12]. Diese Veröffentlichung beschäftigt sich mit dem Finden einer initialen Gruppe von Empfängern, die am erfolgversprechendsten für die opportunistische Verteilung einer Nachricht ist. Erfolg bedeutet dabei, dass eine möglichst kleine Gruppe von Empfängern eine Datennachricht per Mobilfunk erhält und diese an möglichst viele andere Empfänger im System weiterzuleiten versucht (mittels Ad-Hoc Netzwerken). Dabei wird also nur der erste Schritt der opportunistischen Verteilung betrachtet (dem Finden einer optimalen initialen Gruppe). Diese Bachelorarbeit erläutert aber, wie bereits erwähnt (siehe 1.2 Lösung), den nächsten Schritt, also das periodische Finden eines geeigneten Empfängers.

2.2 Weitere

2.2.1 Objective Function

Eine Arbeit mit einer sehr ähnlichen Zielsetzung wurde unter dem Titel „Relieving the Wireless Infrastructure: When Opportunistic Networks Meet Guaranteed Delays“ veröffentlicht [WAL+11]. Dabei werden allerdings andere Ansätze zur Bestimmung und zur Anzahl der Ziele gewählt. Das Kernstück der Zielauswahl ist dabei die sogenannte „Objective Function“. Das ist eine Funktion die zu jedem Zeitpunkt angeben kann, wie viele Empfänger die Nachricht bereits erhalten haben sollten. Damit lässt sich dann leicht bestimmen, ob eine

Nachricht ein weiteres Mal vom Mobilfunkanbieter verschickt werden muss (und an wie viele Ziele) oder ob bereits ausreichend viele Nachrichten empfangen wurden. Nach der Berechnung der Anzahl der nötigen Ziele kann dann auf eine Reihe verschiedener Methoden zurückgegriffen werden, um ein bestmögliches Ziel zu berechnen (unter anderem auch GPS-basierte und randomisierte Methoden).

Dieser Ansatz wurde in der vorliegenden Arbeit nicht weiter verfolgt, kann aber mit begrenztem Aufwand zur bestehenden Code-Architektur hinzugefügt werden. Die in dem Paper beschriebenen Anwendungsfälle (z.B. Übertragung von Verkehrsdaten) lassen sich allerdings durchaus auf die vorliegenden Arbeit anwenden (siehe auch 1.3 Einsatzmöglichkeiten)

2.2.2 Mobilitätsvorhersage/Bandbreitenvorhersage

Das Wissen über die Mobilität von Empfängern kann sehr nützlich sein, um Daten Ad-Hoc zu übertragen. Beispielsweise können Bewegungsdaten von Empfängern über einen gewissen Zeitraum aufgezeichnet und daraus Vorhersagen generiert werden. Je nach Genauigkeit der Mobilitätsvorhersage und Verhalten der Empfänger, können zukünftige Treffpunkte zwischen den Geräten sehr genau vorhergesagt werden. Das Verhalten der Empfänger ist insofern wichtig, als dass für Empfänger die immer wieder ähnliche Strecken zurücklegen (z.B. der Weg zum Arbeitsplatz, zum Supermarkt, etc.) sehr viel genauere Vorhersagen möglich sind, als für Empfänger die kein festes Bewegungsschema haben. Vorhersagen zur Bandbreite sind dann wichtig, wenn WLAN Hotspots anstelle von Ad-Hoc Übertragungen verwendet werden. Wenn also beispielweise ein Mobilfunkanbieter, um das 3G Netz zu entlasten, an wichtigen Punkten Hotspots erstellt, können Datenübertragungen dort per WLAN erfolgen. Durch das Wissen über die Auslastung eines Hotspots kann dann berechnet werden, wieviel Daten ein Gerät erhalten kann, während er sich innerhalb des Hotspots befindet. Damit kann der Mobilfunkanbieter dann bestimmen, ob ein Senden per UMTS/3G nötig ist, oder ob der entsprechende Empfänger vor Ablauf der Nachrichtenlebenszeit alle nötigen Daten mittels eines Access Points übertragen konnte.

Eine Arbeit, die Mobilitätsvorhersagen nutzt ist „Enhancing Mobile Data Offloading with Mobility Prediction and Prefetching“ von Siris und Kalyvas [SK12]. Dabei werden die von einem Empfänger angefragten Daten in mehreren WLAN Hotspots gecached und gesendet, sobald der anfragende Empfänger sich in Übertragungsreichweite befindet. Dies passiert beispielsweise, wenn der Empfänger mit einem Auto in die Nähe des Hotspots kommt. Die Übertragung der Daten wird über das Mobilfunknetz fortgesetzt, sobald der Empfänger die Sendezone eines Hotspots verlässt. Für den erfolgreichen Einsatz dieser Technik ist also unter anderem ein nahtloser Übergang zwischen beiden Übertragungstechniken (WLAN und Mobilfunk) nötig. Um feststellen zu können, welche Daten in einem Hotspot vorgehalten werden müssen, wird außerdem Wissen über die verfügbaren Bandbreiten des Mobilfunknetzes und der Hotspots vorausgesetzt. So wird zu jedem Zeitpunkt sichergestellt, dass ein Hotspot nur die Daten vorhält, die der Empfänger in dem Moment seiner Ankunft auch benötigt.

Einen sehr ähnlichen Ansatz findet sich auch in der Arbeit von Balasubramanian, Mahajan und Venkataramani [BMV10]. Die Idee dabei ist, dass eine Auslagerung der Daten auf WLAN nur dann angewendet wird, wenn dadurch Traffic im Mobilfunknetz eingespart werden kann. Das heißt, eine längere Nachrichtengültigkeit für die eingehenden oder ausgehenden Daten wird genau dann toleriert, wenn dadurch Traffic im Mobilfunknetz eingespart werden kann. Kann ein Empfänger also einen WLAN Hotspot nicht in einem gewissen Zeitfenster erreichen, wird kein Versuch unternommen auf eine WLAN Verbindung zu warten, sondern sofort über das Mobilfunknetz gesendet. Um bestimmen zu können, ob das Warten auf einen Hotspot sinnvoll ist, wird vorausgesetzt, dass Empfänger die Standorte sowie die Bandbreite der Hotspots kennen.

Wissen über die Mobilität von Empfängern wird auch in der vorliegenden Arbeit vorausgesetzt, allerdings nicht im Sinne der hier beschriebenen Vorhersage. Lediglich Abschätzungen über die wahrscheinlich nächste Position der Geräte sind erforderlich. Dafür müssen keine aufgezeichneten Bewegungsprofile von Emfpängern erstellt und ausgewertet werden. Eine Bandbreitenvorhersage ist ebenfalls nicht nötig, da keine Auslagerung von Daten auf WLAN Access Points erfolgt.

2.2.3 Target Set Selection

Arbeiten auf dem Gebiet der opportunistischen Kommunikation beschäftigen sich oft mit dem sogenannten „Target Set Selection Problem“. Dabei ist die Problemstellung prinzipiell die gleiche wie in dieser Arbeit. Durch das Senden einer Nachricht per UMTS/3G an wenige, ausgewählte Empfänger, soll durch die darauf folgende Verbreitung über Ad-Hoc Netzwerke ein möglichst großer Kreis an Personen die Nachricht erhalten. Das Finden einer optimalen, initialen Gruppe von Empfängern (dem „Target Set“) ist dabei sehr komplex und extrem rechenintensiv. Optimal bedeutet dabei, dass eine möglichst kleine Gruppe gefunden werden soll, die bis zum Ende der Nachrichtenlaufzeit allen anderen Empfängern im System die Nachricht weiterleiten kann.

Algorithmen zum Finden eines möglichst optimalen Target Sets werden unter anderem in der Veröffentlichung „Mobile Data Offloading through Opportunistic Communications and Social Participation“ [HHK+12] diskutiert. Dabei nutzen die Autoren in erster Linie einen heuristischen Ansatz, der voraussetzt, dass die Bewegungen der Empfänger bereits teilweise bekannt sind. Das heißt, es wurden Bewegungsdaten aufgezeichnet und aufgrund der Wiederholungen darin (z.B. wenn eine Person immer zur gleichen Uhrzeit auf dem gleichen Weg geht) können zukünftige Treffen zum Austausch von Daten vorhergesagt werden. Damit fällt die genannte Veröffentlichung unter anderem auch in die Kategorie 2.2.2 Mobilitätsvorhersage/Bandbreitenvorhersage. Der wesentliche Unterschied zwischen der erwähnten Arbeit und dieser Arbeit ist, dass in dieser Arbeit ein Target Set nicht relevant ist. Die möglichst hohe Anzahl an Ad-Hoc Übertragungen wird nicht durch eine initiale Gruppe, sondern durch regelmäßige Verbreitung an ein einzelnes Ziel erreicht.

2.2.4 Energieeffizienz

Laut der Veröffentlichung „Energy Efficient Offloading of 3G Networks“ von Ristanovic, Boudec, Chaintreau und Erramilli [RLBCE11] kann die opportunistische Verteilung von Nachrichten auch zur Erhöhung der Energieeffizienz genutzt werden. Demnach kann sowohl durch die Verwendung einiger WLAN Accesspoints (ähnlich dem unter 2.2.2 Mobilitätsvorhersage/Bandbreitenvorhersage angesprochenen Ansatz), als auch die Verbreitung über die Ad-Hoc Netzwerke der Empfänger zu Energieeinsparungen führen[1]. Da WLAN effizienter ist bei der Datenübertragung als das verwendete Mobilfunknetz (das heißt mit derselben Energiemenge mehr Daten übermitteln kann[2]), ist die Energieeinsparung proportional zur Menge der aus dem Mobilfunknetz ausgelagerten Daten. Daher sind die in der erwähnten Arbeit gezeigten Ansätze ebenfalls auf eine maximale Datenauslagerung aus. Der Unterschied zu dieser Bachelorarbeit besteht darin, dass die von Ristanovic et al. erstellte Arbeit eine Erhöhung der Auslagerung durch eine Verlängerung der Nachrichtenlebenszeit erzielt. Im Vergleich zu dieser Arbeit (in der Nachrichten wenige Minuten gültig sind) werden dort Nachrichtenlebenszeiten bis zu mehreren Stunden verwendet.

2.2.5 Räumliche Verlagerung

Die bisher erwähnten Arbeiten legen den Fokus hauptsächlich auf die Optimierung bzw. Reduzierung des Downloadtraffics, also Nachrichten von einem Server zu einem oder mehreren Empfängern. Die Veröffentlichung „Spatial Uplink Mobile Data Offloading Leveraging Store-carry-forward Paradigm“ von H. Izumikawa, S. Awiphan und J. Katto [IAK12] rückt dagegen den Uploadtraffic in den Mittelpunkt. Es geht also darum Nutzer-generierte Daten zu versenden. Die Idee der Arbeit ist dabei, ausgelastete Basisstationen zu meiden und stattdessen das Hochladen so lange zu verschieben, bis eine weniger ausgelastete Funkzelle erreichbar ist. Daher ist nicht die Vermeidung von Traffic das Ziel, sondern die gleichmäßigere Auslastung von Funkzellen. Um das zu erreichen werden die von einem Nutzer generierten Daten nicht sofort per Mobilfunk versendet, sondern einem anderen Nutzer, der sich zu einer weniger ausgelasteten Basis Station bewegt, per Ad-Hoc Übertragung (WLAN oder Bluetooth) weitergegeben. Erreicht dieser „Überträger“ (bzw. „Messenger“) dann eine besser geeignete Basisstation, so überträgt er die Daten per Mobilfunk. Falls sich ein Nutzer in einer stark ausgelasteten Funkzelle befindet und kein Überträgerclient erreichbar ist, werden die Daten wie gewohnt per UMTS oder 3G übertragen. Der größte Unterschied zur vorliegenden Arbeit ist, dass im Folgenden nur die Reduzierung von Traffic im Mobilfunknetz, nicht die Verlagerung betrachtet wird.

[1] „We find that both solutions [Anm.: WLAN APs und Ad-Hoc Netze] succeed in offloading a significant amount of traffic, with a positive effect on user battery lifetime.“ siehe [RLBCE11] S. 2

[2] Siehe [RLBCE11] Tabelle 1 S. 3

KAPITEL 3

SYSTEMMODELL

Dieses Kapitel beschreibt das Systemmodell dieser Arbeit. Darauf basieren die entwickelten Algorithmen sowie die Evaluierung.

3.1 Definitionen

Client Mobiles Gerät das Nachrichten per Mobilfunk empfangen und verschicken kann. Außerdem ist es in der Lage per WLAN oder Bluetooth Nachrichten an umliegende Geräte zu senden. Im Folgenden ist mit Client immer ein Smartphone gemeint, ebenso denkbar wären beispielsweise auch Tabletcomputer oder Empfangseinheiten in Fahrzeugen.

Server Stellt die Infrastruktur, also den Mobilfunkanbieter sowie dessen Sendeeinrichtungen dar. Muss in der Lage sein, an Clients Nachrichten per Mobilfunk (3G oder UMTS Standard) zu verschicken.

Request Eine Datennachricht (beispielsweise der Inhalt einer Nachrichtenwebseite oder ein Video) die sowohl von der Infrastruktur zu einem Client, als auch zwischen Clients versendet werden kann.

3.2 Kommunikation

3.2.1 Interfaces

Im Simulator werden zwei Kommunikationsarten vorausgesetzt. Für die Datenübertragung zwischen dem Server (bzw. der Infrastruktur) und den Clients wird 3G genutzt, da dieses zumindest in vielen Städten inzwischen verfügbar ist. Außerdem bietet es ausreichend hohe

Bandbreiten für die hier getesteten Dateigrößen (siehe 5.2.1 Aufbau). Die angenommene Bandbreite beträgt 2,5 Mbit/s also 0,3125 MB/s [1] Für die Kommunikation zwischen den Clients wird WiFi-Direct verwendet[wif] (ebenso wäre „Bluetooth“ möglich, dies wurde allerdings wegen des geringeren Übertragungsradius hier nicht gewählt). WiFi-Direct setzt sich bei immer mehr der aktuellen Smartphones durch, daher ist in naher Zukunft damit zu rechnen, dass die meisten neuen Smartphones mit dieser Technik ausgerüstet sind. Die Geschwindigkeiten von WiFi-Direct hängen von dem im Gerät implementierten WLAN Standard ab. Viele neue Geräte (beispielsweise das Model „Galaxy S3“ von Samsung) unterstützen den IEEE Standard „802.11n“ [802]. Dieser bietet verschiedene Übertragungsgeschwindigkeiten zwischen wenigen Mbit/s bishin zu 600Mbit/s. Für die hohen Übertragungsgeschwindigkeiten werden meist mehrere parallele Übertragungen zusammengerechnet. Davon kann man bei der Ad-Hoc Übertragung, die in dieser Arbeit angestrebt wird nicht ausgegangen werden. Hier stehen sequentielle Übertragungen (meist) im Vordergrund. Daher wird im Folgenden mit einer niedrigeren Übertragungsgeschwindigkeit von 16Mbit/s bzw. 2Mbyte/s gerechnet [2]. Als Übertragungsradius wird bei WLAN bis zu 200m gerechnet [3]. Auch dieser Wert ist theoretischer Natur und daher meist in der Realität wesentlich kleiner (z.B. durch schlechtere Übertragungen zwischen Häuserschluchten oder in Gebäuden). Daher wird im Folgenden mit einem Übertragungsradius von 25m ausgegangen.

Neben den verschiedenen Kommunikationsinterfaces werden zwischen den Teilnehmern auch unterschiedliche Arten von Nachrichten versandt. Besonders wichtig sind dabei die „Acknowledge“ Nachrichten (kurz „ACK“) welche von einem Client an den Server geschickt werden um diesen über den erfolgreichen Empfang eines Requests zu informieren. Näheres zu den verschiedenen Nachrichtentypen unter: 4.2 Nachrichtentypen.

3.2.2 Kommunikationspartner

Die für die opportunistische Verteilung der Nachrichten wichtigste Kommunikation ist die zwischen den Clients. Denn hierbei werden die Datennachrichten über das zur Verfügung stehende Ad-Hoc Netzwerk ausgetauscht. Unter anderem bietet der ONE Simulator diese Funktion bereits an, sodass diese nicht implementiert werden muss. Weiterhin gibt es die Kommunikation zwischen einem Client und dem Server (bzw. der Infrastruktur). Hierbei werden hauptsächlich Bestätigungsnachrichten, also eine Nachricht die den Erhalt einer Datennachricht quittiert und Positionsnachrichten gesendet. Die Positionsdaten sind nötig, da die auf dem Server ausgeführten Algorithmen anhand der Position den bestmöglichen Zielclient bestimmen.

[1] Siehe [BDR12] S. 6
[2] Siehe [802] Seite 345ff.
[3] Siehe [HSC09] S.1 Tabelle 1.

3.2.3 Delay Tolerance

Wie bereits unter 1.1 Problembeschreibung beschrieben, sind Requests „delay tolerant“. Das heißt, es ist nicht zwingend notwendig, dass ein Request sofort an alle Empfänger übermittelt wird, sondern eine gewisse Verzögerung kein Problem darstellt. Wie hoch diese Verzögerung ist muss je nach Anwendungszweck ermittelt werden. Eine möglichst lange Verzögerung ist jedoch anzustreben, da die Anzahl der Ad-Hoc Übertragungen direkt mit der möglichen Verzögerung der Nachrichten steigt. Für alle folgenden Annahmen beträgt die maximale Verzögerung eines Requests 24 Minuten, nach Ablauf dieser Zeit muss der Request alle Empfänger erreicht haben. Damit entspricht dies auch der Lebensdauer bzw. Gültigkeit eines Requests. Ist ein Request mehr als 24 Minuten alt, wird dieser nicht mehr als gültig, sondern als „veraltet“ angesehen. Die adaptive Verteilung der Requests, also der Datenaustausch zwischen den Clients, geschieht allerdings nur während der ersten zehn Minuten. Die verbleibende Zeit ist ein Puffer, der genutzt wird um Clients den Request zu senden, die diesen noch nicht erhalten haben (siehe auch 4.1.2 Regulärer Ablauf (mit adaptiver Verteilung)).

3.3 Clients

Clients sind internetfähige Smartphones, die sowohl von einem Server als auch von benachbarten Clients Daten empfangen können. Des Weiteren muss jeder Client in der Lage sein, per Ad-Hoc Übertragung (hier mittels WiFi Direct, möglich wäre aber auch Bluetooth) Nachrichten an Empfänger in seiner Umgebung weiterleiten zu können.

3.3.1 Bewegung

Clients bewegen sich immer mit derselben Geschwindigkeit, mit der sich die Person die das Mobiltelefon besitzt, bewegt. Es wird nur der Fall betrachtet, dass eine Person sich zu Fuß bewegt, was etwa einer Geschwindigkeit zwischen 0.5 und $1.5 m/s$ entspricht. Diese Geschwindigkeit wird während der gesamten Simulation beibehalten. Als weitere Einschränkung können Clients sich nur auf Straßen oder Wegen aufhalten. Positionen außerhalb der im Kartenmaterial vorhandenen Wege sind nicht vorgesehen.

3.3.2 Voraussetzung

Jeder Client muss in der Lage sein, seine korrekte GPS-Position zu bestimmen um diese regelmäßig an den Server zu schicken. Des Weiteren wird vorausgesetzt, dass auf jedem beteiligten Client ein bestimmtes Programm („App“) installiert ist, welches sowohl den Datenaustausch zwischen den Telefonen als auch die Kommunikation mit dem Server steuert. Außerdem müssen alle Clients eine gemeinsame Schnittstelle zum Austausch von

Daten mit benachbarten Geräten besitzen. Im Folgenden wird angenommen, dass diese Schnittstelle WiFi Direct ist.

KAPITEL 4

ADAPTIVE VERTEILUNG

Dieses Kapitel beschreibt die verschiedenen Algorithmen die zur opportunistischen Verteilung der Requests implementiert wurden.

4.1 Grundlagen

4.1.1 Regulärer Ablauf (ohne adaptiver Verteilung)

Wie bereits unter 1.1 Problembeschreibung beschrieben, ist das Reduzieren des Datenverkehrs zunächst der wichtigste Punkt. Bisher sieht die Verteilung von Nachrichten ungefähr wie folgt aus: Angenommen es existieren drei Clients *A* - *C*, die alle zu einem ähnlichen Zeitpunkt dieselben Daten anfordern.

1. Server schickt Nachricht an Client *A*
2. Server schickt Nachricht an Client *B*
3. Server schickt Nachricht an Client *C*

Normalerweise würde der Server also an jeden einzelnen Client, die entsprechenden Daten nacheinander versenden. Das heißt, der verursachte Traffic entspricht dreimal der Größe der versendeten Nachricht.

4.1.2 Regulärer Ablauf (mit adaptiver Verteilung)

Anstatt nun jedem Client, der Daten anfordert, diese zuzuschicken und damit dieselbe Nachricht mehrfach zu versenden, soll nun mit Hilfe einer adaptiven Verteilung der erforderliche Datenverkehr reduziert werden. Möglich wird dies, wenn mehrere Clients auf die gleiche Nachricht warten (z.B. weil dieselbe Nachrichtenwebseite oder dasselbe Video betrachtet wird). In diesem Fall soll nur ein einziger Client den Request (bzw. die Nachricht) erhalten und diesen selbstständig an andere Clients in seiner Umgebung weiterleiten. Optimal wäre es also, wenn der Request nur einmal verschickt wird und alle anderen Clients im System

diesen durch benachbarte Clients erhalten. Die Verteilung wird durch die Bewegungen der Clients begünstigt. Bewegen sich die Clients, die einen Request empfangen haben sehr schnell, verteilt sich der Request geografisch gesehen ebenfalls sehr weit. Eine optimale Verteilung (also die Verteilung eines Requests, bei der der Request nur einmal vom Server verschickt wird) ist allerdings ohne eine sehr lange Nachrichtenlebensdauer zuzulassen, nicht möglich. Dafür müsste die Lebensdauer eines Requests mehrere Stunden betragen. Was wiederum bedeutet, dass ein Request über mehrere Stunden aktuell ist, also auch über mehrere Stunden von den Clients verteilt werden kann. Dies ist im Hinblick auf die hier genannten Einsatzmöglichkeiten (siehe 1.3 Einsatzmöglichkeiten) nicht praktikabel. Daher wird ein Request regelmäßig an einen neuen Client verschickt, um so eine möglichst hohe Zahl an Ad-Hoc Übertragungen zu gewährleisten, ohne dass die Nachrichten dabei eine sehr hohe Lebensdauer benötigen. Die Übertragung zwischen den benachbarten Clients kann dabei entweder durch WiFi Direct oder Bluetooth erfolgen. Im Folgenden wird davon ausgegangen, dass WiFi Direct verwendet wird. Der Prozess der angewendet wird um Nachrichten adaptiv zu verteilen gliedert sich in mehrere Phasen:

1. Erstellen eines neuen Request bzw. Nachricht.
2. Suchen eines Zielclients für die entsprechende Nachricht und versenden der Nachricht an diesen.
3. Überprüfen ob die Erstellung oder ein erneutes Senden eines Requests nötig ist.
   a) Es sind mehr als 60 Sekunden seit dem letzten Versand eines Requests vergangen und die Anzahl der Ad-Hoc verteilten Nachrichten ist zu gering ⇒ Schritt 2 ausführen.
   b) Es sind mehr als 200 Sekunden vergangen, seit dem zuletzt ein neuer Request erstellt worden ist ⇒ Phase 1 ausführen.
   c) Ist eine Nachricht nur noch weniger als 14 Minuten gültig, starte die „Panic Zone“ ⇒ Phase 4 ausführen
   d) Trifft keiner der vorherigen Fälle zu, wird eine Sekunde gewartet und erneut überprüft ⇒ Phase 3 ausführen.
4. Versenden (per Mobilfunk) der Nachricht an alle Clients, die diese noch nicht erhalten haben. ⇒ Phase 3 ausführen.

Parallel zu jeder dieser Phasen werden empfangene Requests zwischen den Clients selbstständig ausgetauscht.

Phase 1) Initialisierung: Bevor ein Request in das System gegeben wird, werden zunächst alle Clients bestimmt, die den Request empfangen sollen. Zur Vereinfachung wird in dieser Arbeit angenommen, dass immer alle Clients die sich im System bewegen einen Request empfangen sollen. Dies ist nötig, damit ein Client überprüfen kann, ob eine Nachricht an seine Nachbarn weitergeleitet werden muss.

Phase 2) Ziel bestimmen (Verteilungsstrategie anwenden): Nachdem ein Request mit der Liste seiner Empfänger initialisiert wurde, wird mit dem gewählten Algorithmus der erste Zielclient bestimmt. Diesem wird die Nachricht übertragen, und der Client beginnt daraufhin allen anderen Clients, denen er begegnet den Request weiterzuleiten. Jeder Client der daraufhin den Request erhält, leitet ab diesem Zeitpunkt den Request wiederum an jeden Client innerhalb seines Übertragungsradius weiter. Insbesondere durch die Bewegung der Clients, verteilt sich daher der Request im Laufe der Zeit immer weiter.

Phase 3a) Überprüfung (erneutes Versenden): Um die Anzahl der Ad-Hoc Übertragungen zwischen den Clients so hoch wie möglich zu halten, wird im Laufe der Lebenszeit eines Requests regelmäßig ein neuer Zielclient bestimmt. Insbesondere dann, wenn die Anzahl der eingehenden Bestätigungsnachrichten (ein Client bestätigt dem Server den Erhalt eines Requests) unter ein gewisses Niveau fällt (siehe auch: 4.4.2 Reinjection Threshold). Dadurch wird regelmäßig in Clientansammlungen, die den Request noch nicht erhalten haben die entsprechende Nachricht injiziert. Daraufhin steigt die Anzahl der Ad-Hoc Übertragungen, da sich die Nachricht schnell unter den benachbarten Clients verbreitet. Das periodische Bestimmen eines neuen Zielclients ist notwendig, da keine Garantien für die Bewegungsrichtung oder Geschwindigkeit eines Clients möglich sind. Es kann also nicht der global bestmögliche Client bestimmt werden, der bis zum Ende der Simulation am meisten Ad-Hoc Übertragungen auslöst, da über die zukünftigen Bewegungen der Clients keine Aussage gemacht werden kann.
Nach dem initialen Auswählen eines Zielclients wird also in regelmäßigen Abständen (ein Mal pro Sekunde) überprüft, wie viele ACKs (Acknowledge-Nachrichten) in den letzten 60 Sekunden beim Server für einen Request eintrafen. Der Request wird genau dann erneut versendet, wenn zwei Bedingungen erfüllt sind:

1. Der Request wurde vor mehr als 60 Sekunden zuletzt versendet.
2. Ein gewisser Prozentsatz (siehe 4.4.2 Reinjection Threshold) an eingehenden ACKs wurde nicht erreicht.

Das heißt, es wird einmal pro Sekunde überprüft ob es nötig ist einen Request erneut zu versenden, aber es liegen mindestens 60 Sekunden zwischen dem Versenden desselben Requests. Die sekündliche Überprüfung ist insbesondere auch deshalb nötig, weil nicht alle aktiven Requests zum selben Zeitpunkt erneut verschickt werden müssen. Ist es nötig die Nachricht erneut zu versenden (beide Bedingungen sind erfüllt), wird mittels des aktuellen Algorithmus wieder ein neues Ziel bestimmt und an den entsprechenden Client der Request verschickt. Konnte der Algorithmus kein Ziel finden, wird das nächste Ziel per Zufall bestimmt. Dies passiert beispielsweise, wenn bereits sehr viele Clients die Nachricht erhalten haben und die wenigen verbleibenden Clients sehr weit auseinander sind. Dieser Vorgang wiederholt sich während der gesamten Gültigkeitsdauer des Requests.

Nach dem Bestimmen eines Zielclients wird außerdem gespeichert, welche Clients durch den Zielclient „überdeckt“ werden (also den Request durch den Zielclient im Laufe der Zeit erhalten). Welche Clients als überdeckt gewertet werden, hängt von der verwendeten Verteilungsstrategie ab. Bei Verwendung der DISTANCE Strategie gelten beispielsweise alle

Clients die sich im Übertragungsradius des Zielclients befinden als überdeckt. Außerdem wird noch festgehalten, zu welchem Zeitpunkt diese Clients den Request spätestens erhalten sollten. Dies ist insbesondere für den Parameter 4.4.1 Adaptive message inquiry wichtig. Ausgenommen ist dabei der RANDOM Algorithmus, der per Zufall Ziele bestimmt und daher keine Aussagen über Überdeckungen oder Zeitpunkte von Datenübertragungen getroffen werden können.

Phase 3b) Überprüfung (neuer Request): Des Weiteren wird regelmäßig ein neuer Request in das System eingefügt. Dies geschieht alle 200 Sekunden, womit also außer zu Beginn der Simulation immer vier bis fünf aktuelle Requests verteilt werden müssen. Damit wird die Simulation etwas realitätsnaher, da in einem echten System ebenfalls mehrere auszuliefernde Requests unterwegs sein werden.

Phase 3c) & 4) Überprüfung (Panic Zone): Haben nicht alle Ziele eines Requests diesen kurz vor Ende seiner Gültigkeitsdauer erhalten, so startet der Server in die sogenannte „Panic Zone“ (nach [WAL+11] S. 3). Das bedeutet, dass 14 Minuten vor Ablauf des Requests die Nachricht über das UMTS-Netz an alle fehlenden Ziele versendet wird. Hierbei wurden 14 Minuten als Mittelwert verwendet, da dies sich nach einigen Tests als ausreichendes Zeitfenster herausgestellt hat, um allen Empfängern die den Request noch nicht erhalten haben die Nachricht zu senden. Dieser Wert muss bei anderen Bedingungen bzw. dem realen Einsatz unter Umständen (größere Nachricht, langsamere Übertragung der Infrastruktur) geändert werden. Tatsächlich ist dann meist deutlich weniger als die angegebene Zeit erforderlich.

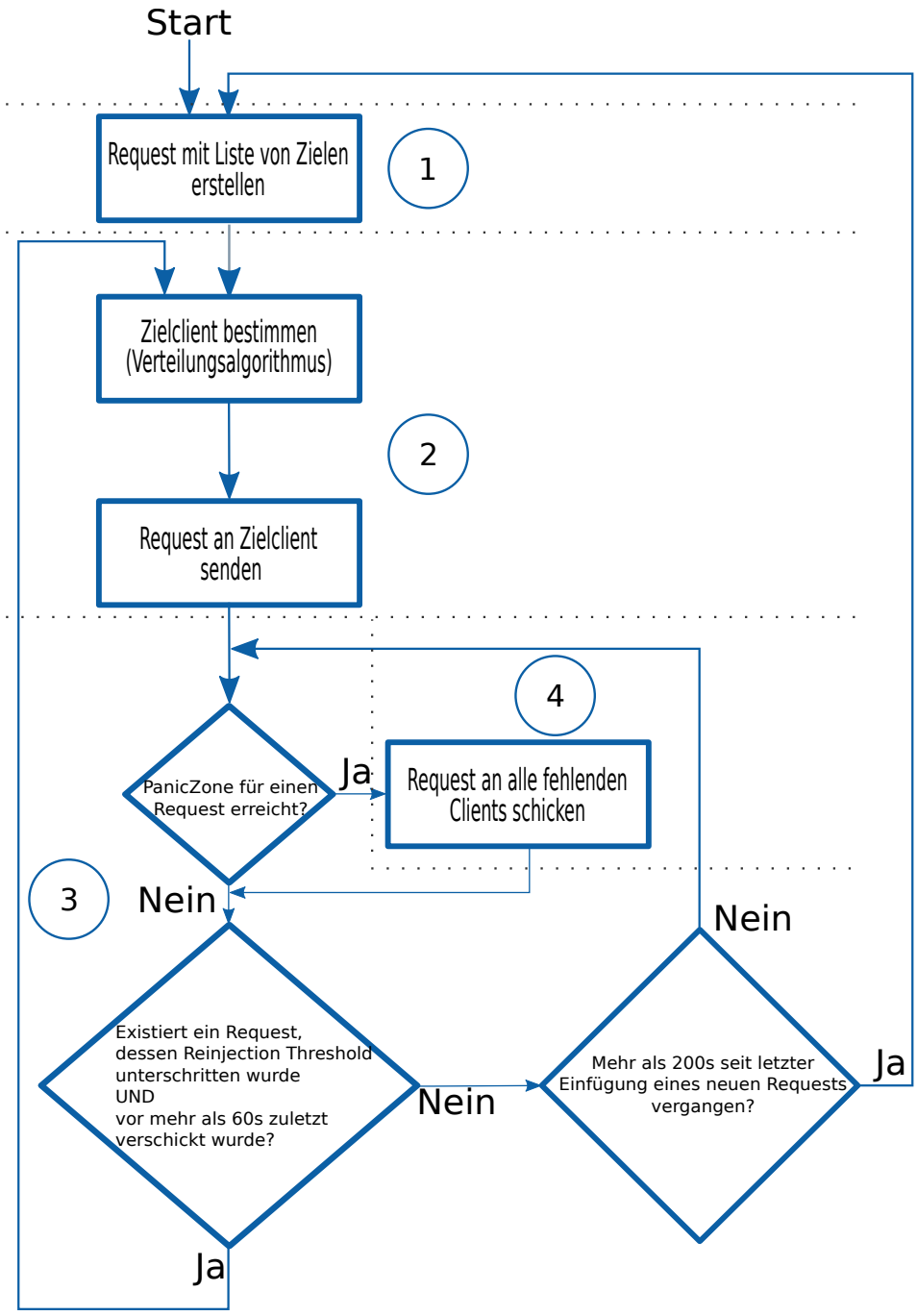

Abbildung 4.1: Darstellung des regulären Ablaufs. Die Zahlen in den Kreisen entsprechen den vorher genannten Phasen.

4.1.3 Beispiel

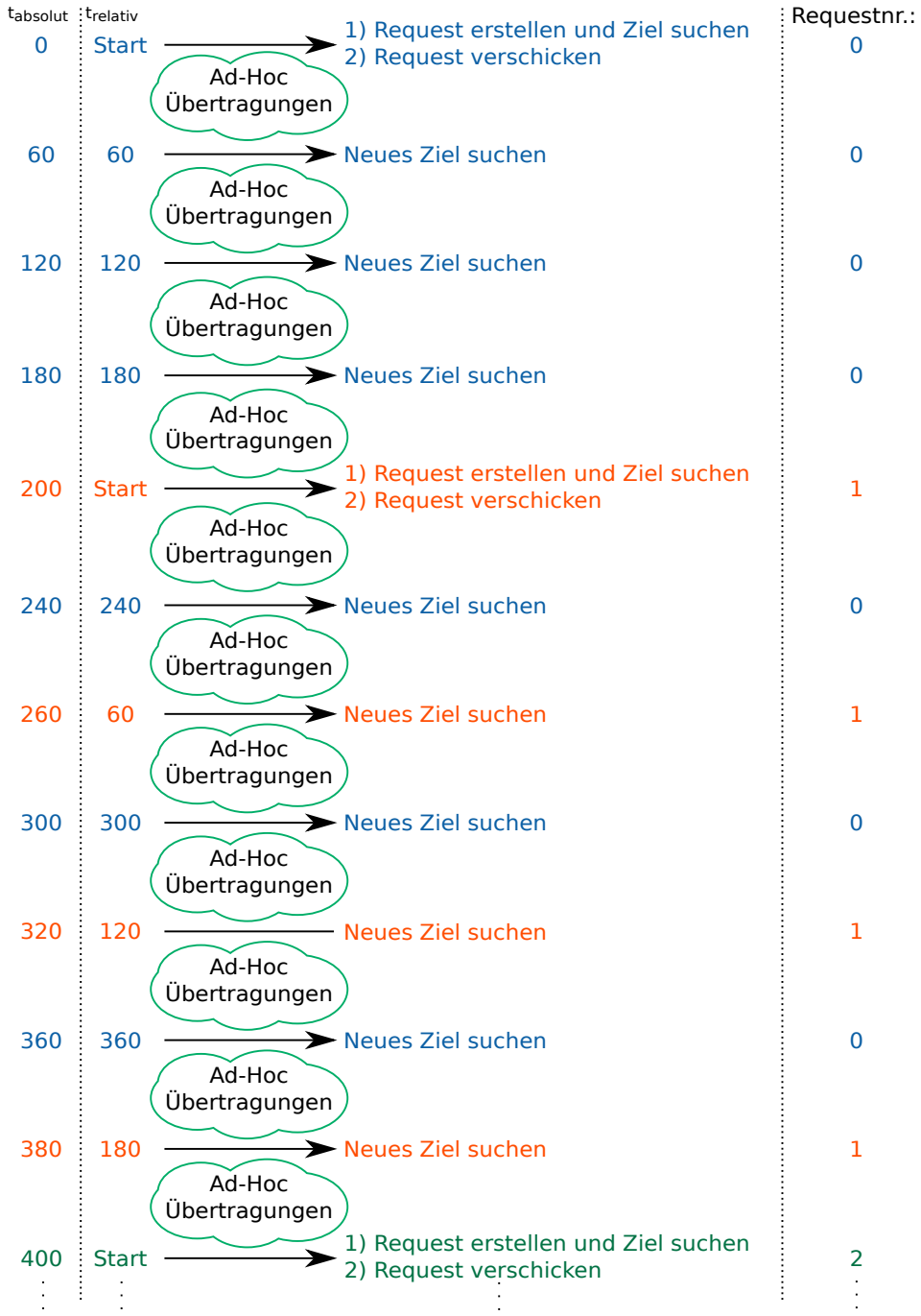

Abbildung 4.2: Schaubild entspricht dem zeitlichen Ablauf, wenn bei keinem Request der Reinjection Threshold überschritten wurde. Das heißt, pro Request wird alle 60s eine neue Nachricht versendet.

4.1.4 Zielclients und fehlende Clients

Für jeden Request existiert eine Liste der „fehlenden Clients“. Diese Liste repräsentiert die Clients, die den Request noch empfangen müssen. Dabei müssen genau zwei Eigenschaften zutreffen, damit ein Client zur der Liste gezählt wird:

1. Der Client darf den entsprechenden Request noch nicht erhalten haben.
2. Für den Client darf kein zukünftiger Zeitpunkt gespeichert sein, zu dem er den Request wahrscheinlich erhält.

Zu Punkt 2: Bei jedem der folgenden Algorithmen (außer RANDOM) wird eine Liste mit den Clients erstellt, denen ein Zielclient mit relativ hoher Wahrscheinlichkeit den Request weiterleiten kann. Je nach Algorithmus basiert diese Wahrscheinlichkeit auf der Tatsache, dass die Clients sehr nah am Zielclient sind, oder die Clients sich zukünftig treffen. Dabei wird für jeden Client der Liste ein fester Zeitpunkt gespeichert, zudem er spätestens den Request empfangen haben sollte. Der berechnete Zeitpunkt hängt wiederum von dem verwendeten Algorithmus ab, d.h. Zeitpunkte die von DISTANCE (nutzt nur Entfernungen zwischen Clients) berechnet wurden liegen i.d.R. weniger weit in der Zukunft als Zeitpunkte die von DISTANCE_MOTION (nutzt Bewegungsrichtung der Clients) berechnet wurden.

Ein Client kann genau dann einen Request vom Server erhalten (also Zielclient werden), wenn er in der Liste der „fehlenden Clients“ ist. Dies ist eine notwendige aber nicht hinreichende Bedingung. Um tatsächlich Ziel zu werden, muss der Client für den jeweilig angewendeten Algorithmus auch am erfolgversprechendsten sein.

4.1.5 Nächste Clientposition

Vereinfachend wird im Folgenden angenommen, dass ein Client neben seiner Geschwindigkeit und aktuellen Position auch die voraussichtlich nächste Position berechnen und an den Server schicken kann. Als „nächste Position“ wird im ebenfalls erstellten Programm der nächste Wegpunkt auf dem in 5.1 ONE Simulator beschriebenen Pfad eines Clients genutzt. Nachfolgend soll nun begründet werden, warum dieses Vorgehen möglich ist und wie die nächste Position eines Clients auch berechnet werden könnte:

Grundsätzlich ist es natürlich nicht möglich, die Bewegung der Clients (und damit das Verhalten z.B. an Kreuzungen) vorherzusagen. Allerdings kann trotzdem angenommen werden, dass die nächste Position in der Regel bekannt ist, da es prinzipiell bereits genügt, wenn ein Client die zwei letzten Positionsdaten kennt, da somit zunächst die Straße auf der er sich befindet und mit dem Zeitpunkt zu dem die Positionsdaten festgehalten wurden die Richtung bestimmt werden kann. Damit kann dann im Kartenmaterial ausgehend von der aktuellen Position bestimmt werden, in welcher Richtung der Client sich auf der Straße bewegt. Dieses Vorgehen funktioniert offensichtlich nur dann nicht, wenn ein Client an einer Kreuzung abbiegt oder seine Richtung plötzlich ändert (auf derselben Straße zurück geht). Aber bereits nach dem ersten Positionsupdate nach einer Richtungsänderung kann wieder die nächste Position nach vorherigem Schema bestimmt werden.

Daher kann als nächste Position immer der nächste Datenpunkt des Pfades auf dem sich ein Client bewegt genutzt werden.

Hier wurde nur der Einfachheit halber die Berechnung der nächsten Postion auf dem Client ausgeführt. In einer realen Anwendung wäre die Berechnung auf dem Server sinnvoller, da die Ressourcen der Clients (Speicherplatz, Rechenzeit, Akkulaufzeit) sehr begrenzt sind.

4.2 Nachrichtentypen

Dieser Abschnitt beschreibt alle Nachrichtentypen die zur Ausführung der beschriebenen Algorithmen nötig sind. Dabei wird zunächst unterteilt in 4.2.2 Serverseitig, dieser Abschnitt enthält die Nachrichtentypen die vom Server ausgehend zu den Clients gesendet werden, und 4.2.3 Clientseitig welcher wiederum die Nachrichtentypen beschreibt, die von den Clients zum Server gesendet werden.

4.2.1 Allgemeines

Für alle hier beschriebenen Nachrichten (mit Ausnahme von „Request“) wird eine Größe von 250 Byte angenommen. Davon sind bis zu 233 Byte für Nutzdaten verwendbar [1]. Diese Werte gelten sowohl für Nachrichten von den Mobilgeräten zur Infrastruktur, als auch von einem Server (bzw. der Infrastruktur) zu Mobilgeräten.

Alle versendeten Nachrichten haben folgenden Inhalt gemeinsam: Senderadresse, Empfängeradresse und Typ der Nachricht. Der Typ entspricht den folgenden Nachrichtenarten.

Bei allen Nachrichten außer dem Request, wird davon ausgegangen, dass die Nachricht per Mobilfunk sofort übertragen wird und daher keine Lebensdauer angegeben werden muss. Lediglich bei der Request Nachricht muss eine Lebensdauer gesetzt werden, da ansonsten der Ansatz des Traffic Offloading hier nicht funktionieren kann.

4.2.2 Serverseitig

Nachrichten die vom Server ausgehend zu den Clients geschickt werden.

[1] Siehe [3gp] S.34 7.3.1.1 und 7.3.1.2

4.2.2.1 LocationUpdateInquiry

Diese Nachricht wird vom Server versandt, um ein Positionsupdate der Clients anzufordern. In der Regel wird diese Nachricht als Broadcast versendet, da meist eine Aktualisierung der Position aller Clients erforderlich ist. Neben dem Standardnachrichteninhalt (Sender- und Empfängeradresse sowie Typ) hat diese Nachrichten keine weiteren Nutzdaten.

4.2.2.2 Request

Die Requestnachricht entspricht der Datennachricht, die vom Server an die Clients verschickt werden soll. Die Größe kann theoretisch variieren, je nachdem welche Daten verschickt werden sollen. Neben den eigentlich Daten enthält diese Nachricht außerdem eine Liste mit den Adressen der Empfänger. Dies ist erforderlich, damit bei der Ad-Hoc Übertragung eines Requests die empfangenden Clients sicherstellen können, dass sie Ziel der Nachricht sind. Im Gegensatz zu den anderen Nachrichten, besitzt jeder Request eine Lebensdauer von 24 Minuten. Innerhalb dieser Zeit muss der Request an alle Empfänger verschickt worden sein. Dies soll das System etwas realistischer machen, denn auch in einem realen System gibt es obere Grenzen, bei denen eine Nachricht ausgeliefert sein muss.

4.2.3 Clientseitig

Nachrichten, die von den Clients ausgehend zum Server geschickt werden. Clientseitige Nachrichten sind entweder Antworten auf vorherige Nachrichten des Servers, z.B. eine Positionsmeldung oder ein Acknowledgement nach erfolgreichem Transfer des Requests per Mobilfunk, bzw. nach dem Erhalt durch Ad-Hoc Übertragungen.

4.2.3.1 Acknowledge (ACK)

Acknowledge Nachrichten werden von Clients beim Empfang eines Requests an den Server gesendet. Damit wird signalisiert, dass die Nachricht erfolgreich empfangen wurde. ACKs werden sowohl dann verschickt, wenn der Request mittels dem Mobilfunknetz empfangen wurde, als auch bei einer Ad-Hoc Übetragung durch einen benachbarten Client. Neben dem Standardnachrichteninhalt (Sender- und Empfängeradresse sowie Typ) hat diese Nachrichten keine weiteren Nutzdaten.

4.2.3.2 LocationUpdate

LocationUpdate Nachrichten sind die Antworten von Clients auf die 4.2.2.1 LocationUpdateInquiry Nachricht. Diese Nachricht hat neben dem Request als einzige Nachricht Nutzdaten. Sie enthalten: Aktuelle Position, zuletzt besuchte Position, nächste Position (Schätzung) und die Geschwindigkeit des Clients. Für die Positionsinhalte müssen je zwei Doubles (also

$3 * 2 * 8byte = 48byte$) und für die Geschwindigkeit ein weiterer Double Wert gespeichert werden. Insgesamt müssen also pro LocationUpdate Nachricht $56byte$ Nutzdaten verschickt werden. Für einen realen Einsatz genügen für eine Position zwei Double Werte nicht, es muss daher unter Umständen eine größere Nachricht gewählt werden um die GPS-Daten zu speichern.

4.3 Verteilungsstrategien

Eine Verteilungsstrategie ist ein Algorithmus, der aus der Menge der vorhandenen Clients den Client heraussucht, der am meisten Ad-Hoc Übertragungen erzielen kann. Jede der folgenden Strategien unterscheidet sich dabei in der Art und Weise, wie mögliche Ad-Hoc Übertragungen bestimmt werden.

Einige der Strategien verwenden die aktuellen, sowie die wahrscheinlich nächsten Positionen der Clients. Für den Server ist es allerdings unmöglich vorherzusagen, wie sich Clients (bzw. die Person die das Mobiltelefon besitzt) in der Zukunft bewegen werden. Daher sind alle hier vorgestellten Maßnahmen zur Bestimmung des optimalen Zielclients lediglich Schätzungen, basierend auf den dem Server vorliegenden Positionsdaten.

4.3.1 RANDOM

RANDOM ist der einfachste der vorhandenen Algorithmen und wurde hauptsächlich zum Vergleich für andere Algorithmen entwickelt. Die Funktion ist sehr simpel: Aus der Menge der fehlenden Clients wird per Zufall ein neuer Client bestimmt, der den Request vom Server erhalten soll. Daraufhin versendet der Server den Request und der Client wird (sobald er in entsprechender Entfernung ist) anderen Clients diese Nachricht übermitteln. Über die Effizienz des Algorithmus kann naturgemäß keine genaue Angabe gemacht werden. Die Simulationen zeigen jedoch, dass dieser Ansatz überraschenderweise relativ hohe Ad-Hoc Transferraten erzielen kann.

4.3.2 DISTANCE

DISTANCE entspricht prinzipiell dem „Static Coverage“ Algorithmus von P. Baier, F. Dürr und K. Rothermel [2] aus der diesem Bericht zugrunde liegenden Arbeit.

[2]Siehe „TOMP: Opportunistic Traffic Offloading Using Movement Predictions“ [BDR12] S. 4

4.3.2.1 Funktion

Der Algorithmus nutzt die Positionsdaten der Clients um mögliche Ad-Hoc Übertragungen zu bestimmen. Dazu wird zunächst eine Distanzmatrix (D) erstellt, in der die Abstände zwischen den Clients eingetragen werden. Genauer: Haben zwei Clients (c_i, c_j) einen Abstand der kleiner als der Ad-Hoc Radius ist, so wird die Distanz, andernfalls 0.0 eingetragen.

$$D[i][j] = \begin{cases} 0.0 & \text{if } dist(c_i, c_j) > RANGE_{Ad-Hoc} \vee i = j \\ dist(c_i, c_j) & \text{if } dist(c_i, c_j) \leq RANGE_{Ad-Hoc} \end{cases} \tag{4.1}$$

Abbildung 4.3: Gleichung zur Berechnung der Distanzmatrix.

Ein guter bzw. viel Erfolg versprechender Client ist nun dadurch erkenntlich, dass sich in einer Zeile der Matrix besonders viele Werte ungleich 0.0 befinden. Die Zeile mit den meisten Einträgen (ungleich 0.0) wird als bester Client betrachtet und damit als nächstes Ziel ausgewählt.

4.3.2.2 Problem

Ein Problem dieses Algorithmus ist, dass in dem Fall, in dem ein Client viele andere überdeckt, nicht alle überdeckten Clients die Nachricht erhalten. Dies liegt in erster Linie daran, dass Clients (bzw. die Personen die die entsprechenden Geräte tragen) in der Simulation nicht stehen bleiben wenn sie anderen begegnen. Durch die (möglicherweise entgegengesetzte) Bewegung verlassen sie den jeweils anderen Übertragungsbereich. Das heißt, dass die Zeit in der zwei Geräte theoretisch Nachrichten austauschen können, relativ kurz ist. Da jede Übertragung zwischen zwei Clients aber eine gewisse Zeit braucht (näheres zu Übertragungsraten unter 3.2.1 Interfaces) passiert es, dass die Übertragung vom Senderclient zum Empfängerclient abbricht, weil die Entfernung größer wird als $RANGE_{Ad-Hoc}$, oder gar nicht erst stattfindet.

Zeit bis zur Auslieferung des Requests: Für die Übertragung an alle benachbarten Empfänger werden 20 Sekunden eingeplant. Es wird dabei für jeden Empfänger (Einträge die in der Distanzmatrix ungleich null waren) eingetragen, bis zu welchem Zeitpunkt dieser den Request erhalten sollte. Dabei ist es unerheblich, ob ein Empfänger diesen von dem Zielclient direkt, oder von einem anderen Client erhält. Geht man, wie im Systemmodell beschrieben, von WiFi Direct als Kommunikationsinterface aus, reicht diese Zeit auch für sehr große Dateien. Nutzt man allerdings Bluetooth als Übertragungsinterface können nur noch vergleichsweise kleine Dateien übertragen werden. Die Zeit, die für die Übertragung an benachbarte Empfänger nötig ist, muss in einer realen Umgebung an die vorhandenen Gegebenheiten wie Anzahl der Clients, Übertragungsgeschwindigkeit der Mobilgeräte oder

Dateigröße angepasst werden. Während der Implementierung stellte sich jedoch heraus, dass ein Zielclient sehr selten mehr wie fünf andere Clients überdeckt und die angegebene Zeit wurde daher auch für Bluetooth Kommunikation als ausreichend betrachtet.

4.3.3 DISTANCE_ENHANCED

DISTANCE_ENHANCED ist eine Weiterentwicklung des DISTANCE Algorithmus wobei nicht nur direkte Nachbarn als überdeckte Clients gewertet werden, sondern ebenfalls die daraus entstehenden indirekten Überdeckungen. Siehe Abbildung: 4.4 DISTANCE_ENHANCED Beispiel

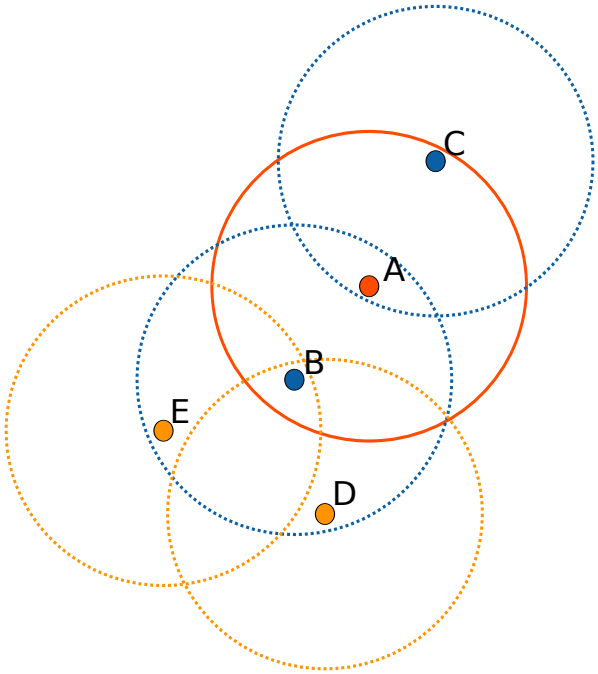

Abbildung 4.4: Beispiel Überdeckung bei DISTANCE_ENHANCED. Client A überdeckt B und C direkt, E und D indirekt.

Bildbeschreibung Die Buchstaben A-E kennzeichnen die durch Punkte repräsentierten Clients. Der Übertragungsradius eines Clients wird durch einen Kreis dargestellt, wobei die durchgezogene Linie dem Übertragungsradius des Zielclients entspricht. Um verschiedene Clients besser unterscheiden zu können, sind die Übertragungsradien farblich entsprechend den Clients markiert. Clients die vom selben Sender überdeckt werden, haben die gleiche Farbe. Der Zielclient A überdeckt B und C direkt, das heißt B und C befinden sich innerhalb seines Übertragungsradius. B überdeckt außerdem die Clients D und E. Der Unterschied zu 4.3.2 DISTANCE besteht nun darin, dass zu der Menge der Überdeckungen des Zielclients auch alle Clients hinzugefügt werden, die durch bereits überdeckte Clients

benachrichtigt werden können. Das heißt, die Menge der von A überdeckten Clients ist: $Cov_A = \{B, C, D, E\}$

4.3.3.1 Funktion

Als Ausgangspunkt wird dieselbe Distanzmatrix wie bei DISTANCE verwendet (siehe 4.3 Gleichung zur Berechnung der Distanzmatrix.). Zunächst wird wie bereits beschrieben nach dem Client (c_{best}) mit den meisten Überdeckungen gesucht. Dabei entsteht eine Menge $M_{c_{best}}$, die alle Clients enthält, die von c_{best} direkt überdeckt werden. Für jeden Client $c_{covered}$ aus dieser Menge wird nun in der Distanzmatrix gesucht, welche weiteren Clients davon überdeckt werden und diese zur ursprünglichen Menge $M_{c_{best}}$ hinzugefügt. Dieser Schritt muss für alle dadurch hinzugefügten Clients ebenfalls durchgeführt werden.

Der Algorithmus in java-ähnlichem Code siehe 4.1 Algorithmus zur Berechnung der überdeckten Clients für DISTANCE_ENHANCED.

```
1  getCoveredClients(double[][] DM, Integer mainClientAdr)
   {
3      Set<Integer> coveredClients = new Set<Integer>(); // set with client IDs
       for (int column = 0; column < DM.length; column++) { // go through
           distance matrix
5          if (DM[mainClientAdr][column] != 0.0) { // other client is in range
              DM[column][mainClientAdr] = 0.0; // set new clients value to 0.0
7             coveredClients.add(c);
           }
9      }
       Iterator i = coveredClients.iterator(); // empty if no new client is
           covered
11     while (i.hasNext()) {
          // recursiv call to get further coverage of newly added clients
13        coveredClients.addAll(getCoveredClients(DM, (Integer)i.next());
       }
15     return coveredClients;
   }
```

Listing 4.1: Algorithmus zur Berechnung der überdeckten Clients für DISTANCE_ENHANCED

Wichtig bei diesem Algorithmus ist, in der Distanzmatrix den Eintrag der die Überdeckung zwischen c_{best} und einem neu gefundenen Client repräsentiert, zu löschen (siehe Code Zeile 6). Das ist notwendig, damit der Algorithmus terminiert, da andernfalls die Menge der gefundenen Clients immer mindestens Größe 1 hat. Offensichtlich wird dies mit einem Minimalbeispiel.
Annahme: Es existieren 2 Clients (A und B) die sich in Übertragungsreichweite zueinander befinden und Zeile 6 des obigen Algorithmus wird nicht ausgeführt.
Ablauf:

1. Zunächst wird für A die Menge $Cov_A = B$ bestimmt (d.h. A überdeckt B). Entspricht Zeilen 3-9 des Algorithmus.

2. Das Unterprogramm wird in Zeile 13 mit B und der unveränderten Distanzmatrix aufgerufen.
3. In Zeilen 3-9 wird A als überdeckter Client gefunden.
4. In Zeile 13 wird das Unterprogramm mit A und der weiterhin unveränderten Distanzmatrix aufgerufen. $\Rightarrow$ Programm terminiert nicht!

Wird nun zusätzlich noch Zeile 6 des obigen Algorithmus ausgeführt, so wird bei Schritt 3 keine weitere Überdeckung gefunden, die Schleife also nicht ausgeführt und die Abbruchbedingung der Rekursion damit erfüllt.

Die beschriebene Art der Überdeckungsbestimmung lässt sich gut als Baum interpretieren (siehe 4.5 DISTANCE_ENHANCED Baumdarstellung). An der Wurzel des Baumes ist der Zielclient. Die direkten Kinder entsprechen den Clients, die sich im Übertragungsradius des Zielclients befinden. Jedes dieser Kinder kann wiederum eigene Kinder haben. Diese befinden sich dann außerhalb des Übertragungsradius des Elternknotens, da es anderenfalls Kinder des Elternknotens wären. Damit lassen sich die überdeckten Clients in verschiedene Level einteilen.

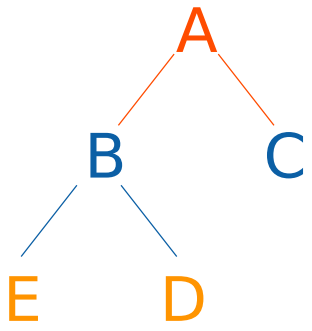

Abbildung 4.5: Beispiel eines einfachen Überdeckungsbaumes bei DISTANCE_ENHANCED anhand des vorherigen Beispiels.

4.3.3.2 Problem

Prinzipbedingt gibt es hier das gleiche Problem wie bei DISTANCE. Überdeckt der Client eines Levels viele andere Clients, reicht möglicherweise die Zeit, die beide in Übertragungsreichweite verbringen, nicht aus um die Nachricht zu übermitteln, da sich beide Clients voneinander weg bewegen. Dieser Effekt verstärkt sich dadurch, dass Clients in äußeren Leveln auf die Übertragung der Nachricht durch die Elternknoten warten müssen und daher weniger Zeit haben die Nachricht an die eigenen Kindknoten zu verteilen. Bei diesem Algorithmus ist das Problem besonders stark ausgeprägt, da unter Umständen bereits eine einzelne nicht ausgelieferte Nachricht ausreicht, dass ein ganzer Ast des Überdeckungsbaumes keine Nachricht erhält. Dieser Effekt lässt sich einschränken, indem die Übertragungsradien äußerer Level kleiner gewählt werden als die der inneren. Das

führt dazu, dass vom Wurzelknoten weiterhin viele Clients die Nachricht erhalten können (voller Übertragungsradius). Je weiter die Nachricht jedoch nach außen wandert, desto näher müssen die Clients zusammen sein, um in die Überdeckungsbestimmung aufgenommen zu werden. Durch die geringere Distanz zwischen den äußeren Clients wird sichergestellt, dass sie sich länger in ihren jeweiligen Übertragungsradien aufhalten (die tatsächlichen Übertragungsreichweiten werden natürlich nicht verringert). Der entsprechende Parameter findet sich unter: 4.4.3 Radius reduction

Zeit bis zur Auslieferung des Requests Es gilt dieselbe Zeit wie bei 4.3.2 DISTANCE, da auch bei DISTANCE_ENHANCED ein Client nur sehr selten mehr als fünf andere Clients überdeckt. Weiterhin kann ein Teil der Übertragung parallel ausgeführt werden, da ein Client der den Request erhalten hat an den nächsten Level bereits senden kann, während der Zielclient noch an die Clients die ihn direkt umgeben sendet.

4.3.4 DISTANCE_MOTION

Im Vergleich zu den vorherigen DISTANCE Algorithmen basiert DISTANCE_MOTION nicht einzig auf der Entfernung zwischen den Clients, sondern hauptsächlich auf deren Bewegungsrichtung und Geschwindigkeit. Daher ist es für diesen Algorithmus nötig, dass Clients regelmäßig ihre aktuelle, die voraussichtlich nächste Position und ihre Geschwindigkeit an den Server senden.

4.3.4.1 Funktion

Wichtigster Schritt ist zunächst die Bewegungsrichtung der Clients festzustellen. Dazu wird für jeden Client ein Richtungsvektor berechnet, der von der aktuellen Position auf die nächste (wahrscheinliche) Position zeigt. Dieser Vektor wird normalisiert ($|\vec{v}| = 1$) und um einen bestimmten Faktor verlängert (siehe 4.4.4 Vector Extension). Die Verlängerung repräsentiert dabei die Zeit, von der angenommen wird, dass sich der Client noch in der berechneten Richtung bewegt. Das heißt, der Bewegungsvektor kann beispielsweise mit dem Faktor 60 verlängert werden, wenn man davon ausgehen kann, dass sich der Client weitere 60 Sekunden in der entsprechenden Richtung bewegt. Durch die Streckung um einen fixen Faktor kann der Endpunkt des Richtungsvektors berechnet werden. Offensichtlich wird der berechnete Endpunkt unwahrscheinlicher, je höher der Verlängerungsfaktor gewählt wird. Damit existiert dann für jeden Client ein Liniensegment, auf dem sich der Client voraussichtlich bewegt.
Beispiele siehe Abbildung 4.6. Die Länge der Vektoren (blau) repräsentiert deren Geschwindigkeit und die Streckung der Richtungsvektoren (grün) um einen bestimmten Faktor. Alle eingezeichneten Schnittpunkte (blaue Kreise) entsprechen möglichen Treffpunkten, d.h. die Clients erreichen diese Punkte wahrscheinlich zu einem ähnlichen Zeitpunkt.

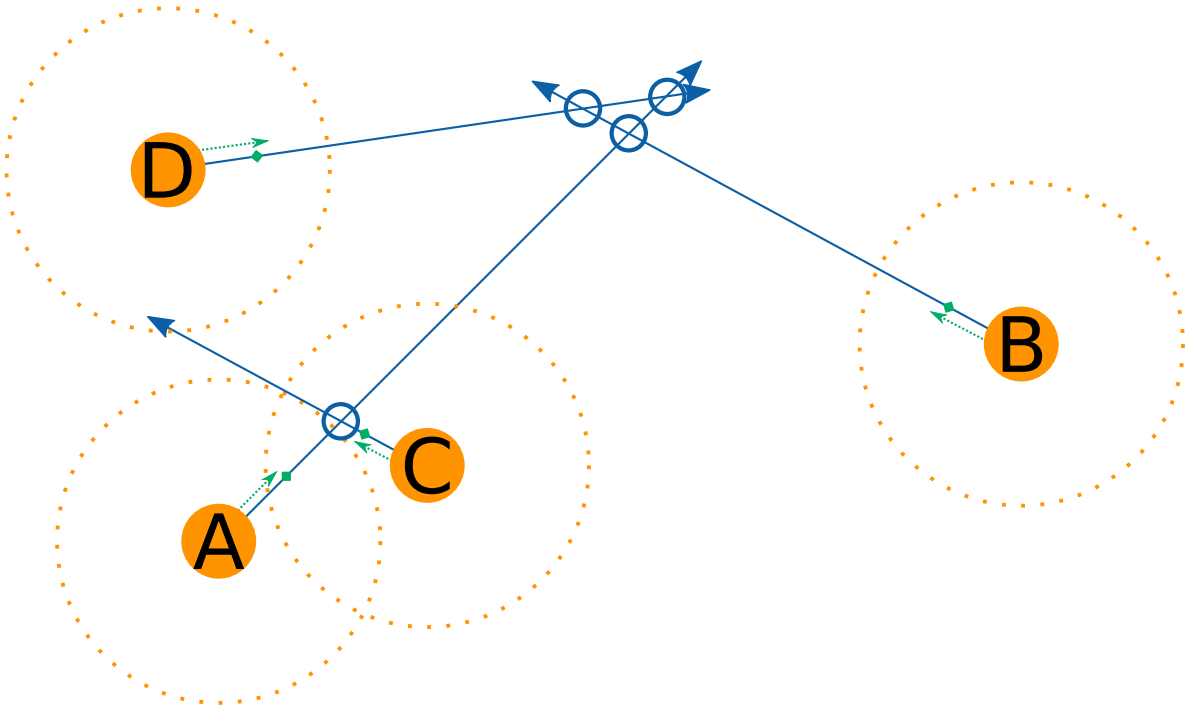

Abbildung 4.6: Beispiel der DISTANCE_MOTION Strategie mit gekennzeichneten Schnittpunkten.

Diese Segmente können zunächst wie einfache Geraden gehandhabt und auf Schnittpunkte untersucht werden. Nach der Schnittpunktanalyse gibt es zwei Sonderfälle die beachtet werden müssen:

1. Liegt der Schnittpunkt vor oder hinter einem Liniensegment ⇒ Kein Treffen der Clients!
2. Liegt der Schnittpunkt auf den Liniensegmenten, aber beide Clients erreichen zu unterschiedlichen Zeiten den Schnittpunkt ⇒ Kein Treffen der Clients!

Beispiele siehe Abbildung 4.7.

Zu 2: Die Ankunftszeit beider Clients am berechneten Schnittpunkt darf aufgrund des Übertragungsradius leicht variieren. Beispiel(vereinfacht): Angenommen Clients A und B erreichen den Schnittpunkt zu den Zeiten t_A und t_B und bewegen sich mit $v_A = 0.5m/s$ bzw. $v_B = 1.5m/s$. Geht man von einem Übertragungsradius $RANGE_{Ad-Hoc} = 10m$ aus, so wäre A 20 Sekunden nach Erreichen des Schnittpunktes 10 m vom Treffpunkt entfernt und damit nicht mehr in Übertragungsreichweite zu B. Selbiges gilt bis 20 Sekunden vor Erreichen des Treffpunktes. Das bedeutet, dass ein Nachrichtentransfer zwischen A und B möglich ist, solange B den Treffpunkt zu einem Zeitpunkt $t_A - 20 < t_B < t_A + 20$ erreicht. Hierbei wurde die Dauer der Datenübertragung zur Vereinfachung nicht berücksichtigt. Es müsste

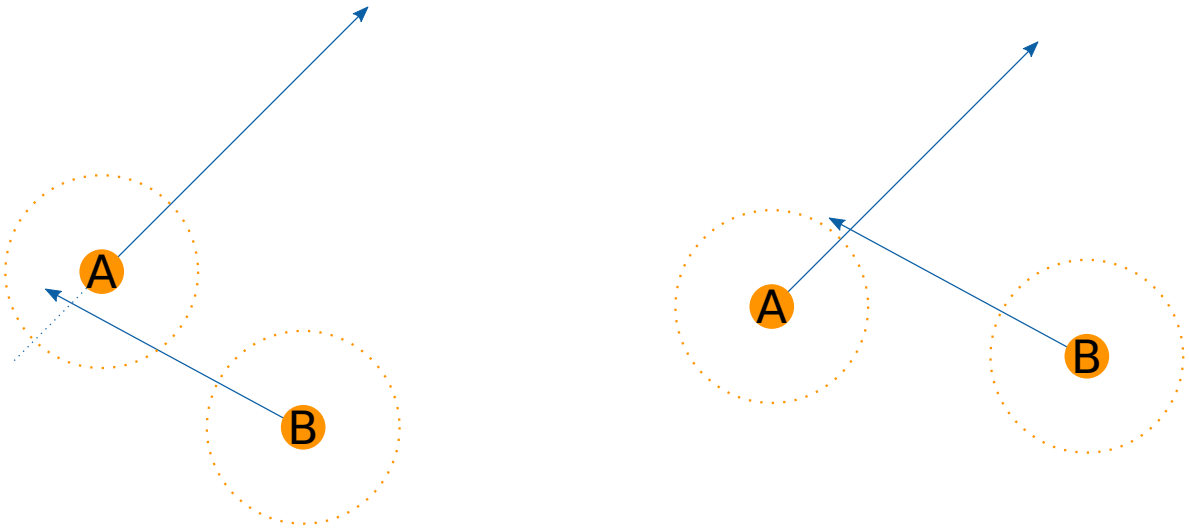

Abbildung 4.7: Beispiel der beiden Sonderfälle, in denen kein Treffen zwischen den Clients stattfindet. Links entspricht Fall 1, rechts Fall 2

daher ein weiterer Wert $t_\delta = \frac{Nachrichtengröße}{Übertragungsgeschwindigkeit}$ in der Ungleichung subtrahiert bzw. entsprechend addiert werden.

Um den tatsächlichen Ankunftszeitpunkt der beiden Clients zu bestimmen, ist jedoch noch ein weiterer Wert erforderlich. Da die berechneten Liniensegmente Start- und Endpunkt per Luftlinie verbinden (Clients sich aber anhand der Straßen bewegen müssen) kann nicht allein die Länge der Liniensegmente als Grundlage für die Dauer bis zum Erreichen des Endpunktes genügen. Daher wird die berechnete Ankunftszeit um 20% erhöht, um dem längeren Weg auf der Straße Rechnung zu tragen.

4.3.5 DISTANCE_COMBINED

DISTANCE_COMBINED verbindet die Ansätze 4.3.4 DISTANCE_MOTION und 4.3.3 DISTANCE_ENHANCED.

4.3.5.1 Funktion

Zunächst werden mögliche Zielclients mittels der DISTANCE_MOTION Strategie berechnet. Danach wird zusätzlich bestimmt wie viele Clients sich direkt oder indirekt in Übertragungsreichweite zueinander befinden (durch DISTANCE_ENHANCED). Nähere Informationen zu den jeweiligen Strategien siehe oben. Ziel des Algorithmus ist es einen Client zu bestimmen, der die höchst mögliche Überdeckung generiert und dabei die direkte Nachbarschaft sowie die in der Zukunft eintretenden Übertragungen berücksichtigt.

Zunächst wird DISTANCE_MOTION verwendet, um die Anzahl der Übertragungen die in der Zukunft stattfinden können, zu bestimmen. Dabei liegt der in der Zukunft betrachtete Bereich, wie bereits erwähnt, innerhalb der durch den Vector Extension Parameter angegebenen Zeit (siehe 4.4.4 Vector Extension). Daraufhin wird DISTANCE_ENHANCED genutzt, um die Anzahl der Ad-Hoc Übertragungen die in naher Zukunft möglich sind zu bestimmen.

4.3.6 SAME_STREET

Beim SAME_STREET Algorithmus werden Clients als erfolgversprechend eingestuft, wenn sie besonders viele andere Clients auf derselben Straße treffen und somit eine Nachricht an viele weitere Empfänger verteilt werden kann. Da keine exakte Aussage darüber getroffen werden kann wohin sich die Clients in Zukunft bewegen (und damit auch nicht darüber wie sie sich an Kreuzungen verhalten) wird im Folgenden zur Vereinfachung angenommen, dass ein Client immer auf der Straße bleibt auf der er sich aktuell bewegt. Um festzustellen, ob zwei Clients sich auf einer Straße treffen, ist es wie bei 4.3.4 DISTANCE_MOTION nötig, die (wahrscheinlich) nächste Position eines Clients zu wissen. Dabei ist es wiederum unerheblich ob die nächste Position tatsächlich korrekt ist. Es kann lediglich temporär die Anzahl der Ad-Hoc Übertragungen etwas verschlechtern, wenn mit inkorrekten Positionsdaten gerechnet wird. Daher wird wie in 4.1.5 Nächste Clientposition beschrieben, vorausgesetzt, dass der Client auch für diese Verteilungsstrategie seine nächste Position sendet.

Dies widerspricht nicht der vorherigen Aussage, dass kein exaktes Vorherbestimmen der Bewegung möglich ist, da die nächste Position weiterhin nur eine Abschätzung ist.

4.3.6.1 Funktion

Um bestimmen zu können, ob zwei Clients sich auf derselben Straße bewegen, wird zunächst der Pfad zwischen ihnen bestimmt. Hierzu besitzt der Simulator [KOK09] die Möglichkeit, zwischen zwei Knoten des Kartenmaterials den kürzesten Pfad zu berechnen. Zwei Clients werden genau dann als auf derselben Straße befindlich angesehen, wenn auf dem Pfad zwischen ihnen keine Kreuzung ist (der Pfad ist kreuzungsfrei). Ein Pfad zwischen zwei Clients entspricht der (kürzesten) Verbindung die mit den vorhandenen Straßen/Wegen möglich ist. Der Pfad ist dabei aus mehreren Elementen aufgebaut, wobei jedes Element einem Datenpunkt aus dem zugrunde liegenden Kartenmaterial entspricht. Für jedes Element des Pfades ist weiterhin bekannt, welche Datenpunkte direkt benachbart sind. Daher ist ein Pfad genau dann kreuzungsfrei, wenn jedes Element des Pfades nicht mehr als zwei Nachbarelemente hat.

Zunächst gibt es zwei Möglichkeiten, wie sich zwei Clients auf derselben Straße treffen können:

1. Sie kommen sich entgegen.
2. Beide Clients bewegen sich in dieselbe Richtung, wobei der hintere Client eine höhere Geschwindigkeit besitzt.

Fall 1 ist relativ einfach zu bestimmen. Liegt sowohl die nächste Position des ersten, als auch die nächste Position des zweiten Clients auf dem Pfad der beide verbindet, so bewegen sich beide Clients aufeinander zu und werden sich unweigerlich treffen. Das heißt, die Zeit bis zum Treffen beträgt $t = \frac{d}{v_1 + v_2}$ Sekunden, wobei d die Distanz ist, die zwischen den Clients liegt. Siehe auch Abbildung 4.8; gestrichelte Abschnitte deuten zur wahrscheinlich nächsten Position hin.

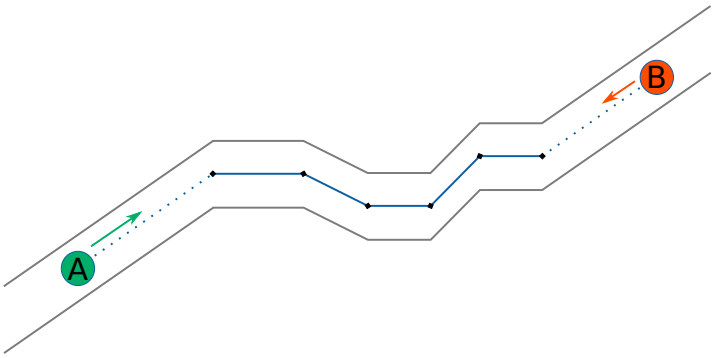

Abbildung 4.8: Fall 1: Beide Clients bewegen sich aufeinander zu.

Fall 2 wird auf eine ähnliche Weise bestimmt. Da sich beide Clients in dieselbe Richtung bewegen (d.h. sich auf einer Straße hintereinander bewegen) muss die nächste Position eines Clients außerhalb des berechneten Pfades liegen. Dabei können sich die Clients nur treffen, wenn der „hintere“ Client eine höhere Geschwindigkeit besitzt. Der hintere Client ist in diesem Fall immer der, dessen nächste Position auf dem Pfad liegt. Da sich der vordere Client durchgehend vom hinteren fortbewegt, muss eine andere Formel zur Ermittlung der Dauer bis zum Treffen genutzt werden. Angenommen v_1 und v_2 sind die Geschwindigkeiten der Clients, wobei sich Client 1 hinter Client 2 befindet: $t = \frac{d}{v_1 - v_2}$ (d entspricht der Distanz die zwischen den Clients liegt.) Siehe auch Abbildung 4.9; gestrichelte Abschnitte deuten zur wahrscheinlich nächsten Position hin.

Alle Clients müssen nun gegeneinander geprüft werden, d.h. für jeden Client wird untersucht welcher andere Client auf derselben Straße ist. Wird dabei ein Treffer gefunden, wird mittels der oberen beiden Fälle die Zeit bestimmt, zu der sich die Clients treffen. Dabei muss beachtet werden, dass Clients nur dann als überdeckt bzw. erreichbar zählen, wenn die Zeit des Zusammentreffens noch innerhalb der Gültigkeitsdauer der Nachricht liegt! Nach dieser Berechnung steht für jeden Client fest, ob und wie viele andere Clients von ihm getroffen werden und die Zeit des möglichen Treffpunkts. Der beste Empfänger (Zielclient) ist, wie bei den Strategien, nun der, der die meisten anderen Clients überdeckt. Für die Auswahl des besten Empfängers ist der Zeitpunkt zu dem die Clients erreicht werden irrelevant; lediglich die Anzahl der überdeckten Clients ist ausschlaggebend.

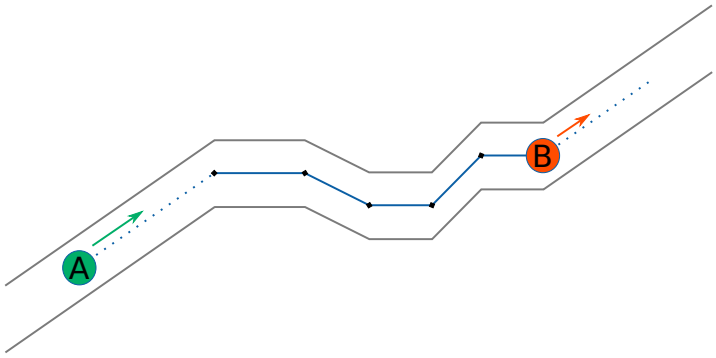

Abbildung 4.9: Fall 2: Beide Clients bewegen sich in die gleiche Rictung, dabei ist Client A schneller als Client B.

4.4 Parameter

Neben den Verteilungsstrategien gibt es noch einige Parameter anhand derer man die Verteilung der Requests bzw. die Auslastung des UMTS-Netzes verändern kann.

4.4.1 Adaptive message inquiry

„Adaptive message inquiry“ (kurz: AMI) wird genutzt um die Anzahl der versendeten Positionsnachrichten zu reduzieren. Im Prinzip sollen statt regelmäßiger Updates der Positionsinformation nur noch dann Aktualisierungen erfolgen, wenn die vorhandenen Daten zu alt bzw. zu ungenau sind. Um festzustellen, wann Positionsdaten zu alt sind, wird überprüft wie viele der berechneten Übertragungen tatsächlich eingetroffen sind. Wie bereits in 4.1.4 Zielclients und fehlende Clients erwähnt, wird für alle Algorithmen (außer RANDOM) gespeichert, wann die vom Zielclient erreichbaren Empfänger die Nachricht übermittelt bekommen. Das heißt, es kann überprüft werden, wie viele der eingegangen ACKs rechtzeitig beim Server eintrafen. Bevor für einen Request also ein weiteres Mal ein Zielclient gesucht wird, wird die Anzahl der seit der letzten Überprüfung rechtzeitig eingegangenen Requests berechnet.

Berechnung: Zunächst wird überprüft für welche Clients ein Empfangszeitpunkt vorliegt und ob dieser Zeitpunkt bereits verstrichen ist.

- Ist der Zeitpunkt bereits vorbei und es liegt ein ACK vor, so ist der Request rechtzeitig eingegangen. ⇒ ergibt die Zahl der erfolgreichen Transfers (t_{succ}).
- Ist der Zeitpunkt bereits vorbei und es liegt *kein* ACK vor, so ist der Request nicht bei dem betreffenden Client eingegangen obwohl dieser ihn hätte empfangen sollen. ⇒ ergibt die Zahl der Transfers die nicht stattgefunden haben (t_{failed}).

Daraufhin werden alle gefundenen Zeitpunkte, sowohl korrekt durchgeführte als auch die nicht stattgefundenen, aus der Liste gelöscht, damit bei der nächsten Überprüfung nur Elemente betrachtet werden, die nicht bereits geprüft wurden.

Liegt das Verhältnis der rechtzeitig eingetroffenen ACKs zur Gesamtanzahl ($t_{succ}/(t_{failed} + t_{succ})$) unter dem mit „adaptive message inquiry“ definierten Prozentsatz, kann davon ausgegangen werden, dass die Positionsdaten des Servers veraltet sind und daher viele berechnete Treffen nicht stattgefunden haben. Daher wird ein neues Positionsupdate ausgelöst. Der gewünschte Prozentsatz lässt sich in der Settings-Datei des Simulators einstellen. Soll dieser Parameter nicht genutzt werden, d.h. es soll nach einem fixen Timeout immer wieder die Position der Clients abgefragt werden, so kann man den Parameter auf -1 setzen.

Zulässiger Wertebereich: $[0, 1]$ falls Adapative Message Inquiry verwendet werden soll, ansonsten -1.

4.4.2 Reinjection Threshold

Der Reinjection Threshold Parameter (kurz: RT) soll dazu beitragen, die Anzahl der per Mobilfunk versendeten Requests weiter zu reduzieren. Durch eine Überprüfung der Anzahl der eingehenden ACKs kann bestimmt werden, wie hoch die Verteilungsrate eines Requests aktuell ist. Wurde innerhalb eines bestimmten Zeitfensters eine hohe Anzahl an ACKs empfangen, so kann man darauf schließen, dass der Request hinreichend gut verteilt wird. Ist die Anzahl der eingegangenen ACKs in diesem Zeitfenster allerdings gering, ist es offenbar nötig, einem weiteren Client den Request zu schicken, um so weitere Sender zu erhalten.

Problembeschreibung: Ursprünglich wurde alle 60 Sekunden jeder Request ein weiteres Mal in das System gegeben, um weitere Clients zu erreichen, die den Request weiterleiten können. Oft ist dies jedoch nicht nötig, da der berechnete Zielclient den Request in naher Zukunft z.B. von einem benachbarten Client erhalten hätte. In diesem Fall wäre der entstehende Traffic um den Request an den Zielclient zu senden vergebens. Um das zu vermeiden, wird die Anzahl der versendeten Datennachrichten reduziert, sofern das die aktuelle Verteilungsquote zulässt.

Lösungsansatz: Statt einer fixen Verteilung (alle 60 Sekunden) der Requests, soll ein erneutes Senden vom Server nur dann erfolgen, wenn die Zahl der eingehenden ACKs sinkt. Dazu überprüft der Server fortlaufend, wie viele ACKs in den letzten 60 Sekunden eingegangen sind. Liegt das Verhältnis von eingegangenen ACKs zur Anzahl der noch fehlenden Clients unter einer gewissen Grenze (die mit „Reinjection Threshold“ festgelegt wird), wird der Request erneut an einen Zielclient geschickt. Dieser wird wiederum vom aktuell verwendeten Algorithmus berechnet.

Zulässiger Wertebereich: $[0, 1]$ Wobei 1 bedeutet, dass von allen fehlenden Clients ACKs erwartet wurden. Da dies aber unrealistisch ist, entspricht 1 also dem regelmäßigen (im 60 Sekundentakt) Senden der Requestnachricht. Wird der Parameter zu hoch gewählt, (z.B. größer als 0.3) so sinkt die erreichte Überdeckung leicht, da oft an einen Client der Request versandt wird, der diesen in naher Zukunft durch einen anderen Client erhalten hätte. Insgesamt wird dadurch also die Zahl der Ad-Hoc Übertragungen um eins reduziert. Dieser Fall tritt häufiger ein, je höher der Parameter gesetzt wird.

4.4.3 Radius reduction

Durch den Radius Reduction Parameter (kurz: RR) soll bei der Verwendung der DISTANCE_ENHANCED Strategie der Fall, dass Clients in äußeren Leveln Requests nicht erhalten, vermieden werden. Dazu wird der Übertragungsradius der weit außen liegenden Clients kleiner simuliert.

Problembeschreibung: Wie bereits bei DISTANCE_ENHANCED beschrieben (siehe 4.3.3.2 Problem), existiert das Problem, dass äußere Clients Nachrichten nicht erhalten. Dies liegt daran, dass die Zeit, in der die ausgewählten Clients in Übertragungsreichweite sind, nicht ausreicht um eine Nachricht von Innen (Zielclient) nach Außen (überdeckte Clients) zu transportieren. Insbesondere tritt der Fall dann auf, wenn die Clients der höheren Level (also die äußeren Clients) bereits weit von ihrem Sendeclient (also ein Client einen Level tiefer) entfernt sind. Dann werden sie zwar zur Überdeckung hinzugerechnet (sind also im Überdeckungsbaum), sind meist aber nur wenige Sekunden innerhalb der Übertragungsreichweite eines Senders, der ebenfalls einige Sekunden warten muss bis die Nachricht ihn erreicht.

Lösungsansatz: Für die Berechnung der Überdeckung wird mit einem kleineren Übertragungsradius der höheren Level gerechnet. Genauer bedeutet das, dass der Übertragungsradius mit steigendem Level um einen festen Wert kleiner gewählt wird. Als Standardwert wird eine Reduktion um drei Meter genutzt. Geht man als Übertragungsradius von zehn Metern aus (weil beispielsweise Bluetooth genutzt wird) gilt bei Level 0 (Zielclient) also der volle Übertragungsradius. Für alle vom Zielclient überdeckte Clients (Level 1) geht der Radius mit sieben Metern in die Bestimmung ein. In Level 2 beträgt der, für die Berechnung genutzte, Übertragungsradius dann nur noch vier Meter. Da der Radius in Level 3 bereits nur noch einen Meter beträgt ist offensichtlich, dass bei dieser Wahl des Reduzierungswerts ein weiterer Level nicht mehr möglich ist. Damit Clients in die Überdeckungsberechnung einfließen, müssen sie also näher an einem überdeckten Client sein, je weiter sie sich vom Zielclient entfernen.

Dadurch wird sichergestellt, dass besonders Clients in einem höheren Level sich länger im Übertragungsradius ihres Übermittlers aufhalten. Damit werden die berechneten Überdeckungen genauer, es gibt also weniger Abweichung zwischen Anzahl der berechneten Überdeckungen und der tatsächlich eingehenden ACKs der Clients. Durch die

höhere Genauigkeit des Algorithmus steigt letztendlich die Anzahl der versendeten Ad-Hoc Nachrichten, womit wiederum der Traffic in dem Mobilfunknetz gesenkt werden kann.

Zulässiger Wertebereich: $[0, \infty)$ Wobei nur ganzzahlige Werte zulässig sind. Ein Wert der größer ist als der Übertragungsradius des verwendeten Kommunikationsinterface würde dazu führen, dass der Algorithmus wie DISTANCE agiert. Das heißt für einen Client (Zielclient) könnte bestimmt werden welche Clients überdeckt sind, aber ein weiterer Level wird nicht beachtet, da für die überdeckten Clients ein Übertragungsradius von weniger als null Metern simuliert wird.

4.4.4 Vector Extension

Wie bereits unter 4.3.4 DISTANCE_MOTION beschrieben, repräsentiert „Vector Extension“ (kurz: VE) die Verlängerung des Richtungsvektors um einen bestimmten Faktor. Wird dabei beispielsweise der Faktor „120“ gewählt, entspricht das der Annahme, dass der Client sich weitere 120 Sekunden auf der berechneten Richtung bewegt. Veränderungen an der Länge des Richtungsvektors verändern direkt die Zahl der berechneten Datentransfers. Wird der Verlängerungsvektor zu kurz gewählt, so kann der Fall eintreten, dass ein Client zum Ziel gewählt wird der wenige andere Clients überdeckt, diese aber sehr früh trifft. Das hat zur Folge, dass ein Client der wesentlich mehr Clients treffen könnte, das aber zu einem (etwas) späteren Zeitpunkt, nicht gewählt wird da in der direkten, überprüften Umgebung weniger Schnittpunkte sind. Andererseits kann auch der Fall eintreten, dass ein Client gewählt wird der in großer Entfernung viele andere Clients trifft, aber aufgrund der Ungenauigkeit dieses Ansatzes diese nie erreicht. Dann würden bessere Ziele, die weniger Clients überdecken diese aber früher und damit wahrscheinlicher treffen, ignoriert. Den optimalen Wert für die Verlängerung des Richtungsvektors zu finden ist Teil der Evaluation.

Zulässiger Wertebereich: $[0, \infty)$ Wobei nur ganzzahlige Werte zulässig sind. Ein Wert der größer ist als die Nachrichtenlebenszeit (in Sekunden) würde bedeuten, dass Treffpunkte berechnet werden, bei deren Erreichen die entsprechende Nachricht bereits nicht mehr gültig ist.

KAPITEL 5

EVALUIERUNG

5.1 ONE Simulator

Zur Simulation der Bewegungen und Datentransfers aller Clients wird der Simulator „One“ [KOK09] genutzt. Dieser simuliert in erster Linie die Bewegungen und Datentransfers von Clients. Alle Clients verschicken eingehende Nachrichten automatisch per Ad-Hoc Netzwerk an alle verfügbaren Nachbarn. Der Datentransfer wird abgebrochen, sobald die Distanz zwischen zwei Clients größer als die Reichweite des verwendeten Kommunikationsinterfaces ist, oder ein Client die entsprechende Nachricht bereits gespeichert hat (also die Nachricht in einem Puffer vorliegt).
Um die Bewegung des Clients realitätsnah darstellen zu können, wird dem Simulator Kartenmaterial zugewiesen. Das in der folgenden Auswertung genutzte Kartenmaterial entspricht einem Ausschnitt von Helsinki und ist etwa $16,2 km^2$ groß. Damit ist es dann möglich, dass sich die Clients nicht willkürlich, sondern nur noch auf den vorhandenen Straßen fortbewegen. Clients bewegen sich auf dem Kartenmaterial allerdings nicht rein zufällig, sondern folgen einem (zu Beginn einer Simulation generierten) Pfad. Diese Pfade sind zufallsgeneriert, bleiben jedoch bei jeder Simulation gleich. Das heißt, dass sich ein bestimmter Client in jeder Simulation auf demselben Pfad bewegt. Dies ist notwendig um gleiche Grundbedingungen zu schaffen und um die entstehenden Ergebnisse sinnvoll vergleichen zu können.

5.2 Simulation

5.2.1 Aufbau

Vor dem Start einer Simulation müssen einige Optionen gesetzt werden, damit das Programm ordnungsgemäß funktionieren kann. Dazu zählen unter anderem die unter 4.4 Parameter genannten Parameter sowie der Algorithmus, anhand dessen die Verteilung der Nachrichten erfolgen wird. Weiterhin muss spezifiziert werden, wie groß die zu versendenden Nachrichten sind und wie viele Clients sich in dem System aufhalten. Die verwendeten

Nachrichtengrößen für Requests betragen hier $0.25MB$ und $0.5MB$. Größere Nachrichten wurden nicht simuliert, da das zugrunde liegende Mobilfunknetz nicht genug Bandbreite bietet. Bei der Anzahl der simulierten Clients würden die nötigen Nachrichtenlebensdauern dadurch stark ansteigen. Die Größe eines Request, sowie die Anzahl der Clients im System sind nicht für alle Simulationen identisch. Alle simulierten Kombinationen stehen unter 5.2.2 Parameterkombinationen. Während einer laufenden Simulation wird die verwendete Strategie nicht gewechselt, d.h. wurde die Simulation beispielsweise mit RANDOM gestartet, so wird für alle Requests RANDOM als Verteilungsstrategie verwendet. Selbes gilt ebenfalls für Parameter. Ein zu Beginn der Simulation gesetzter Wert bleibt erhalten, bis die Simulation abgeschlossen ist.

Der Simulator wird mit einer zu simulierenden Zeit initialisiert, die angibt wie viele Sekunden simuliert werden sollen. Wichtig ist dabei, dass der Simulator nicht in Echtzeit simuliert. Das heißt, eine Simulationsdauer von einer Stunde kann unter Umständen wesentlich mehr als eine Stunde Rechenzeit beanspruchen. Die zu simulierende Zeit beträgt 7200 Sekunden (zwei Stunden) mit einer Ausnahme bei der SAME_STREET Verteilungsstrategie. Da der Algorithmus zum Berechnen kürzester Pfade zwischen zwei Clients extrem rechenintensiv ist, wurde hier die Simulationsdauer auf 3600 Sekunden reduziert.

Wie unter 4.1.2 Regulärer Ablauf (mit adaptiver Verteilung) beschrieben, wird während der Simulation alle 200 Sekunden (Simulationszeit) ein neuer Request erstellt, um den Datendurchsatz etwas realistischer zu gestalten. Auch diese Zeit wurde bei Simulationen bezüglich SAME_STREET verändert und beträgt dort 400 Sekunden. Damit werden pro Simulation weniger Requests erstellt. Das führt dazu, dass weniger Zielclients berechnet werden müssen, da pro Request periodisch neue Zielclients bestimmt werden (siehe auch 4.1.2 Regulärer Ablauf (mit adaptiver Verteilung)). Dies trägt ebenfalls dazu bei, die Dauer dieser besonders rechenintensiven Simulation etwas zu verkürzen.

Eine Besonderheit des Simulators betrifft die „Panic Zone“ (siehe 4.1.2 Regulärer Ablauf (mit adaptiver Verteilung)). Dort haben Nachrichten, die per Mobilfunk gesendet werden bisher noch keine Laufzeit. Das heißt wenn eine Nachricht, unabhängig ihrer Größe, per Mobilfunk versendet wird, erreicht diese sofort ihr Ziel. Daher erhalten alle fehlenden Clients zu Beginn der Panic Zone den entsprechenden Request.

5.2.2 Parameterkombinationen

Die in 4.4 Parameter beschriebenen Parameter können über eine Einstellungsdatei dem Simulator direkt übergeben werden. Die Einstellungsdatei wird bei jedem Start einer Simulation neu eingelesen und kann aus dem Quellcode des Simulators bequem ausgewertet werden. Durch das Generieren mehrerer Einstellungsdateien, lassen sich so viele Simulationen automatisiert erstellen. Für eine Übersicht der verwendeten Parameter siehe 5.1 Parameterkombinationen.

	RANDOM	DISTANCE	DISTANCE_ENHANCED	DISTANCE_MOTION	DISTANCE_COMBINED[a]	SAME_STREET
Anzahl Clients	300, 500, 750	300, 500, 750	300, 500, 750	300, 500, 750	300, 500, 750	300, 500
Dateigröße (MB)	0.25, 0.5	0.25, 0.5	0.25, 0.5	0.25, 0.5	0.25, 0.5	0.5
Adaptive Message Inquiry	-	-1, 0.5, 0.75, 0.9	-1, 0.5, 0.75, 0.9	-1, 0.5, 0.75, 0.9	-1, 0.5, 0.75, 0.9	-1, 0.5, 0.75, 0.9
Reinjection Threshold	0.02, 0.05, 0.1, 0.2	0.02, 0.05, 0.1, 0.2	0.02, 0.05, 0.1, 0.2	0.02, 0.05, 0.1, 0.2	0.02, 0.05, 0.1, 0.2	0.02, 0.05, 0.1
Radius Reduction	-	-	0, 4, 8	-	0, (4), (8)	-
Vector Extension (s)	-	-	-	30, 60, 120, 180, 240	30, 60, (120), 180, 240	-

Tabelle 5.1: Diese Tabelle zeigt die während der Simulation genutzten Parameterkombinationen.

[a] DISTANCE_COMBINED wurde wegen der hohen Zahl der Parameter nicht mit allen Kombinationen simuliert. Um die Auswirkungen des „RadiusReduction“ Parameters zu untersuchen, wurde „VectorExtension“ fest mit 120 belegt. Entsprechend wurde beim Test verschiedener „VectorExtensions“ ebenfalls „RadiusReduction“ fest gewählt (Wert: 4 und 8).

5.3 Erwartungen

Mit dem Wissen über die Funktion der verschiedenen Algorithmen ergeben sich einige Erwartungen bzgl. der Ergebnisse der Simulation. Dies betrifft sowohl die Menge der erreichten Ad-Hoc Übertragungen eines Algorithmus, als auch die dafür erforderlichen Positionsnachrichten.

RANDOM Die niedrigste Überdeckung (d.h. am wenigsten Ad-Hoc Übertragungen) hat wahrscheinlich RANDOM, da dabei keinerlei Positionsinformationen genutzt werden. Außerdem wird die Überdeckungsrate vermutlich stärker variieren als bei den anderen Algorithmen. Denn neben den sicherlich häufig vorkommenden niedrigen Überdeckungen, kann es, zufallsbedingt, auch zu vielen Ad-Hoc Übertragungen kommen, wenn per Zufall eine gute Folge von Zielclients getroffen wird.

DISTANCE Vermutlich sehr gute Überdeckungswerte und neben DISTANCE_ENHANCED mit die besten. Dieser Ansatz hat sich bereits in [BDR12] unter dem Namen „Static Coverage“ als gut erwiesen. Wahrscheinlich wird dieser Ansatz vergleichsweise viele Positionsnachrichten erzwingen, da die Clients sich fortlaufend bewegen und sich von einer einmal bekannten Position schnell entfernen. Ist ein Client nicht mehr in der Nähe der von ihm bekannten Position, kann der Algorithmus wahrscheinlich auch keine korrekten Ergebnisse mehr berechnen.

DISTANCE_ENHANCED Die Überdeckungswerte werden wahrscheinlich etwas besser sein als die des DISTANCE Algorithmus (insbesondere dann besser, wenn der Radius Reduction Parameter genutzt wird). Das liegt in erster Linie daran, dass die Vorgehensweisen der Algorithmen gleich sind, lediglich der durch DISTANCE_ENHANCED betrachtete Bereich ist größer. Dadurch können Clients gefunden werden, die mehr Nachbarn direkt oder indirekt überdecken und so die Nachricht häufiger per Ad-Hoc Netzwerk versandt werden kann. Die Anzahl der nötigen Positionsnachrichten wird vermutlich ähnlich wie bei DISTANCE oder etwas höher sein. Mehr Positionsnachrichten könnten erforderlich sein, da für mehr Clients ein mögliches Treffen berechnet wird, daher auch eine höhere Zahl korrekter Übertragungen stattfinden muss. Da sich aber die äußeren Clients unter Umständen schnell von einem möglichen Sender wegbewegen, ist es unwahrscheinlicher, dass diese die Nachricht erhalten (dieser Effekt sollte reduziert werden, wenn der Parameter Radius Reduction genutzt wird).

DISTANCE_MOTION Wahrscheinlich wird die Überdeckung etwa gleich hoch wie die des DISTANCE Algorithmus sein. Dies lässt sich dadurch begründen, dass wie bei DISTANCE, Clients als Ziel bevorzugt werden, die viele andere Clients aus verschiedenen Richtungen treffen. Entgegengesetzt dazu wäre der SAME_STREET Ansatz, bei dem Clients bevorzugt werden, die viele Schnittpunkte mit anderen Clients derselben Bewegungsrichtung haben. Die Anzahl der nötigen Positionsnachrichten ist schwierig abzuschätzen. Wahrscheinlich

sind weniger Nachrichten als bei den übrigen Algorithmen nötig, da durch die Berechnung der Richtungsvektoren kleinere Abweichungen von der Position nicht so stark ins Gewicht fallen. Der Richtungsvektor ändert sich dadurch unter Umständen nicht so stark, wodurch auch mit alten Positionsdaten korrekte Ergebnisse berechnet werden können.

DISTANCE_COMBINED Da dies eine Zusammenführung der beiden vorherigen Algorithmen ist, ergeben sich daraus ähnliche Erwartungen. Die Überdeckungswerte sollten denen von DISTANCE_ENHANCED entsprechen oder besser sein. Die Anzahl der Positionsnachrichten, liegt dagegen eher auf Niveau des DISTANCE_MOTION Ansatzes.

SAME_STREET Die Überdeckungswerte werden vermutlich niedriger als die des DISTANCE Algorithmus sein. Hauptgrund ist wahrscheinlich, dass hierbei Clients als Ziel gewählt werden, die ähnliche Bewegungsrichtungen haben. Das heißt, Clients bewegen sich entweder hintereinander (was wenige neue Ad-Hoc Übertragungen bringt) oder aufeinander zu (was bis zum Treffpunkt ebenfalls keine neuen Ad-Hoc Übertragungen bewirkt). Insgesamt verteilt sich ein Request also hauptsächlich entlang einer Straße, nicht wie bei anderen Algorithmen in verschiedene Richtungen. Die Anzahl der nötigen Positionsnachrichten wird wahrscheinlich geringer als bei anderen Algorithmen ausfallen. Nimmt man an, dass Clients auf einer Straße bleiben, ist die exakte Position auf der Straße nicht mehr so bedeutend um mögliche Treffpunkte zu bestimmen.

5.4 Vergleich

Es folgt die Auswertung jedes einzelnen Algorithmus, sowie zuletzt eine Gegenüberstellung der Ergebnisse. Für alle folgenden Auswertungen gilt, der Anteil der Ad-Hoc Übertragungen erhöht sich, je mehr Clients im System sind. Daher sind alle Auswertungen mit der maximalen Anzahl von Clients (750) aufgeführt. Alle folgenden Angaben bzgl. der Einsparung von Traffic beziehen sich immer auf die Einsparung des insgesamt erzeugten Traffics. Welche Leistungseinbußen durch das zusätzliche Versenden vieler, relativ kleiner Positionsnachrichten entsteht, kann hier nicht abgeschätzt werden.

5.4.1 RANDOM

Die Ergebnisse der Simulation entsprechen relativ genau den Erwartungen. Die erreichten Überdeckungswerte schwanken deutlich (zwischen 20% und 50% hauptsächlich). Dateigrößen machen bei diesem Algorithmus keinen Unterschied, daher werden die Ergebnisse nur bzgl. einer Dateigröße gezeigt. Weiterhin interessant ist, dass eine Veränderung des Reinjection Threshold Parameters keine signifikante Änderung der Überdeckung bewirkt (siehe dazu Abbildung 5.1).

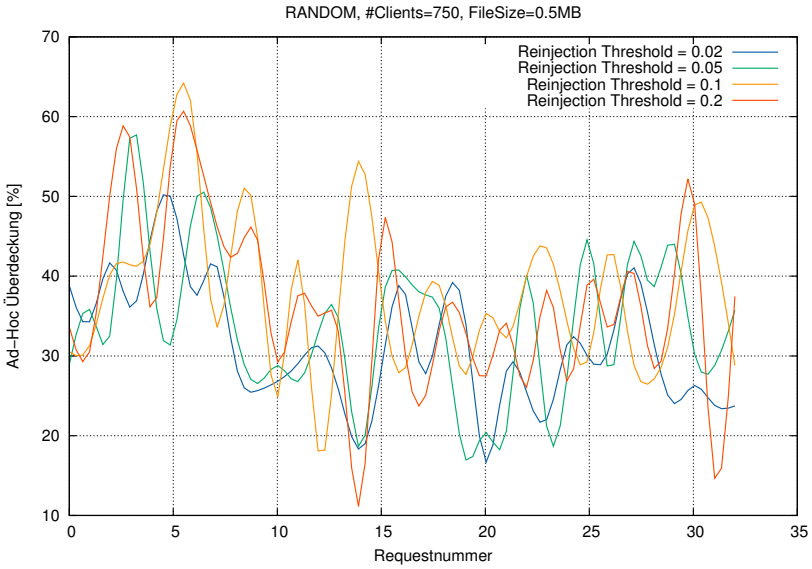

Abbildung 5.1: Überdeckungen des RANDOM Algorithmus anhand verschiedener Reinjection Threshold Werte. Die Anzahl der Clients beträgt 750 und die Dateigröße 0,5MB

Beispielhaft soll für diesen Algorithmus gezeigt werden, wie stark sich eine Veränderung der Anzahl der Clients auf die erreichten Überdeckungswerte auswirkt. Deutlich sichtbar ist, dass mit steigender Anzahl an Clients prozentual auch mehr Übertragungen stattfinden. Siehe dazu Abbildung 5.2

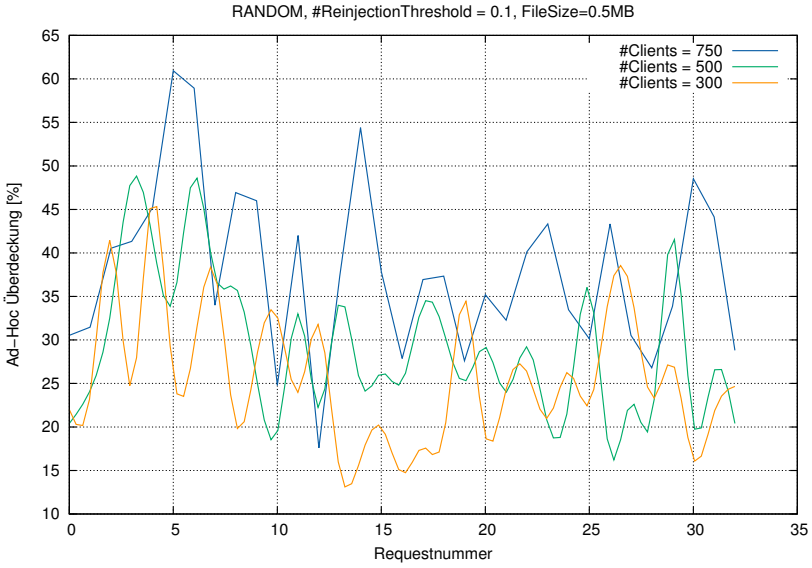

Abbildung 5.2: Überdeckungen des RANDOM Algorithmus anhand verschiedener Clientzahlen.

5.4.2 DISTANCE

Als Dateigröße wurde für die folgende Darstellung $0.5MB$ gewählt. Der Unterschied zu den Ergebnissen der Simulation mit $0.25MB$ großen Requests ist gering. Der Großteil der Überdeckungen (fast 80%) ist zwischen 45% und 60% wie auch in der folgenden Darstellung.

Adaptive Message Inquiry

Der Wert des Adaptive Message Inquiry Parameters hat beinahe keine Auswirkungen auf die Überdeckungswerte gezeigt. Wurde der AMI Parameter genutzt (d.h. $AMI \in [0,1]$) waren die Ergebnisse gleich, unabhängig vom Wert des Parameters. Lediglich ein minimaler Unterschied in der Überdeckung ist feststellbar, wenn der Parameter nicht genutzt wird (d.h. $AMI = -1.0$).
Das angestrebte Ziel (Reduzierung der Positionsnachrichten) konnte allerdings erreicht werden. Ohne die Verwendung des AMI Parameters wurden 510000 Positionsnachrichten versendet. Mit der Verwendung wurden während der Simulation nur 4500 Positionsnachrichten versendet. Davon je 2250 Location Update Inquiry (also Positionsanfragen vom Server) und 2250 Antworten bzw. Positionen der Clients an den Server. Damit liegt die Anzahl der benötigten Positionsnachrichten deutlich unter den Erwartungen. Umgerechnet auf die Anzahl der simulierten Clients (750) ergibt das drei Positionsaktualisierungen während der Simulation. Entgegen der Erwartungen reichen also bereits relativ alte Positionsangaben aus, um gute Ergebnisse mit diesem Algorithmus zu erzielen. Weiterhin lässt sich daraus schließen, dass „Nachbarschaften“, also Gruppen benachbarter Clients, mit hoher Wahrscheinlichkeit während der Simulation benachbart bleiben. Andernfalls könnten die vom Algorithmus für einen Client berechneten Ziele, den Request nicht während der richtigen Zeitspanne erhalten. Schließlich wird bei der Berechnung eines Zielclients gespeichert, welche Clients durch diesen, den Request erhalten und zu welchem Zeitpunkt dies spätestens geschehen musste.

Reinjection Threshold

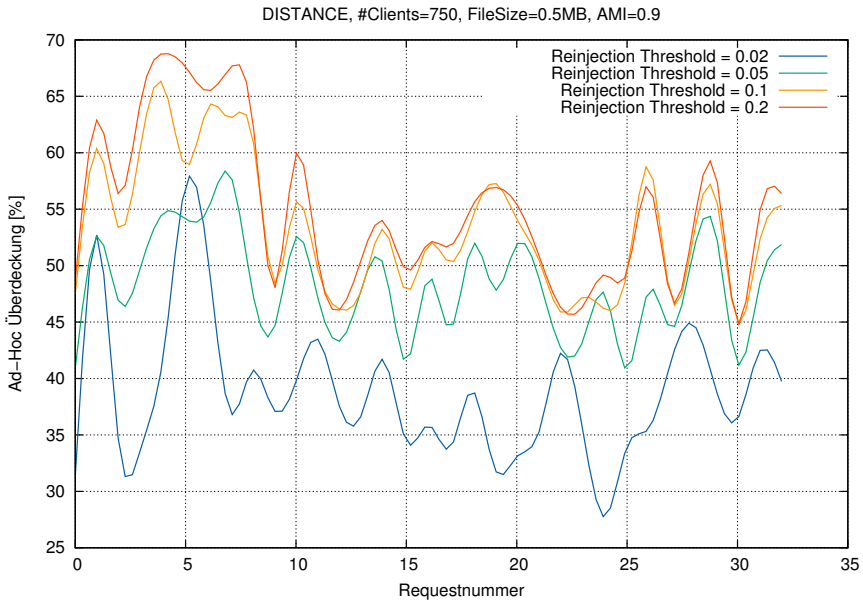

Abbildung 5.3: DISTANCE Algorithmus mit verschiedenen Werten des Reinjection Threshold Parameters.

Gut zu sehen ist hier, dass mit steigendem Reinjection Threshold bis zu einem Wert von 0.1 auch die erreichte Überdeckung deutlich steigt. Ab einem Wert von 0.2 gibt es keine signifikanten Veränderungen mehr, weshalb auf die Darstellung und Simulation dieser Werte verzichtet wurde.

Traffic Einsparung

Wie bereits erwähnt wurden je 2250 LocationUpdateInquiry sowie die entsprechenden Positionsantworten versendet. Das entspricht einem zusätzlichen Traffic von: $T_{ges} = 4500 * 250byte \approx 1099Kbyte$. Wobei $250Byte$ der Größe einer Nachricht entspricht (siehe 4.2.1 Allgemeines) Da mit einer Requestgröße von $0.5MB$ simuliert wurde, folgt daraus, dass bereits die Auslagerung von drei Datennachrichten aus dem UMTS Netz hin zu Ad-Hoc Übertragungen genügt, um den Traffic, der für einen Request erzeugt wurde zu senken. Für eine Requestgröße von $0.25MB$ müssten entsprechend mindestens fünf Nachrichten ausgelagert werden. Wie aus dem Diagramm ersichtlich ist, werden diese Zahlen bei weitem überschritten. Geht man von einer durchschnittlichen Auslagerung zwischen 45% und 55% der Nachrichten aus, wurden zwischen 337 und 412 Datennachrichten pro Request eingespart. Das entspricht einer Trafficeinsparung zwischen 168 und $206MB$ (pro Request) gegenüber dem zusätzlichen Traffic von etwa $1MB$ während der gesamten Simulation für Positionsnachrichten. Je größer Datennachrichten werden, desto größer wird offensichtlich

auch die Einsparung an Traffic. Die tatsächliche Trafficeinsparung ist natürlich etwas niedriger, da der zusätzliche Traffic der Positionsnachrichten abgezogen werden muss, in diesem Fall ist dies allerdings vernachlässigbar.

5.4.3 DISTANCE_ENHANCED

Als Dateigröße für die folgenden Auswertungen wurde $0.5MB$ gewählt. Der Unterschied der Überdeckung zu $0.25MB$ großen Nachrichten ist vernachlässigbar. Der Großteil der erreichten Überdeckungen (72%) liegt zwischen 45 und 60%.

Adaptive Message Inquiry

Wie bei dem vorherigen Algorithmus, ist der interessanteste Punkt der, dass durch die verschiedenen Werte des AMI Parameters quasi kein Unterschied in der Überdeckung entsteht. Lediglich die Verwendung ($AMI \in [0,1]$) bzw. Nicht-Verwendung ($AMI = -1$) ergibt einen Unterschied in den erreichten Überdeckungswerten. Ohne die Verwendung des AMI Parameters können etwas höhere Spitzenwerte bei den Überdeckungen erreicht werden, im Mittel ist der Unterschied jedoch gering. Zudem ist die Zahl der erforderlichen Positionsnachrichten bei der Verwendung deutlich geringer (10500) gegenüber der Simulation ohne AMI Parameter (508500). Die Anzahl der Positionsnachrichten liegt damit unter den Erwartungen. Im Vergleich zur DISTANCE Strategie ist die erwartete leichte Steigerung der nötigen Positionsnachrichten eingetroffen (4500 auf 10500).

Reinjection Threshold

Auch hier sind die Ergebnisse sehr ähnlich zum vorherigen Algorithmus. Eine hohe Steigerung der Überdeckung lässt sich von $RT = 0.02$ auf $RT = 0.05$ und von $RT = 0.05$ auf $RT = 0.1$ feststellen. Die Änderung von $RT = 0.1$ auf $RT = 0.2$ bewirkt dagegen nur noch eine geringe Erhöhung der Überdeckung.

Radius Reduction

Der Radius Reduction Parameter (RR) wird genutzt, um speziell bei diesem Algorithmus den Fall, dass Clients in äußeren Leveln eine Nachricht nicht erhalten, zu vermeiden. Die Auswirkungen des Radius Reduction Parameters sind in dieser Simulation jedoch sehr gering (siehe Abbildung 5.4). Durch die Verwendung eines relativ schnellen Kommunikationsinterfaces zwischen den Clients (hier: WiFi Direct) können Daten sehr schnell auch in äußere Levels transportiert werden. Dadurch verringert sich die Wirksamkeit des Parameters. Die durchschnittlichen Überdeckungen betragen: 55.7%, 56.5% und 55.4% für Radius Reduction Werte von 0, 4 und 8.

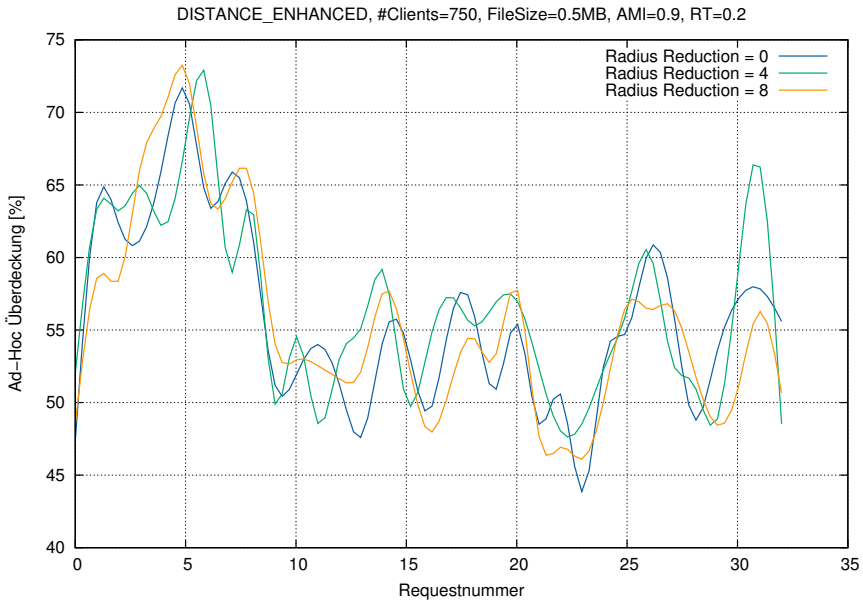

Abbildung 5.4: DISTANCE_ENHANCED Algorithmus mit verschiedenen Werten des Radius Reduction Parameters.

Wesentlich wirksamer wird der Parameter, wenn größere Nachrichten übertragen werden müssen oder die Kommunikation zwischen Clients sehr langsam ist. Größere Datennachrichten wurden hier allerdings nicht simuliert, da das zugrunde liegende Mobilfunknetz relativ geringe Übertragungsgeschwindigkeiten bietet und bei größeren Nachrichten die Nachrichtenlebensdauer daher stark ansteigen würde.

Auf die gemessenen Überdeckungswerte hat dieser Parameter also wenig Einfluss. Die Anzahl der entstehenden Positionsnachrichten verändert sich jedoch drastisch. Sind bei einem Wert von $RR = 0$ 91500 Nachrichten nötig, so sind bei $RR = 4$ 10500 bzw. bei $RR = 8$ nur 9000 Nachrichten nötig. Das heißt, wählt man als Radius Reduction vier Meter (also $RR = 4$) erhöht sich zwar die Überdeckung nur minimal, die Positionsnachrichten werden jedoch auf ein Neuntel ihrer ursprünglichen Menge (bei $RR = 0$) reduziert.

Traffic Einsparung

Die 10500 zusätzlich nötigen Positionsnachrichten (wiederum zu Hälfte Location Updates und Location Update Inquiries) entsprechen einem zusätzlichen Traffic von ca. 2.5 MB ($10500 * 250byte$). Dieser zusätzliche Traffic fällt über die gesamte Laufzeit der Simulation (zwei Stunden) an, nicht pro Request. Bei einer Requestgröße von 0.5 MB ist der zusätzliche Traffic also nach der Ad-Hoc Übertragung von fünf Requests kompensiert. Legt man die durchschnittliche Auslagerung von 56.5% zugrunde, so werden pro Request etwa

423 Nachrichten Ad-Hoc übertragen. Das bedeutet, das tendenziell schon durch die Auslagerungen während des ersten Requests der zusätzliche Traffic der Positionsnachrichten vernachlässigbar ist.

5.4.4 DISTANCE_MOTION

Alle folgenden Auswertungen der Simulationen beziehen sich auf eine Dateigröße von $0.5MB$. Der Unterschied zur Simulation mit $0.25MB$ ist nur marginal und wird daher nicht näher betrachtet.

Adaptive Message Inquiry

Die Simulationen haben gezeigt, dass es keinen Unterschied macht welcher der drei positiven simulierten Werte des AMI Parameters verwendet wird. Positive AMI Werte produzieren, abhängig von den übrigen Parametern Reinjection Threshold und Vector Extension, immer die gleichen Überdeckungskurven. Eine andere Überdeckungskurve kommt nur dann zustande, wenn ohne den AMI Parameter (d.h. $AMI = -1$) simuliert wird. Dadurch ist die dynamische Versendung von Positionsnachrichten deaktiviert, d.h. es werden etwas mehr Positionsnachrichten versendet (siehe dazu Abbildung 5.5). Wesentlich bessere Ergebnisse lassen sich dadurch allerdings nicht erzielen. Für den Vector Extension Parameter wurde als Wert 120 gewählt, da die Anzahl der entstehenden Positionsnachrichten relativ gering ist, die Überdeckungswerte aber nicht wesentlich schlechter sind, als bei anderen Werten.

Ohne die Verwendung des AMI Parameters (d.h. mit regelmäßigen Positionsaktualisierungen) werden 510000 Positionsnachrichten über die gesamte Dauer der Simulation versendet, davon je eine Hälfte vom Server zu den Clients bzw. von den Clients zurück zum Server. Mit $AMI = 0.9$ kann diese Zahl auf 396000 reduziert werden. Trotzdem liegt diese Zahl weit über den Erwartungen, da für DISTANCE_MOTION eigentlich eine geringe Anzahl von Positionsnachrichten erwartet wurde.

Reinjection Threshold

Veränderungen des Reinjection Threshold Parameters verhalten sich wie bei den vorherigen Algorithmen. Deutliche Steigerungen der Überdeckung sind bis zu einem Wert von 0.1 zu sehen. Die Erhöhung von $RT = 0.1$ auf $RT = 0.2$ bringt dagegen nur noch eine geringe Steigerung der Überdeckung. Der optimale Wert ist hier $RT = 0.2$. Bereits vor der Simulation hatte sich herausgestellt, dass höherer Werte keine Verbesserung mehr bringen.

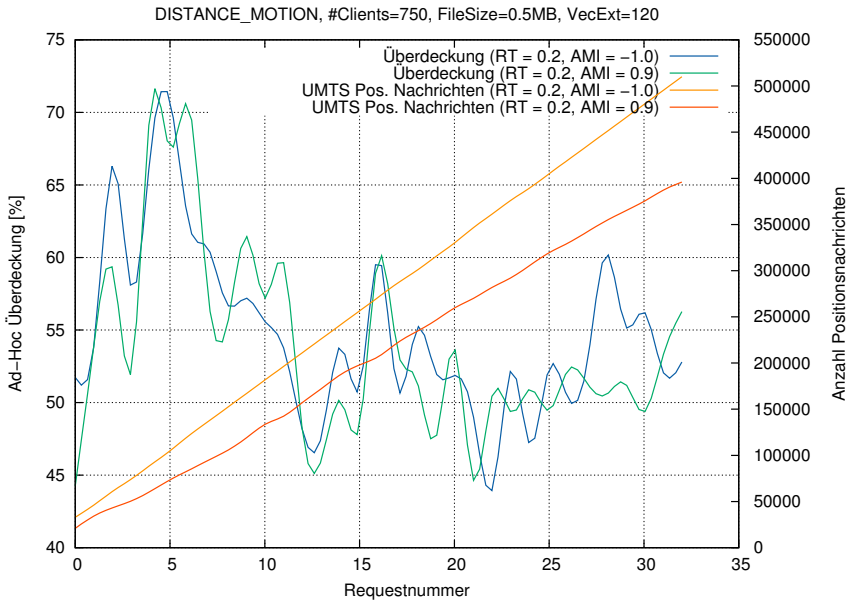

Abbildung 5.5: DISTANCE_MOTION Algorithmus mit verschiedenen Werten des Adaptive Message Inquiry Parameters. Die beiden Geraden zeigen wie die Anzahl der Positionsnachrichten über die Zeit wächst.

Vector Extension

Interessanterweise, hat die Wahl des Vector Extension Parameters keinen großen Einfluss auf die Überdeckungswerte. Insgesamt schwanken die erreichten Überdeckungswerte relativ stark, wobei die Schwankungen bei allen simulierten Werten meist zum selben Zeitpunkt stattfinden. Zu erkennen ist dies in Abbildung 5.6. Die Anzahl der nötigen Positionsnachrichten (hier nicht dargestellt) zeigt ein ähnliches Verhalten. Eine leichte Steigerung von $VE = 30$ zu $VE = 60$ ist dabei zu erkennen, danach allerdings nur noch marginale Veränderungen.

Traffic Einsparung

Wie unter 5.4.4 Adaptive Message Inquiry beschrieben, werden 396000 bzw. 510000 zusätzliche Nachrichten im Laufe der Simulation benötigt, um die gezeigten Überdeckungen zu erreichen. Das heißt, es entstehen $510000 * 250Byte \approx 122MB$ neuer Traffic ohne die Verwendung von AMI bzw. $396000 * 250Byte \approx 95MB$ mit der Verwendung. Da die Verwendung des AMI Parameters einige zusätzliche Nachrichten einsparen kann und keine deutliche Verschlechterung der Überdeckung bewirkt, kann mit diesem Ergebnis die Einsparung berechnet werden. Für die Einsparung der $95MB$ zusätzlichem Traffic, müssen mindestens 190 Nachrichten im Laufe der Simulation per Ad-Hoc Übertragung versendet werden. Nimmt man als Vector Extension Parameter den Wert 120, so liegt die niedrigste Überdeckung bei ca. 44%. Das heißt das etwa 330 Nachrichten pro Request erfolgreich per Ad-Hoc Übertragung

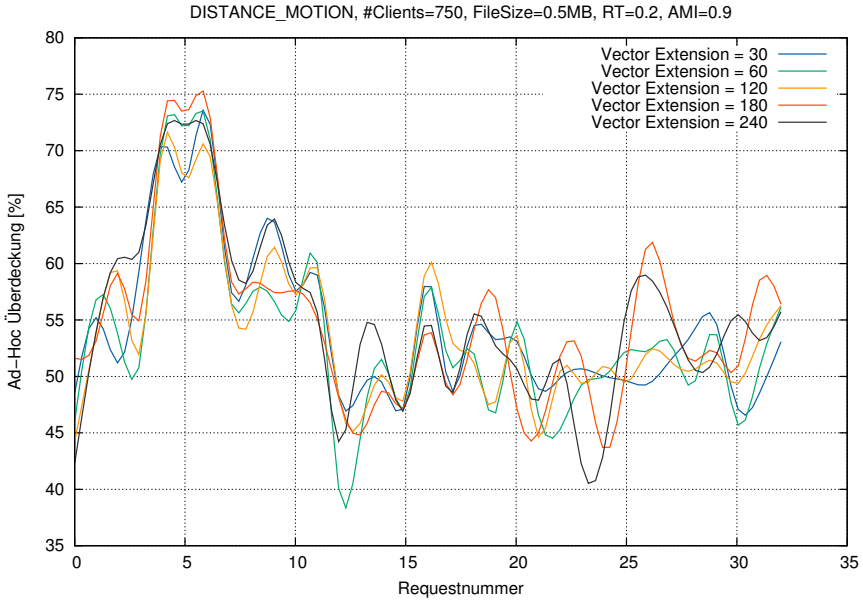

Abbildung 5.6: DISTANCE_MOTION Algorithmus mit verschiedenen Werten des Vector Extension Parameters.

versendet wurden und damit der Algorithmus tatsächlich Traffic einsparen kann. Nimmt man statt der bisherigen Dateigröße von $0.5MB$ die kleinere Wahl mit $0.25MB$, müssten mindestens 380 Datennachrichten eingespart werden. Aber auch dies ist über die Dauer der Simulation problemlos erreichbar.

5.4.5 DISTANCE_COMBINED

Die weiteren Auswertungen beziehen sich auf eine simulierte Dateigröße von $0.5MB$ und einer Clientanzahl von 750. Auch hier liegt ein Großteil (78%) der Überdeckungen zwischen 45% und 60%.

Adaptive Message Inquiry

Wie bei den vorherigen Algorithmen ist bei dem Adaptive Message Inquiry Parameter ein Unterschied zwischen den simulierten Werten nicht feststellbar. Lediglich in dem Fall, in dem dieser Parameter nicht genutzt wird ergeben sich andere Überdeckungswerte. Allerdings sind die erreichten Überdeckungskurven in beiden Fällen sehr ähnlich.
Das Ziel, die Anzahl der Positionsnachrichten zu reduzieren, konnte erreicht werden. Die Einsparungen sind allerdings, im Vergleich zur DISTANCE oder DISTANCE_ENHANCED Strategie, sehr gering. Die benötigten Positionsnachrichten liegen sogar etwas höher als bei dem DISTANCE_MOTION Ansatz. Mit der Verwendung des AMI Parameters (hier $AMI = 0.9$) waren 438000 Positionsnachrichten nötig, ohne dessen Verwendung 505500.

Reinjection Threshold

Änderungen am Reinjection Threshold Parameter verhalten sich wie bei den vorherigen Algorithmen. Deutliche Steigerungen der Überdeckungen sind vor allem von dem Wert $RT = 0.02$ bis $RT = 0.1$ zu beobachten. Von $RT = 0.1$ auf $RT = 0.2$ ist dann allerdings nur noch eine geringe Steigerung feststellbar.

Radius Reduction

Als bester Wert des Radius Reduction Parameters hat sich $RR = 8$ herausgestellt (siehe auch Abbidung 5.7). Die im Durchschnitt erreichten Überdeckungen unterscheiden sich jedoch nur marginal und variieren zwischen 54.7% und 55.3% Der höchste durchschnittliche Überdeckungswert wurde mit $RR = 8$ erzielt, weshalb für die Auswertungen dieser Strategie immer $RR = 8$ gesetzt wurde.

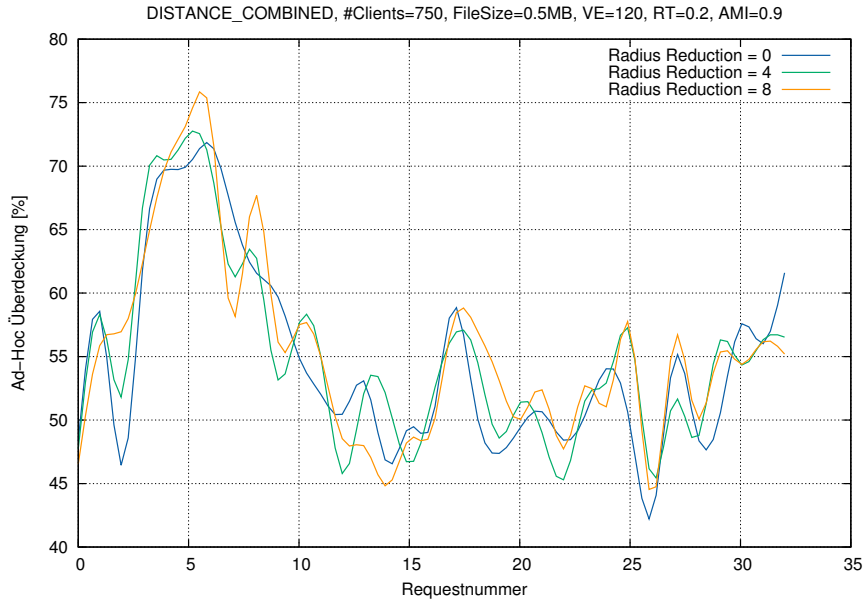

Abbildung 5.7: DISTANCE_COMBINED Algorithmus mit verschiedenen Werten des Radius Reduction Parameters.

Vector Extension

Auch bei dem Vector Extension Parameter schlagen sich Änderungen am Wert das Parameters nur vernachlässigbar auf die erreichten Überdeckungswerte nieder. Insgesamt schwanken die durchschnittlichen Überdeckungswerte zwischen 54.9% und 55.3%. Der höchste durchschnittliche Überdeckungswert wurde mit $VE = 60$ und $VE = 120$ erzielt, weshalb für die Auswertungen dieser Strategie immer $VE = 120$ gesetzt wurde.

Traffic Einsparung

Wie oben erwähnt, werden 438000 bzw. 505500 zusätzliche Nachrichten im Laufe der Simulation benötigt, um die gezeigten Überdeckungen zu erreichen. Das heißt, mit der Verwendung des AMI Parameters entstehen $438000 * 250Byte \approx 104.5MB$ neuer Traffic. Um diesen neu entstehenden Traffic zu kompensieren, müssen mindestens 209 Datennachrichten im Laufe der Simulation eingespart werden (d.h. Ad-Hoc versendet werden). Die durchschnittliche erreichte Überdeckung liegt bei 55.3%, d.h. von den 750 Datennachrichten die pro Request an die Clients verschickt werden müssen, werden etwa 414 per Ad-Hoc Übertragung versendet. Das heißt, der für Positionsnachrichten zusätzlich entstehende Traffic wird bereits durch die Ad-Hoc Übertragungen bei einem Request ausgeglichen.

5.4.6 SAME_STREET

Wie unter 5.2.1 Aufbau beschrieben, musste bei dem SAME_STREET Algorithmus auf eine niedrige Anzahl an Simulationen geachtet werden. Darum wurde um die Auswirkung der Parameter zu testen, nur mit einer Nachrichtengröße von 0.5 MB sowie mit einer Clientanzahl von 300 und 500 simuliert. Die vorherigen Algorithmen verhalten sich bezüglich der Auswirkungen der Parameter sehr ähnlich. Wie z.B. die Verbesserung der Überdeckung bis zu einem Reinjection Threshold von 0.2 oder einer Reduzierung der Positionsnachrichten ohne Einbußen der Überdeckung bis zu einem Adaptive Message Inquiry Wert von 0.9. Zur besseren Vergleichbarkeit wurde deshalb eine einzelne Simulation mit den selben Parameterbelegungen, wie auch in den vorherigen Algorithmen, durchgeführt (siehe dazu Abbildungen 5.10 und 5.11). Alle folgenden Auswertungen beziehen sich allerdings auf eine Nachrichtengröße von 0.5 MB bzw. 500 Clients.

Adaptive Message Inquiry

Die Simulationen haben gezeigt, dass es keinen Unterschied macht welcher der drei positiven simulierten Werte des AMI Parameters verwendet wird. Positive AMI Werte produzieren, abhängig von dem übrigen Parameter (Reinjection Threshold) immer die gleichen Überdeckungskurven. Eine andere Überdeckungskurve kommt nur dann zustande, wenn ohne den AMI Parameter (d.h. $AMI = -1$) simuliert wird. Dadurch ist die dynamische Versendung von Positionsnachrichten deaktiviert, d.h. es werden etwas mehr Positionsnachrichten versendet (siehe dazu Abbildung 5.8). Wesentlich bessere Ergebnisse lassen sich dadurch allerdings nicht erzielen.

Ohne die Verwendung des AMI Parameters (d.h. mit regelmäßigen Positionsaktualisierungen) werden 74000 Positionsnachrichten im Laufe der Simulation versendet, davon je eine Hälfte vom Server zu den Clients bzw. von den Clients zurück zum Server. Mit $AMI = 0.9$ kann diese Zahl auf 65000 reduziert werden. Trotzdem liegt diese Zahl höher als die Erwartungen, da die Anzahl der zusätzlichen Positionsnachrichten größer ist, als bei dem DISTANCE_MOTION Algorithmus mit denselben Parameterkombinationen.

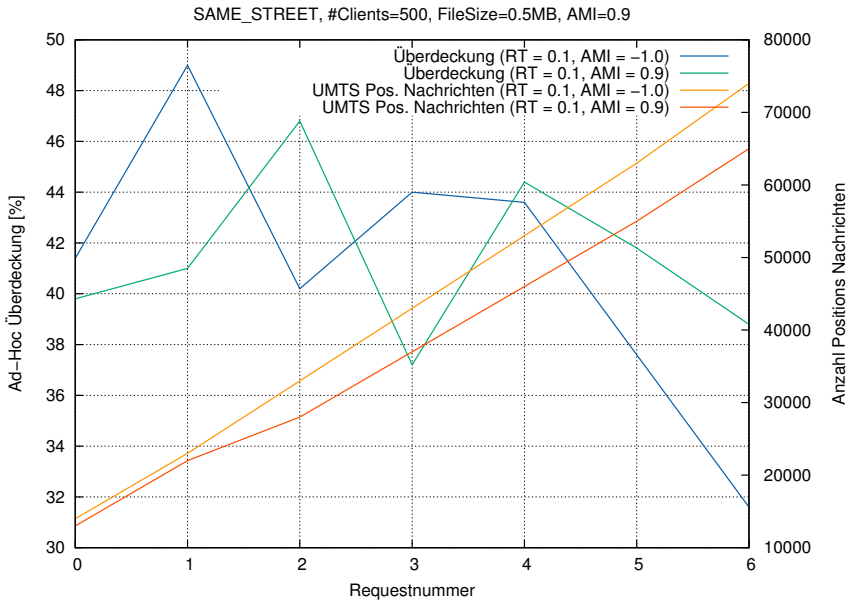

Abbildung 5.8: SAME_STREET Algorithmus mit verschiedenen Werten des Adaptive Message Inquiry Parameters.

Insgesamt macht die Verwendung des AMI Parameters keinen großen Unterschied auf die erreichten Überdeckungswerte. Die Überdeckung ohne den AMI-Parameter beträgt im Durchschnitt etwa 41,1% bzw. 41,4% mit.

Reinjection Threshold

Zur Reduzierung der Anzahl der Simulationen wurde auf die Simulieren mit $RT = 0.2$ verzichtet. Das heißt es werden nur die Werte 0.02, 0.05 und 0.1 verglichen. Wie bei den vorherigen Algorithmen zeigt sich die größte Steigerung der Überdeckung wieder von dem Wert 0.02 auf 0.05, wobei die maximale Überdeckung wiederum bei $RT = 0.1$ liegt (siehe auch Abbildung 5.9).

Traffic Einsparung

Die 65000 zusätzlich erforderlichen Positionsnachrichten im Laufe der Simulation entsprechen einem weiteren Traffic von 15.5 MB. Mit der simulierten Nachrichtengröße von 0.5 MB ist dieser zusätzliche Aufwand nach der Auslagerung von 31 Nachrichten kompensiert. Bei der Verwendung von $AMI = 0.9$ und $RT = 0.1$ wurden während der Simulation im Durchschnitt 207 Nachrichten pro Request ausgelagert. Das heißt, der zusätzliche Aufwand der durch die Positionsnachrichten zustande kommt, wurde bereits nach der Dauer eines Requests vollständig ausgeglichen.

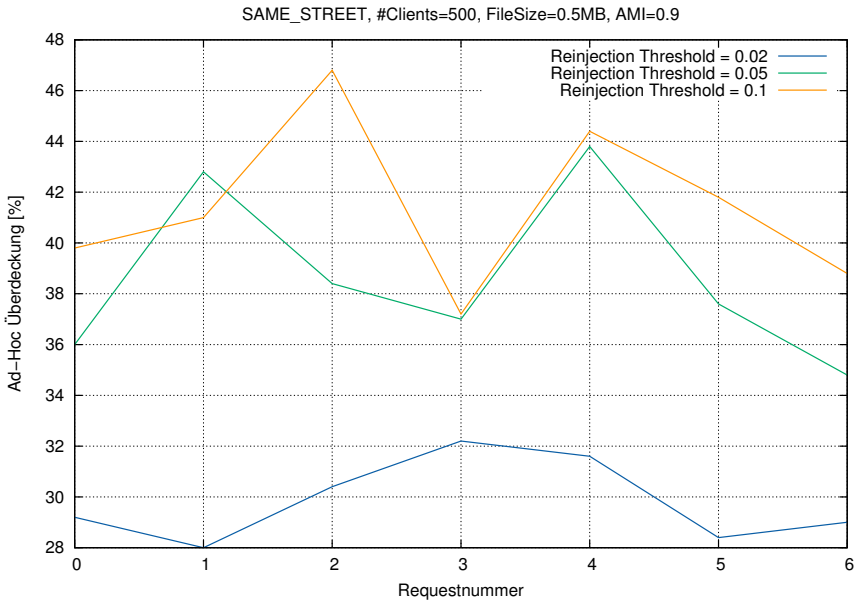

Abbildung 5.9: SAME_STREET Algorithmus mit verschiedenen Werten des Reinjection Threshold Parameters.

5.5 Fazit

5.5.1 Ergebnisse

Alle hier aufgeführten Ergebnisse beziehen sich auf Simulationen mit 750 Clients, einer Nachrichtengröße von 0.5 MB, einem Reinjection Threshold von 0.2 sowie einem Adaptive Message Inquiry Wert von 0.9 (ausgenommen RANDOM).

RANDOM Das Ergebnis der RANDOM Strategie entspricht relativ genau den Erwartungen. Die erreichten Überdeckungswerte sind geringer als bei den übrigen Strategien und sie schwanken vergleichsweise stark.

durchschnittliche Überdeckung: 36.4%

DISTANCE Die erwarteten guten Überdeckungswerte wurden erreicht, jedoch waren wesentlich weniger Positionsnachrichten erforderlich, als angenommen.

durchschnittliche Überdeckung: 54.8%

DISTANCE_ENHANCED Maximale durchschnittliche Überdeckung wurde mit einem Radius Reduction Wert von 4 erreicht. Das Ergebnis Entspricht ebenfalls den Erwartungen. Die Überdeckungswerte sind ähnlich wie die des DISTANCE Algorithmus und eine leichte Erhöhung der nötigen Positionsnachrichten kann ebenfalls festgestellt werden.

durchschnittliche Überdeckung: 56.5%

DISTANCE_MOTION Maximale durchschnittliche Überdeckung wurde mit einem Vector Extension Wert von 120 erreicht. Die Höhe der Ad-Hoc Überdeckung entspricht den Erwartungen. Die Annahme, dass dieser Algorithmus weniger Positionsnachrichten bewirkt als beispielsweise DISTANCE konnte nicht bestätigt werden. Es ist sogar die etwa vierzigfache Menge an Nachrichten nötig im Vergleich zum DISTANCE_ENHANCED Algorithmus.

durchschnittliche Überdeckung: 53.9%

DISTANCE_COMBINED Maximale durchschnittliche Überdeckung wurde mit einem Radius Reduction Wert von 8, sowie einem Vector Extension Wert von 120 erreicht. Entspricht den Erwartungen insofern, als dass die Überdeckung etwa so hoch wie die des DISTANCE_ENHANCED Algorithmus ist. Auch die Anzahl der nötigen Positionsnachrichten stimmt mit den Erwartungen grundsätzlich überein, lediglich die Annahme das DISTANCE_MOTION und DISTANCE_COMBINED weniger Positionsnachrichten bewirken als die übrigen Algorithmen ist falsch.

durchschnittliche Überdeckung: 55.3%

SAME_STREET Die erreichte Überdeckung entspricht den Erwartungen. Allerdings ist die Anzahl der nötigen Positionsnachrichten deutlich höher als angenommen.

durchschnittliche Überdeckung: 44.0%

Ein interessantes Resultat ist, dass die erreichten Überdeckungswerte sich offenbar nur minimal unterscheiden. Mit Ausnahme des SAME_STREET Algorithmus variieren die Ergebnisse nur zwischen 53.9% und 56.5%. Das heißt, unabhängig davon ob nur die Umgebung eines Clients betrachtet wird (Algorithmen DISTANCE und DISTANCE_ENHANCED) oder dessen zukünftige Bewegungsrichtung (DISTANCE_MOTION und DISTANCE_COMBINED) sind die Überdeckungen sehr ähnlich. Siehe dazu auch Abbildung 5.10.

Ganz im Gegensatz zu diesem Ergebnis stehen die Resultate bezüglich der nötigen Positionsnachrichten (Siehe Abbildung: 5.11). Dabei erzeugen die Algorithmen die lediglich die Umgebung eines Clients betrachten wesentlich weniger zusätzlichen Traffic (etwa ein Vierzigstel verglichen mit den Algorithmen die die Bewegungsrichtung nutzen). Auch hier stellt der SAME_STREET Algorithmus wieder eine Ausnahme dar, der ebenfalls relativ viele Positionsnachrichten benötigt.

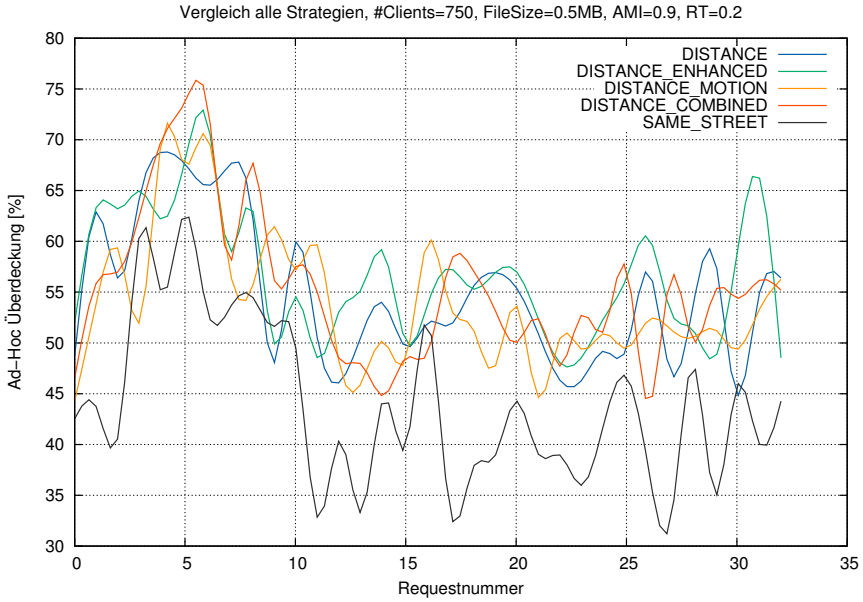

Abbildung 5.10: Vergleich der erreichten Überdeckungswerte aller Algorithmen (mit Ausnahme von RANDOM).

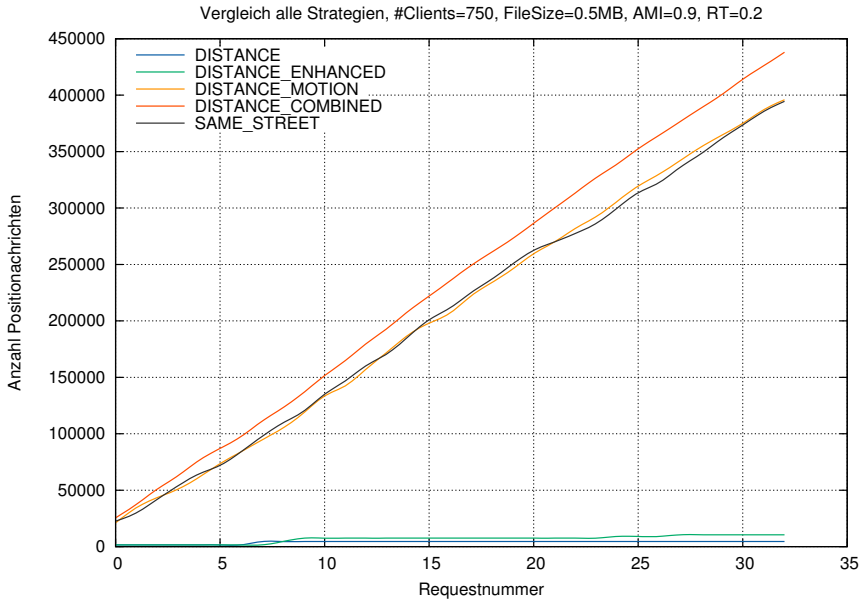

Abbildung 5.11: Vergleich der nötigen Positionsnachrichten aller Algorithmen.

KAPITEL 6

AUSBLICK

6.1 Strategien

6.1.1 DISTANCE_ ENHANCED

Wie bereits unter 4.3.3.2 Problem beschrieben, gibt es vor allem bei der DISTANCE_ENHANCED Strategie das Problem, dass weit außen liegende Clients zwar als überdeckt eingestuft werden, aber Nachrichten nicht bekommen, da die Übertragung über mehrere Clients zu lange dauert. Dies ist insbesondere dann der Fall, wenn ein Knoten im Überdeckungsbaum (siehe 4.5 DISTANCE_ENHANCED Baumdarstellung) viele Kinder hat, selbst aber die Nachricht erst zum spätest möglichen Zeitpunkt erhält, also nachdem alle anderen Knoten des Levels die Nachricht bereits empfangen haben. Je häufiger der genannte Fall eintritt, desto höher ist die Wahrscheinlichkeit, dass einige Clients (bzw. Knoten) in hohen Leveln die Nachrichten nicht empfangen. Eine Möglichkeit, die Auswirkungen dieses Problems zu reduzieren wurde bereits beschrieben. Dabei wird für Clients, um die Bewegungen zu kompensieren, ein kleinerer Übertragungsradius simuliert je weiter sie vom Zielclient entfernt sind.

Ein weiterer Ansatz ist die Verteilungsreihenfolge der Nachrichten aktiv zu steuern. Das heißt, dass eine vom Server festgelegte Reihenfolge der Clients bei dem Versenden der Nachrichten eingehalten werden muss. Anhand des Überdeckungsbaumes kann serverseitig für jeden Kindknoten der Wurzel bestimmt werden, welcher Ast die meisten Knoten und damit die am weitesten entfernten Clients hat. Die Kindknoten erhalten dann die Nachricht entsprechend in der absteigenden Reihenfolge der Astlänge.
Im nächsten Level muss wieder für jeden Kindknoten die Länge der Äste bestimmt werden. Dabei erhalten die Kindknoten wiederum die Nachricht in absteigender Reihenfolge der Astlänge.

Dieser Vorgang muss wiederholt werden, bis für jeden Knoten (außer der Blätter des Baumes) die Astlänge der Kindknoten bekannt ist und dadurch die Reihenfolge erstellt

werden kann. Die zu verschickende Nachricht muss dann um die global berechnete Reihenfolge erweitert werden, sodass jeder Empfänger überprüfen kann, an wen die Nachricht als nächstes verschickt werden muss.

6.1.2 Panic Zone

Die Panic Zone startet wie unter Punkt 4.1.2 Regulärer Ablauf (mit adaptiver Verteilung) erwähnt 14 Minuten vor Ablauf der Gültigkeit einer Nachricht. Ab diesem Punkt wird, um eine vollständige Auslieferung zu gewährleisten, eine Nachricht an alle Clients versandt, die diese noch nicht erhalten haben. 14 Minuten wurden gewählt, da dies auch bei schlechten Überdeckungen (d.h. es wurden nur relativ wenig Ad-Hoc Übertragungen ausgeführt) ausreicht, um ausstehenden Clients die Nachricht zu schicken.

Ein Verbesserung des Ansatzes ließe sich erreichen, indem die Länge der Panic Zone dynamisch gewählt wird. Zur Laufzeit muss dann folgendes bekannt sein und u.U. fortlaufend geprüft werden:

- Wie viele Clients die Nachricht noch nicht erhalten haben ($x_{clients}$)
- Wie hoch die aktuelle Bandbreite ist ($bandwith_{curr}$ in MBytes/s)
- Wie große der aktuelle Request ist ($size_{req}$ in MB)

Die Dauer (in Sekunden) der Panic Zone lässt sich dann wie folgt berechnen:

$$time = \frac{x_{clients} * size_{req}}{bandwith_{curr}}$$

Dadurch verkürzt sich unter Umständen die Panic Zone, was wiederum zu weiteren Ad-Hoc Übertragungen führt.

6.1.3 Einsatzmöglichkeit Verkehrsinformation

Für einen Einsatz in einem Verkehrsinformationssystem sind weitere Evaluierungen nötig. Unter Umständen sind die Algorithmen DISTANCE und DISTANCE_ENHANCED weniger effizient, wenn die Begegnungszeiten der Clients sehr kurz sind. Da für diese Algorithmen gute Ziele die sind, die viele Nachbarn haben (d.h. der Zielclient nach Empfang des Requests diesen relativ oft versenden muss), könnten die Ergebnisse bei höheren Bewegungsgeschwindigkeiten schlechter ausfallen. Dagegen könnte der DISTANCE_MOTION Ansatz im Vergleich dazu bessere Ergebnisse erzielen, da hier die Anzahl der Treffen mit anderen Clients innerhalb eines bestimmten Zeitraums ausschlaggebend ist. Interessant wären auch die Ergebnisse des SAME_STREET Ansatzes. Vermutlich schneidet dieser in einer Automobilumgebung etwas schlechter ab, da ein kritischer Punkt bei der Zielberechnung eine kreuzungsfreie Verbindung zwischen zwei Clients ist. Dies ist für die Bewegungen von Menschen auf einer Straße durchaus akzeptabel, da die Abstände zwischen zwei Kreuzungen

im Verhältnis zur Geschwindigkeit der Personen deutlich größer ist. Für eine Anwendung mit Fahrzeugen und den daraus resultierenden höheren Geschwindigkeit, ist dies allerdings eine schlechte Voraussetzung.

6.2 Privacy

Bisher unbeachtet sind Überlegungen zur Privatsphäre der Clients. Da es prinzipiell um das Auslagern von Daten geht, die viele Empfänger erreichen sollen, sind vertrauliche Daten in jedem Fall ausgeschlossen. Es ist vermutlich ein System nötig, bei dem bei der Übertragung eines Requests per WLAN der sendende Client nicht weiß, ob der Empfänger den Request tatsächlich benötigt, oder ob dieser nur eine weiterleitende Funktion hat.

6.3 Datenübertragung

Besonders ausschlaggebend bei der gemessenen Überdeckung ist die Übertragungsgeschwindigkeit sowie der Übertragungsradius der Smartphones. Um aussagekräftige Werte dafür zu erhalten, sind Tests unter realen Bedingungen daher unvermeidbar. Deutlich höhere Überdeckungen könnten beispielsweise erreicht werden, indem ein Kommunikationsinterface mit größerer Reichweite als Bluetooth oder WLAN verwendet wird.

LITERATURVERZEICHNIS

[3gp] Point-to-Point (PP) Short Message Service (SMS) support on mobile radio interface. URL http://www.3gpp.org/ftp/Specs/html-info/24011.htm. S. 34. (Zitiert auf Seite 29)

[802] IEEE Standard for Information Technology—Telecommunications and information exchange between systems—LANs and MANs—Specific requirements—Part 11: WLAN MAC and PHY Specifications—Amendment 5: Enhancements for Higher Throughput. URL http://standards.ieee.org/getieee802/download/802.11n-2009.pdf. S. 345ff. (Zitiert auf Seite 19)

[BDR12] P. Baier, F. Dürr, K. Rothermel. TOMP: Opportunistic Traffic Offloading Using Movement Predictions. In *Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*, S. 1–8. IEEE Computer Society, Clearwater, 2012. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2012-26&engl=0. (Zitiert auf den Seiten 14, 19, 31 und 48)

[BMV10] A. Balasubramanian, R. Mahajan, A. Venkataramani. Augmenting mobile 3G using WiFi. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, S. 209–222. ACM, New York, NY, USA, 2010. doi:10.1145/1814433.1814456. URL http://doi.acm.org/10.1145/1814433.1814456. (Zitiert auf Seite 16)

[HHK+12] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, J. Shao, A. Srinivasan. Mobile Data Offloading through Opportunistic Communications and Social Participation. *IEEE Transactions on Mobile Computing*, 11, Issue: 5:821 – 834, 2012. (Zitiert auf den Seiten 9 und 16)

[HSC09] S. Hasan, N. Siddique, S. Chakraborty. Femtocell versus WiFi - A survey and comparison of architecture and performance. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, S. 916 –920. 2009. doi:10.1109/WIRELESSVITAE.2009.5172572. (Zitiert auf Seite 19)

[IAK12] H. Izumikawa, S. Awiphan, J. Katto. Spatial uplink mobile data offloading leveraging store-carry-forward paradigm. In *Proceedings of the first ACM international workshop on Practical issues and applications in next generation wireless networks*, PINGEN '12, S. 33–38. ACM, New York, NY, USA, 2012. doi:10.1145/2348714.2348721. URL http://doi.acm.org/10.1145/2348714.2348721. (Zitiert auf Seite 17)

[KOK09] A. Keränen, J. Ott, T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, New York, NY, USA, 2009. (Zitiert auf den Seiten 39 und 45)

[RLBCE11] N. Ristanovic, J. Le Boudec, A. Chaintreau, V. Erramilli. Energy Efficient Offloading of 3G Networks. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, S. 202 –211. 2011. doi:10.1109/MASS.2011.27. (Zitiert auf Seite 17)

[SK12] V. A. Siris, D. Kalyvas. Enhancing mobile data offloading with mobility prediction and prefetching. In *Proceedings of the seventh ACM international workshop on Mobility in the evolving internet architecture*, MobiArch '12, S. 17–22. ACM, New York, NY, USA, 2012. doi:10.1145/2348676.2348682. URL http://doi.acm.org/10.1145/2348676.2348682. (Zitiert auf den Seiten 9 und 15)

[WAL+11] J. Whitbeck, M. Amorim, Y. Lopez, J. Leguay, V. Conan. Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, S. 1 –10. 2011. doi:10.1109/WoWMoM.2011.5986466. (Zitiert auf den Seiten 12, 14 und 25)

[wif] WiFi Direct. URL http://www.wi-fi.org/discover-and-learn/wi-fi-direct. (Zitiert auf Seite 19)

Alle URLs wurden zuletzt am 24. März 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

(Dominik Olp)