Technische Universität Ilmenau

Fakultät für Elektrotechnik und Informationstechnik

Studienarbeit

Simulation der Wegewahl in Ad-hoc Netzen

vorgelegt von	:	Christian Bornträger
geboren am	:	09.05.1978
E-Mail Adresse	:	christian@borntraeger.net
Telefon	:	0177 / 5152290
Studiengang	:	Ingenieurinformatik/M97
Studienrichtung	:	Multimediale Informations- und Kommunikationssysteme
Matrikel-Nr.	:	26092
Betreuender wiss. Mitarbeiter	:	Dipl.-Ing. Maik Debes
Abgabetermin	:	20.06.2002

Ilmenau, 17. Juni 2002

Inhaltsverzeichnis

1 Grundlagen **9**

1.1 Aufgabenstellung 9

1.2 Einführung 9

2 Netzwerksimulatoren **10**

2.1 Ad Hoc Network Simulator - AHNS [BP] 10

2.2 GloMoSim/Parsec [UCL] 10

2.3 Maryland Routing Simulator - MaRS [Unib] 11

2.4 Ptolemy [Unia] 12

2.5 adhocsim [PC] 12

2.6 The Network Simulator - NS-2 [Unic] 13

2.7 BlueHoc [IBM] 13

2.8 Folgerung 14

3 Bewegungsmodelle **16**

3.1 Theoretische Grundlagen 16

3.2 Bewegungsmodelle aus zellularen Mobilfunknetzen 17

3.2.1 Einleitung 17

3.2.2 Fluid-Flow-Modelle 17

3.2.3 Hong/Rappaport-Modell 18

3.2.4 Modell nach Guérin 18

3.2.5 Modellierung nach Zonoozi 19

3.2.6 Brownsche Bewegung 19

3.2.7 ETSI-Mobilitätsmodelle 19

3.3 Bewegungsmodelle für Ad-Hoc-Netze 20

3.3.1 Monarch-Projekt[Ric] 20

3.3.2 Modell des Ad Hoc Network Simulator 22

3.3.3 Vergleich der Bewegungsmodelle 23

4 NS-2 **24**

4.1 Einführung 24

4.2 Installation unter Linux/Unix 24

4.2.1 All-in-one Installation 24

4.2.2 Installation aus den Teilen 24

4.3 Installation unter Windows 25

4.4 Integration der Bewegungsmodelle 25

4.5 Anpassen der grafischen Ausgabe 26

5 Eigene Erweiterungen **27**

5.1 Einleitung 27

5.2 Implementierte Bewegungsmodelle 27

5.2.1 Grundlagen 27

5.2.2 Guérin-ähnliches Modell 27

5.2.3 Zonoozi-ähnliches Modell 28

5.2.4 Bewegung anhand von Gebietsdefinitionen 29

5.2.5 Szenario 1: konzentrische Bewegungen 30

5.2.6 Szenario 2: Rennsport 31

5.2.7 Szenario 3: ETSI-ähnliches Manhattan-Modell 31

5.3 manager.tcl 33

5.4 viewtraffic 33

5.5 gentraffic 34

5.6 Aufbau von gentraffic und viewtraffic 35

5.6.1 Klassendiagramm gentraffic 35

5.6.2 Klassendiagramm viewtraffic 35

5.6.3 Klasse Area 35

5.6.4 Klasse Scheduler 36

5.6.5 Klasse Event 37

5.6.6 Klasse Markov, Minded, User, TcpConn und Udpconn 37

5.6.7 Klasse Parse 37

5.6.8 Klasse Node 38
5.6.9 Klasse MyWidget 38
5.6.10 Klasse Conns 39
5.6.11 Klasse Draw 39
5.7 Installieren der Tools 39
5.8 Erzeugen von Verteilungen 40
5.8.1 Grundlagen 40
5.8.2 Gleichverteilung 40
5.8.3 negative Exponentialverteilung 40
5.8.4 Gauss-Verteilung 40

6 Simulation und Bewertung **41**
6.1 Vergleich der Routingverfahren 41
6.1.1 Kriterien 41
6.1.2 Szenario 1: Kreisbewegungen, eine TCP-Verbindung 42
6.1.3 Szenario 2: Rennsport 46
6.1.4 Szenario 3: Manhattan 49

7 Ausblick **51**

8 Resümee **51**

A Gebietsdatei Manhattan-Szenario **55**

B Matlab-Skripte **56**
B.1 prepare.m 56
B.2 writeps.m 56

C Neue Routing-Protokolle **57**

Abbildungsverzeichnis

1 Bewegung Flur/Zimmer, Zimmer/Flur oder den Flur entlang 20
2 Manhattan-artige Stadtstruktur-Outdoor 21
3 Richtungsänderung 21
4 Markov-Bewegung: zufällig anhand von Verteilungen 28
5 Bewegung abhängig von vorheriger Bewegung 28
6 Vier Gebiete die konzentrische Rechteckbewegungen definieren 30
7 Rennsport-Szenario 32
8 Manhattan-Szenario, ähnlich ETSI 32
9 Oberfläche des manager.tcl 33
10 Klassendiagramm gentraffic 35
11 Klassendiagramm viewtraffic 36
12 Datenraten des Szenarios „konzentrische Bewegung“ 43
13 AODV, DSR und TORA: Routing bei 107 und 108 Sekunden 44
14 Routing bei 268 Sekunden. TORA und DSDV 44
15 Route von AODV/TORA im Gegensatz zur Route von DSDV 45
17 Eine mögliche Route 46
16 Datenrate des Szenarios „Rennsport“ 47
18 Wechsel der Route nach 133 Sekunden 48
19 Hopanzahl beeinflusst Datenrate 48
21 Route von DSDV, AODV und TORA 49
20 Datenraten des Szenarios „Manhattan“ 50

Tabellenverzeichnis

1 Vergleich der Simulatoren 15
2 Vergleich einiger bestehender Bewegungsmodelle 23

Abkürzungsverzeichnis

802.11 Wireless LAN Medium Access Control and Physical Layer Specifications

ACK Acknowledgement

AHNS Ad Hoc Network Simulator

AODV Ad-hoc On-Demand Distance Vector Routing

API Application Program Interface

CBR Constant Bit Rate

DSDV Destination-Sequenced Distance-Vector Routing

DSR Dynamic Source Routing

ETSI European Telecommunications Standards Institute

FTP File Transfer Protocol

GloMoSim Global Mobile Information Systems Simulation Library

LOTOS Language of Temporal Ordering Specification

m/s Meter je Sekunde

MaRS Maryland Routing Simulator

NAM Network AniMator

ODMRP On Demand Multicast Routing Protocol

OTCL Object TCL

Parsec Parallel Simulation Environment for Complex Systems

QoS Quality of Service

SDL Specification and Description Language

TCL Tool Command Language

TCP Transmission Control Protocol

TORA Temporally -Ordered Routing Algorithm

UDP User Datagram Protocol

UMTS Universal Mobile Telecommunications System

wish Windowing Shell

WRP Wireless Routing Protocol

ÜBERSICHT

Mobile Ad-hoc Netze sind Gegenstand aktueller Forschungen. Die Wegewahl in diesen Netzen spielt dabei eine besondere Rolle. Diese Arbeit befasst sich mit der Simulation dieser Wegewahl. Dazu steht eine Vielzahl an Netzwerksimulatoren zur Verfügung. Die Wahl eines dieser Simulatoren war ebenfalls Bestandteil dieser Arbeit. Anhand von diversen Kriterien fiel die Wahl letztendlich auf den Network Simulator 2.

Zur reproduzierbaren Simulation und Leistungsbewertung ist ein flexibles Bewegungsmodell notwendig. Neben der Vorstellung existierender Bewegungsmodelle wird ein Tool vorgestellt, welches in den NS-2 ein Bewegungsmodell auf Basis von Gebietsdefinitionen integriert. Für dieses Modell werden einige Szenarien entwickelt und vorgestellt. Diese Szenarien werden jeweils mit den Routing-Protokollen AODV, DSR, DSDV und TORA simuliert und die Ergebnisse werden verglichen. Dabei werden Bewegungsmodelle und Routingverfahren bewertet und eingeschätzt.

Es wird außerdem ein weiteres Tool zur graphischen Darstellung der Verbindung entwickelt und vorgestellt, welches bei der Bewertung den Network Animator ergänzt, indem es die Routen anzeigt. Der Network Animator beschränkt sich hierbei auf Einzelpakete.

Es folgt ein Ausblick auf zukünftige Entwicklungen auf diesem Gebiet und mögliche Erweiterungen dieser Arbeit.

1 Grundlagen

1.1 Aufgabenstellung

Simulation der Wegewahl in Ad-hoc-Netzen: Zur Wegewahl in Ad-hoc-Netzen stehen zum gegenwärtigen Zeitpunkt eine Reihe von Routingprotokollen zur Verfügung. Funktionsweise und Eigenschaften dieser Protokolle können durch entsprechende Simulationen beurteilt werden. Dazu ist ein geeigneter Netzsimulator auszuwählen. Der Simulator soll weiterhin eine modulare Einbindung verschiedener Routingprotokolle gestatten. Es sind konkrete Szenarien für ein Routingprotokoll zu simulieren und die Ergebnisse zu bewerten. Dazu ist auf die Wahl eines geeigneten Bewegungsmodells wert zu legen. Die Bewegungen der Teilnehmer sollen über eine grafische Ausgabe veranschaulicht werden.

1.2 Einführung

Zusätzlich zu den bisherigen zellularen mobilen Netzen haben in den letzten Jahren die Ad-hoc-Netze an Bedeutung gewonnen. Allerdings ist der Begriff Ad-hoc nicht genau definiert. Beispielsweise wird in Bluetooth die Fähigkeit der automatischen Festlegung eines Masters in einem Piconetz als Ad-hoc-Fähigkeit bezeichnet. In WaveLan 802.11b dagegen ist es die Fähigkeit der Kommunikation zwischen zwei Teilnehmern ohne Access Points.

Diese oben genannten Netze implementieren also nur einfache Formen von Ad-hoc-Mechanismen, bieten aber keine weiterführenden Ad-hoc-Routingverfahren.

Hierfür sind eine größere Anzahl von Routing-Verfahren mit spezifischen Vor- und Nachteilen bekannt. Von Interesse sind hier die Zusammenstellungen der Manet [man] sowie das Hauptseminar „Ad-hoc-Netzwerke [Bor01]“.

Die spezifischen Eigenschaften, Funktionsweisen sowie Leistungsmerkmale können oftmals nur durch anwendungsbezogene Simulationen ermittelt werden. Für diese Simulationen ist es notwendig, Anwendungsszenarien und Bewegungsmodelle zu definieren. Da die breite kommerzielle Nutzung von Ad-hoc-Netzen noch bevorsteht, werden in

Zukunft weitere Bewegungsszenarien aus der Praxis hinzukommen. Deshalb sollte ein Bewegungsmodell erweiterbar und parameterisierbar sein. Als Parameter können unter anderem Geschwindigkeit und Anzahl der Nodes dienen.
Diese Arbeit beschreibt die Auswahl eines geeigneten Simulators und die Implementierung von Bewegungsmodellen und deren grafischer Veranschaulichung mit dem Network Simulator 2 [Unic].

2 Netzwerksimulatoren

2.1 Ad Hoc Network Simulator - AHNS [BP]

Dieser in Java geschriebene Simulator erlaubt die Entwicklung von Bewegungsszenarien für Ad-hoc-Netze und die Darstellung von Routen mittels graphischem Editor. Der Simulator läuft dabei als Applet in einem Webbrowser und nicht als eigenständiges Programm.
Das Bewegungsmodell des AHNS basiert auf Regionen, Gruppen und Nodes und wird in Abschnitt 3.3.2 näher erläutert.
Durch Auswahl zweier Nodes lässt sich jeweils die kürzeste Verbindung bzw. die Verbindung mit den wenigsten Hops darstellen.
Allerdings ist AHNS nicht geeignet, weitergehende Simulationen durchzuführen. Grund dafür ist das Fehlen von Routingprotokollen und die Nichtberücksichtigung der Datenrate. Betrachtungen zu Quality of Service sind damit nicht zu realisieren.
Da der Quellcode nicht öffentlich verfügbar ist, ist es außerdem nicht möglich eigene Erweiterungen hinzuzufügen.

2.2 GloMoSim/Parsec [UCL]

GloMoSim basiert auf Parsec, einer Programmiersprache für parallele Systeme. Dabei können einzelne Nodes als einzelne Parsec-Instanzen simuliert werden, welche durch die Optimierung für parallele Systeme dabei effizient berechnet werden. Ausgehend

von dieser Basis erhöhen weitere Optimierungen die Ausführungsgeschwindigkeit. Es können zur Zeit nur drahtlose Netze, unter anderem mit den Protokollen DSDV, WRP, DSR, ODMRP und AODV simuliert werden. Als Anwendungsprotokolle stehen unter anderem FTP und Telnet zur Verfügung. Anhand der aufgezählten Features scheint dieser Simulator sehr gute Voraussetzungen zur Simulation von Ad-hoc-Netzen zu besitzen.
Eine freie akademische Variante des Simulators ist zwar erhältlich, der Download ist jedoch auf edu-Domains begrenzt. Auf eine Anfrage an die Universität von Kalifornien Los Angeles wurde die TU-Ilmenau in den erlaubten Adressbereich aufgenommen. Die Aufnahme erfolgte jedoch erst nachdem der Vergleich der Simulatoren abgeschlossen war. Die praktische Nutzung konnte somit an dieser Stelle nicht als Auswahlkriterium berücksichtigt werden.
Eine kommerzielle Version ist als Qualnet ebenfalls erhältlich [Sca]. Diese erlaubt auch die Simulation von Infrastrukturnetzen.

2.3 Maryland Routing Simulator - MaRS [Unib]

Der Maryland Routing Simulator wird verwendet, um Routing-Systeme zu testen. MaRS bietet Möglichkeiten sowohl Link-State- als auch Distance-Vector-Protokolle zu simulieren. Allerdings wird MaRS inzwischen nicht mehr weiter entwickelt. Die letzte Version entstand 1992. Demzufolge fehlen Elemente zur Simulation von Ad-Hoc-Routing und die Unterstützung von mobilen Nodes.
Weiterhin müssen zum Kompilieren des Simulators einige Änderungen an den Quelltexten durchgeführt werden. MaRS basiert teilweise noch auf alten APIs, welche im Laufe der Zeit jedoch geändert wurden.
Da MaRS nicht mehr weiter entwickelt wird, sollte man ihn nicht mehr für zukünftige Simulationen verwenden.

2.4 Ptolemy [Unia]

Ptolemy ist ein Simulator, der insbesondere für die bottom-up-Konstruktion von technischen Systemen entwickelt wurde. Hierbei werden aus Teilkomponenten (stars) größere Systeme (galaxies) entwickelt bzw. zu einem Gesamtsystem (universe) kombiniert.
Haupteinsatzgebiet ist die Echtzeitsimulation von eingebetteten Systemen. Allerdings kann Ptolemy auch in der Nachrichtentechnik eingesetzt werden. Die letzt Version von Ptolemy Classic stammt von 1998. Neue Features werden nicht mehr integriert. Die Arbeit konzentriert sich zur Zeit auf Ptolemy 2, welches komplett in Java geschrieben wurde.
Für die Simulation von Ad-Hoc-Routing-Verfahren scheint Ptolemy weniger geeignet zu sein, da die Verbindung zwischen den Teilkomponenten (stars) bereits zur Kompilezeit festgelegt wird.[1] Ein dynamisches Routing zwischen den Komponenten durch den Simulator ist nur schwer möglich, es muss selbst implementiert werden.
Somit wäre es notwendig Ad-hoc-Routing als eigene Komponente komplett ohne Unterstützung des Simulators zu entwickeln.
Ptolemy hat allerdings den Vorteil auch komplexe Systeme durch Abstraktion simulierbar werden. So muss man bestimmte Teilelemente nicht simulieren, sondern kann sie durch Wahrscheinlichkeitsfunktionen abbilden, sofern diese bekannt sind.

2.5 adhocsim [PC]

Dieser Simulator entstand im Zuge einer Bachelor-Arbeit. Hierbei werden keine klassischen Routingverfahren simuliert, sondern Verfahren bei denen sich ein Teil der Nodes in vorbestimmter Weise bewegt. Diese Verfahren dienen speziell als Basis für die Kommunikation in Gebieten ohne Infrastruktur. Das Routing richtet sich hierbei also nicht nach den Teilnehmern, sondern die Teilnehmer nach den Routinganforderungen.
Diese Protokolle heißen *semi-compulsory* Protokolle. Der Simulator verzichtet zugunsten der Performance auf die exakte Nachbildung der Bitübertragungs- und der Si-

[1] Simulationen werden vor Aufruf kompiliert und als Programm ausgeführt

cherungsschicht und setzt stattdessen auf eine Näherung. Es werden keine „normalen“ Ad-hoc-Routingverfahren simuliert. Der Simulator ist also nicht für die Aufgabenstellung geeignet.

2.6 The Network Simulator - NS-2 [Unic]

Der NS-2 ist ein Simulator, welcher in 2 Programmiersprachen geschrieben wurde. Der Simulationskern basiert auf einer C++-Klassenhierarchie. Das Erzeugen und Aufrufen der entsprechenden Instanzen geschieht dagegen über die Skriptsprache OTCL. Dafür wurde eine Bindung zwischen C++ und OTCL geschaffen.
Dies verbindet die Vorteile der performanten Sprache C++ mit den Vorteilen der flexiblen Skriptsprache OTCL. OTCL ist dabei eine objektoriente Variante von TCL.
Im NS-2 werden zeitkritische Aspekte in C++ umgesetzt, während dynamische Vorgänge, wie Zeitpunkt des Verbindungsaufbaus, in OTCL implementiert werden.
NS-2 wurde entwickelt, um Netze zu simulieren. Deshalb sind Protokolle wie TCP und UDP bereits integriert. Weiterhin stehen Komponenten zur Simulation von drahtlosen Funknetzen wie der MAC-Layer 802.11 bzw. Ad-hoc-Routing-Algorithmen wie DSDV, DSR, AODV und TORA zur Verfügung.
Mit dem Network Animator ist zudem ein grafisches Tool zur Darstellung der Nodes, deren Bewegung und der einzelnen Pakete vorhanden.
Der NS-2 bietet eine breite Unterstützung für diverse Formen der Funkwellenausbreitung[2] und von MAC-Layern[3]. Die Simulation von fest verdrahteten Netzen, der Satellitenkommunikation sowie von hybriden Netzen (*wired-cum-wireless*) ist ebenfalls möglich.

2.7 BlueHoc [IBM]

BlueHoc ist ein Aufsatz auf den NS-2, um Bluetooth-Netze zu simulieren. Diese Erweiterung ist zur Zeit noch nicht integriert und muss manuell auf den NS-2 aufgesetzt

[2]Freiraum (*freespace*), Zweiwegeausbreitung (*TwoRayGround*) und Fading (*Shadowing*)
[3]Ethernet 802.3, WLAN 802.11 und Preamble Based TDMA

werden. Elemente der Erweiterung sind insbesondere:

- Device Discovery
- Verbindungsaufbau und QoS-Aushandlung
- Medium Access Control
- Funkcharakteristiken von Bluetooth
- statistisches Modellieren von Funkwegeübertragung in geschlossenen Räumen

BlueHoc bietet sich für Ad-hoc-Simulationen in Bluetooth-Netzen an. Allerdings beschränkt sich die Ad-hoc-Fähigkeit von Bluetooth auf die automatische Wahl eines Masters in einem Piconetz. Routing zwischen Piconetzen erfordert Mechanismen, welche über den Bluetooth-Standard hinaus gehen.

2.8 Folgerung

Jeder der Simulatoren ist für bestimmte Ziele entwickelt worden und passt deshalb mehr oder weniger gut zur Aufgabenstellung (Abschnitt 1.1). In Tabelle 1 sind relevante Eigenschaften der Simulatoren gegenübergestellt. Das Zeichen x steht dabei für vorhanden, - für nicht vorhanden. Ein „?“steht für unbekannt.
Eine wichtige Anforderung an einen Simulator ist die gute Unterstützung von mobilen Netzen. Deshalb ist MaRS zur Simulation von Ad-hoc-Netzen nicht geeignet.
Weiterhin müssen für die Aufgabenstellung ebenfalls Routingverfahren eingebunden oder erweiterbar sein. Hierbei sind insbesondere NS-2 und GloMoSim von Bedeutung, da diese die umfangreichste Unterstützung von Routingverfahren bieten. Praxisrelevante Simulationen erfordern die Implementation wichtiger Teile der TCP/IP-Protokoll-Suite sowie von Anwendungsprotokollen. Hier bieten GloMoSim und NS-2 ebenfalls die beste Unterstützung.
Schließlich sollten auch Bewegungsmodelle bzw. Szenarien integrierbar sein. Gerade beim AHNS ist bereits ein ausgefeiltes Bewegungsmodell integriert, welches verschiedene Szenarien unterstützt. Beim NS-2 kann die Bewegung der einzelnen Nodes durch

Simulator	AHNS	NS2	GloMoSim	MaRS	Ptolemy	adhocsim
PHY	-	x	x	-	x	-
MAC	-	x	x	-	x	-
IP	-	x	x	x	-	-
TCP	-	x	x	x	-	-
UDP	-	x	x	?	-	-
FTP	-	x	x	x	-	-
HTTP	-	-	x	?	-	-
Telnet	-	x	x	x	-	-
Real	-	x	-	-	-	-
CBR	-	x	x	?	?	?
mobil	x	x	x	-	x	x
fest	-	x	-	x	?	-
hybrid	-	x	-	-	?	-
Bewegungsmodell	x	extern	?	-	-	x
DSDV	-	x	-	-	-	-
AODV	-	x	x	-	-	-
DSR	-	x	x	-	-	-
TORA	-	x	-	-	-	-
WRP	-	-	x	-	-	-
ODMRP	-	-	x	-	-	-
weitere	-	x	x	x	-	-

Tabelle 1: Vergleich der Simulatoren

TCL-Skripte definiert werden. Dies kann durch externe Programme geschehen. Damit ist der NS-2 je nach Aufwand sehr flexibel zu programmieren.
Grundlage dieser Arbeit ist dabei der NS-2, da dieser alle genannten Forderungen erfüllt. Außerdem bietet er mit einem Energiemodell und der Simulation von Schicht 1 und 2 weiterführende Möglichkeiten zur Bewertung von Routingverfahren. Die freie Verfügbarkeit von Programm und Quelltext, sowie eine aktive Nutzer- und Entwicklergruppe bieten ebenfalls gute Aussichten auf weitere Entwicklungen.

3 Bewegungsmodelle

3.1 Theoretische Grundlagen

Verkehrstheoretische Modelle verwenden oft Markovsche oder Semi-Markovsche Zustandsmodelle [Sch01, Kapitel 5]. Es handelt sich dabei um gedächtnislose Modelle bzw. Modelle, welche zu einem bestimmten Zeitpunkt ihr Gedächtnis verlieren. Markovsche Modelle haben den Vorteil einer besseren statistischen Berechenbarkeit, da zu jeder Zeit gleiche Voraussetzungen herrschen und vorherige Begebenheiten nicht berücksichtigt werden müssen.
Gedächtnisbehaftete Modelle und deren statistische Eigenschaften sind oftmals nur noch schwer algorithmisch lösbar, da sehr langfristige Abhängigkeiten in den Gleichungen berücksichtigt werden müssen. Man kann sie oft nur durch eine Simulation bewerten.
Mischformen sind ebenfalls oft anzutreffen. Hierbei werden bestimmte Parameter anhand von Verteilungen bestimmt. Bei einer Exponentialverteilung, ist dieser Parameter ebenfalls gedächtnislos. Man nutzt hierbei aus, dass komplizierte und aufwendige Elemente einer Simulation durch stochastische Prozesse modelliert werden können. Dies wird insbesondere bei der Fehlermodellierung von Übertragungsmedien gemacht.

3.2 Bewegungsmodelle aus zellularen Mobilfunknetzen

3.2.1 Einleitung

Die Bewegungsmodelle der folgenden Kapitel stammen ursprünglich aus Arbeiten zur Simulation von zellularen Mobilfunknetzen. Einige von ihnen sind allerdings sehr flexibel und eignen sich auch für Ad-hoc-Netze.
Viele der genannten Bewegungsmodelle werden in der Doktorarbeit von Enrico Jugl beschrieben[Jug01].

3.2.2 Fluid-Flow-Modelle

Diese Modelle dienen zur Modellierung makroskopischer Eigenschaften. Es handelt sich um eine rein statistische Betrachtung, mit der es möglich ist, große Gebiete mit hoher Bevölkerungsdichte zu beschreiben.
Um diese großen Datenmengen zu verarbeiten, werden keine individuellen Bewegungen und Rufe, sondern der „Fluss“ als Ganzes simuliert.
Grundidee ist die Definition einer Teilnehmerdichte und die Definition einer mittleren Anzahl von Zellüberschreitungen.
Somit kann man Überschreitungsraten von Zellgrenzen modellieren. Dies ist insbesondere bei der Modellierung von Handoverprozessen interessant.
Für die Simulation der Wegewahl sind solche Modelle ungeeignet, da die Teilnehmer nicht einzeln simuliert werden, sondern nur die statistische Verteilung in den Zellen bestimmt wird. Dies verhindert eine sinnvolle Bewertung von Routingverfahren.
Weiterhin entspricht die Ortsauflösung der Zellgröße, einem Wert der bei Ad-hoc-Netzen nicht existiert.
Es besteht evtl. die Möglichkeit, Ergebnisse solcher Simulationen als Parameter eines eigenen Bewegungsmodelles zu verwenden. Das Modell an sich ist aber in keiner Form für die Simulation von Ad-hoc-Netzen geeignet.

3.2.3 Hong/Rappaport-Modell

Hierbei handelt es sich ebenfalls um ein statistisches Modell. Zu Beginn sind die Teilnehmer sowie die Richtungen gleichmäßig verteilt. Der Teilnehmer bewegt sich immer entlang dieser Richtung bis er die Zellgrenze erreicht. Nur dort kann sich die Richtung ändern. Die Wahrscheinlichkeitsdichtefunktion der Zellaufenthaltszeit ist in diesem Modell analytisch bestimmbar. Zur Berechnung werden dabei die Zellen durch hexagonale Wabenstrukturen approximiert.
Dieses Modell ist für zellulare Systeme entwickelt worden. Die Zellgrenzen sind elementarer Bestandteil der Bewegung. Es stellt also ebenfalls eine schlechte Basis für die Simulation nichtzellularer Netze dar.
Der Ersatz der Zellgrenze durch andere Auslöser eines Richtungswechsels würde das Wesen des Modells abwandeln und die analytische Berechenbarkeit der Wahrscheinlichkeitsdichtefunktion müsste neu überprüft werden.

3.2.4 Modell nach Guérin

Auch hier sind die Teilnehmerpositionen und Richtungen zu Beginn der Simulation zufällig und gleichverteilt.
Im Gegensatz zum vorherigen Modell erlaubt Guérins Modell jedoch Richtungswechsel innerhalb von Zellen.
Jeweils nach negativ exponentiell verteilten Zeitabständen erfolgt ein Richtungswechsel. Es handelt sich also um einen gedächtnislosen Prozess. Die Geschwindigkeit bleibt während der gesamten Verbindung konstant.
Zur Vereinfachung verwendet Guérin das Spiegelungsprinzip. Hierbei werden alle Zellen in einer Zelle abgebildet. Verlässt der Teilnehmer die Zelle, wird eine Winkeltransformation durchgeführt und der Teilnehmer tritt wieder in diese Zelle ein.
Dieses Modell kann ohne das Spiegelungsprinzip auch für Ad-Hoc-Netze verwendet werden, da das Bewegungsmodell dann unabhängig von den Zellen ist. Die Bewegung wird zudem auf individuelle Nodes angewandt, da es sich um kein statistisches Modell handelt. Die Simulation des Routings ist somit möglich.

3.2.5 Modellierung nach Zonoozi

Zu Beginn der Simulation sind die Teilnehmer wieder unabhängig und gleichverteilt. Die Startrichtung ist dabei beliebig. Nach einem negativ exponentialverteiltem Zeitabstand kann sich die Richtung in einem Bereich von $\pm\phi$ ändern, hängt also von den vorherigen Bewegungen ab.
Die Anfangsgeschwindigkeit richtet sich nach einer abgeschnittenen Gauss-Verteilung. Bei jeder Richtungsänderung kann sich die Geschwindigkeit ebenfalls im Bereich von $\pm 10\%$ ändern.
Dieses Modell berücksichtigt vorherige Bewegungen, hat also keine Markov-Eigenschaften. Da die Bewegung der Teilnehmer individuell und unabhängig von Zellgrenzen erfolgt, ist auch dieses Modell für Ad-hoc-Netze geeignet.

3.2.6 Brownsche Bewegung

Diese Modelle beschreiben zufällige Bewegungen. Hierbei wird die Wahrscheinlichkeitsdichtefunktion um einen, sich evtl. auch bewegenden Mittelpunkt mit fortschreitender Zeit immer breiter. Zum Zeitpunkt t_0 ist die Wahrscheinlichkeit, dass sich der Teilnehmer an der Position (x_0, y_0) aufhält 1. Diese Art der Bewegung stammt von der Bewegung kleiner Teilchen im Wasser und wurde durch den englischen Botaniker Brown entdeckt. Später wurde dieses Modell in der Teilchenphysik benutzt.
Es wird ebenfalls verwendet um zufällige Bewegungen vieler Teilnehmer zu simulieren. Die Brownsche Bewegung kann problemlos für Ad-hoc-Netze verwendet werden, ist aber nicht immer praxisrelevant.

3.2.7 ETSI-Mobilitätsmodelle

ETSI definiert für UMTS drei Szenarien von denen zwei auch für Ad-Hoc-Netze interessant sind[Eur98, S. 49ff.]:

- Indoor Office Test Environment Deployment Model: Es wird von regelmäßigen Büroräumen ausgegangen (Abbildung 1). Die Teilnehmer kennen dabei die

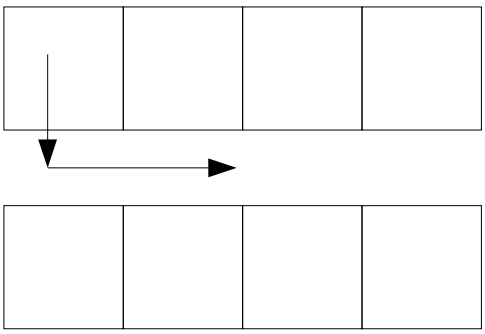

Abbildung 1: Bewegung Flur/Zimmer, Zimmer/Flur oder den Flur entlang

Zustände stationär und mobil. Befindet sich der Teilnehmer in einem Büro und wechselt in den Zustand mobil, so bewegt er sich auf den Gang, wo eine Zielposition festgelegt wird. Der Teilnehmer bewegt sich zur Zielposition und wechselt danach in das entsprechende Büro. Im Büro ändert sich der Zustand zu stationär.

Bei diesem Modell ist die Rate der Teilnehmer in den Büros, die mittlere Aufenthaltszeit im Büro, der Zeitschritt Δt, die Anzahl der Büros und die Geschwindigkeit festgelegt. Aus diesen Werten können die Zustandsübergangswahrscheinlichkeiten bestimmt werden.

- Manhattan-like urban model and deployment scheme: Es wird von einem regelmäßigen Stadtszenario ausgegangen (Abbildung 2).

  Die Bewegung findet dabei nur auf den 30m breiten Straßen zwischen den 200m langen Häusern statt. Die Positionsauflösung beträgt 5 Meter. An jeder Kreuzung ändert der Teilnehmer mit einer gewissen Wahrscheinlichkeit p_w die Richtung (Abbildung 3).

3.3 Bewegungsmodelle für Ad-Hoc-Netze

3.3.1 Monarch-Projekt[Ric]

Das Monarch-Projekt der Rice-Universität befasst sich mit Netzen von mobilen und drahtlosen Hosts. Es basiert auf Arbeiten der Carnegie Mellon's School of Computer

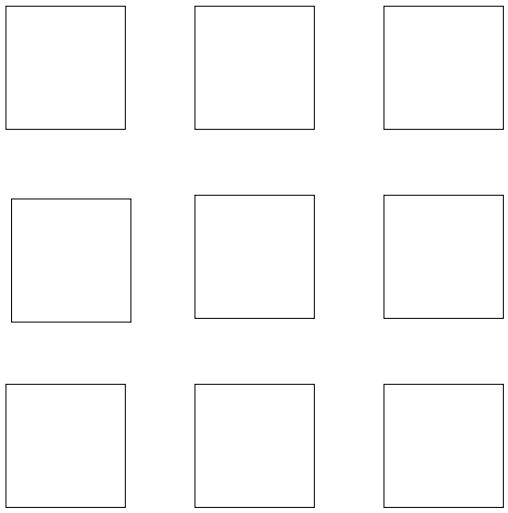

Abbildung 2: Manhattan-artige Stadtstruktur-Outdoor

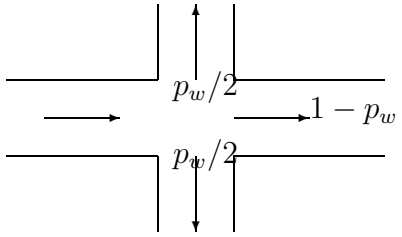

Abbildung 3: Richtungsänderung

Science (http://www.cs.cmu.edu), welche maßgeblich an den Ad-hoc-Fähigkeiten des NS-2 beteiligt war.
Mit Hilfe dieser Erweiterungen wurden diverse Ad-hoc-Routingverfahren verglichen und bewertet [BMJ+98]. Zu diesem Zwecke wurden einfache Mobilitäts-Generatoren entwickelt, welche in aktuellen NS-2 Paketen enthalten sind.
Diese Tools sind:

- setdest: Dies ist ein C++-Programm zum Erzeugen von Bewegungsmustern von Nodes. Parameter sind die Anzahl der Nodes, Pausenzeit, maximale Geschwindigkeit und die Größe des Simulationsgebietes. Es werden dabei Markov-Bewegungsszenarien erzeugt.

- cbrgen.tcl: Dies ist ein TCL-Skript zum Erzeugen von Verbindungen zwischen Nodes. Parameter sind die Anzahl der Nodes, die Wahl zwischen TCP-Verbindungen oder UDP-Verbindungen mit konstanter Bitrate sowie die Anzahl der Verbindungen und die Übertragungsrate.

Die Szenarien, welche mit diesen Tools erzeugt werden, dienen vor allem dem Vergleich von Routingprotokollen ohne spezielle Praxis-Szenarien. Für eine grundlegende Bewertung sind sie auf jeden Fall geeignet. Eine Auswahl und Bewertung von Routingverfahren für bestimmte Praxiseinsätze ist jedoch nicht möglich.

3.3.2 Modell des Ad Hoc Network Simulator

Der Ad Hoc Network Simulator bietet ein komplexeres Bewegungsmodell als das CMU-Monarch-Projekt. Die Basis bilden hier Regionen, welche durch Polygone definiert sind. Für jede Region sind Geschwindigkeit und Richtung der Teilnehmer definierbar.
Ein weiteres Merkmal dieses Bewegungsmodelles sind Gruppen, welche aus mehreren Nodes bestehen. Das Gruppenzentrum ist zentraler Teil der Gruppe bezüglich der Bewegung. Es bewegt sich anhand der Merkmale der definierten Region und ist maßgebend für die Mitglieder dieser Gruppe indem es Mittelwerte für Geschwindigkeit und Richtung vorgibt.

Für die Nodes dieser Gruppe kann festgelegt werden, inwieweit Geschwindigkeit und Richtung vom Gruppenzentrum abweichen können. Ist die Gruppengeschwindigkeit beispielsweise 20 km/h und die Abweichung 5 km/h, so bewegen sich alle Nodes der Gruppe mit einer Geschwindigkeit zwischen 15 und 25 km/h.

3.3.3 Vergleich der Bewegungsmodelle

Tabelle 2 stellt grundlegende Eigenschaften einiger Bewegungsmodelle gegenüber.

Modell	Einzel-nodes	ohne Zellen möglich	Praxis-szenarien	gedächtnislos
Fluid-Flow	-	-	-	-
Hong-Rappaport	-	-	-	-
Guérin	x	x	-	x
Zonoozi	x	x	-	-
Brownsche Bewegung	x	x	-	x
ETSI-indoor	x	x	x	-
ETSI-outdoor	x	x	x	-
Monarch	x	x	-	x
AHNS	x	x	definierbar	?

Tabelle 2: Vergleich einiger bestehender Bewegungsmodelle

Zur Simulation der Wegewahl ist es unerlässlich, jeden Node einzeln zu erfassen - statistische Mittelungen sind nicht geeignet. Weiterhin sollte in einem Modell die Bewegung unabhängig von Zellen sein. Aufgrund dieser Forderungen fallen die Fluid-Flow-Modelle sowie das Hong-Rappaport-Modell für die Simulation von Ad-hoc-Netzen weg.
Alle weiteren Modelle sind zur Simulation von Ad-hoc-Netzen verwendbar. In Zukunft sollten evtl. praxisrelevantere Modelle verwendet werden, da zum Vergleich von Ad-hoc-Netzen schon einige Zufallsmodelle benutzt wurden [BMJ+98].
Insbesondere die ETSI-Modelle für UMTS bieten ebenfalls gute Szenarien für Ad-hoc-Netze.

4 NS-2

4.1 Einführung

4.2 Installation unter Linux/Unix

Es gibt 2 Arten der Installation des NS-2, welche durch jeweilige Vor- und Nachteile charakterisiert ist.

4.2.1 All-in-one Installation

Bei der All-in-one-Installation werden sämtliche benötigten Elemente in einer gepackten Datei mitgeliefert. Diese ist in ein Unterverzeichnis zu entpacken indem das install-Skript auszuführen ist:

```
tar xvzf ns-allinone-2.1b9.tar.gz
cd ns-allinone-2.1b9
./install
```

4.2.2 Installation aus den Teilen

Diese Installationsvariante ist insbesondere dann empfehlenswert, wenn man aktuelle Entwicklungen nutzen möchte. NS-2 ist kein fertiges Produkt, sondern wird immer noch aktiv entwickelt. Bestimmte Features sind evtl. nicht in der letzten All-in-one-Variante enthalten, da die Abstände zwischen den Releases relativ groß sind.
Für diese Art der Installation sollten passende Versionen der folgenden Pakete in ein Verzeichnis entpackt und in folgender Reihenfolge kompiliert werden

- tcl
- tk
- otcl
- tclcl

- NS-2
- nam

Aufeinander abgestimmt Pakete finden sich auf der Website des NS-2[Unic]. Unter Unix werden diese Pakete jeweils mittels folgender Befehle entpackt:

`tar xvzf Paketname.tar.gz` bzw. `tar xvzf Paketname.tgz`

Danach muss in den entsprechenden Ordner ein Wechsel erfolgen um das Paket zu kompilieren. Dies geschieht für einen Nutzer jeweils mittels

```
cd Paketordner
./configure
make
```

oder alternativ für alle Nutzer mittels

```
./configure --prefix=Installationsverzeichnis
make
make install
```

Für tcl und tk muss zusätzlich in den Unterordner des entsprechenden Systems gewechselt werden, beispielsweise nach `tcl-8.2/unix`.

4.3 Installation unter Windows

Für Windows existieren bereits vorkompilierte Binärdateien, die direkt ausführbar sind. Sollte es doch notwendig sein, den NS-2 manuell zu kompilieren, so befindet sich unter http://www.isi.edu/nsnam/ns/ns-win32-build.html eine ausführliche Anleitung.

4.4 Integration der Bewegungsmodelle

NS-2 erlaubt das Festlegen der Position eines Nodes mittels tcl-Befehl. Diese können entweder direkt im Simulationsskript stehen oder in einer Datei, welche von dort mittels `source`-Befehl aufgerufen wird.

Die Koordinaten eines Nodes werden mittels

```
$node set X_ <X-Koordinate>
```

```
$node set Y_ <Y-Koordinate>
$node set Z_ <Z-Koordinate>
```

festgelegt.
Weiterhin besteht die Möglichkeit Zielkoordinaten und Geschwindigkeit festzulegen:

```
$node setdest <X-Koordinate> <Y-Koordinate> <Geschwindigkeit>
```

Beide Befehle können auch jeweils zu einem bestimmten Zeitpunkt aufgerufen werden. Dafür muss dem Scheduler des NS-2 der Zeitpunkt und der Befehl mitgeteilt werden:

```
$ns at <Zeitpunkt> "$node set X_ <X-Koordinate>"
```

bzw.

```
$ns at <Zeitpunkt> "$node setdest <X-Koordinate> <Y-Koordinate>
<Geschwindigkeit>"
```

Zu Beachten ist dabei, dass zur Zeit der NS-2 zwar die Angabe von z-Koordinaten unterstützt, die z-Koordinate wird jedoch für Berechnungen ignoriert.

4.5 Anpassen der grafischen Ausgabe

Der Network Animator (NAM) erlaubt die Beobachtung der Bewegung der Nodes sowie das Erfassen der einzelnen Pakete. Das Betrachten von bestehenden Verbindungen bzw. der aktuell genutzten Route ist dadurch nicht einfach. Das Tool **viewtraffic** ist in der Lage die bestehenden Routen aus der NS-2-Tracedatei auszulesen und darzustellen. Weitere Informationen befinden sich in Abschnitt 5.4.
Weiterhin ist es durch entsprechende Programmierung der Simulation möglich, Dateien zu erstellen, welche tabellarisch die gesendeten Bytes und Pakete je Verbindung und Sekunde in Textdateien aufzeichnen. Dies ist ebenfalls für erneut gesendete Pakete nach fehlendem ACK möglich.
Diese Dateien können mittels MATLAB oder OCTAVE ausgewertet werden. Insbesondere zum Vergleich der Routing-Verfahren bezüglich Datenrate und damit auch möglicher QoS-Aspekte, bieten sich diese Tools an.

5 Eigene Erweiterungen

5.1 Einleitung

Die Integration neuer Bewegungsmodelle und deren grafische Darstellung erfordern einige Erweiterungen des NS-2. Bewegungsmodelle sind zwar per Hand mittels Skriptsprache TCL erstellbar, dies ist aber sehr aufwändig und fehleranfällig. Deshalb wurden drei Tools entwickelt, welche den Umgang mit dem NS-2 vereinfachen sollen.

5.2 Implementierte Bewegungsmodelle

5.2.1 Grundlagen

Der NS-2 ist ein Simulator, der mit beliebigen Bewegungsmustern arbeiten kann. Es besteht nicht die Notwendigkeit, sich auf ein Bewegungsmodell zu beschränken. Die Mobilität der Teilnehmer kann in einer externen Datei definiert werden. Somit ist es möglich verschiedene Bewegungsmodelle gleichzeitig in einer Simulation zu benutzen. Beispielsweise könnte man in einer Stadtsimulation schnelle gerichtete und langsame zufällige Bewegungen gleichzeitig simulieren.
Ziel ist es deshalb, das Bewegungsmodell als Baukasten von Teilmodellen aufzubauen. Die Umsetzung erfolgte dabei in einem Kommandozeilentool auf C++ Basis.

5.2.2 Guérin-ähnliches Modell

Dieses gedächtnislose Modell stellt ein abgewandeltes Guérin-Modell dar. Das Spiegelungsprinzip wird aufgrund fehlender Zellen nicht verwendet. Im Gegensatz zum Guérin-Modell kann sich auch die Geschwindigkeit ändern. Richtung und Geschwindigkeit sind unabhängig von früheren Zuständen. Die Verweildauer in den Bewegungszuständen ist negativ exponential verteilt (Abbildung 4).
Nach Ablauf der Verweildauer wird eine neue Richtung festgelegt (Gleichung 1).

$$\varphi_{neu} = 0 \ldots 360°, gleichverteilt \tag{1}$$

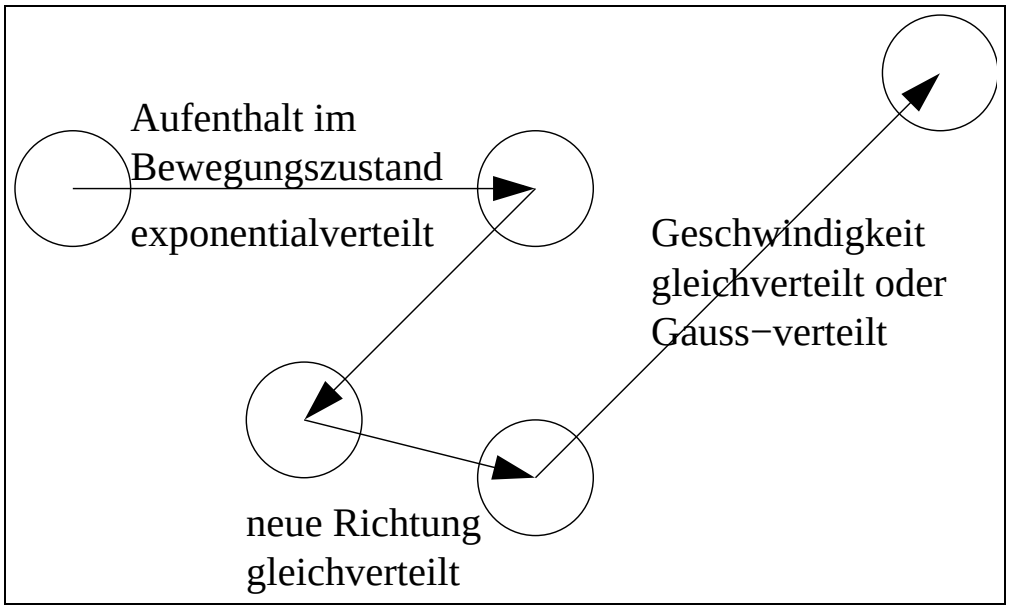

Abbildung 4: Markov-Bewegung: zufällig anhand von Verteilungen

Zur gleichen Zeit wechselt der Teilnehmer auch die Geschwindigkeit. Die Geschwindigkeit v wird ebenfalls zufällig gewählt. Es besteht die Möglichkeit diese Geschwindigkeit anhand einer Gauss-Verteilung oder anhand einer Gleichverteilung zu bestimmen[4].

5.2.3 Zonoozi-ähnliches Modell

Die grundlegende Modellierung ist ähnlich dem Zonoozi-Modell. Die Richtung und die Geschwindigkeit sind von vorherigen Werten abhängig (Abbildung 5). Parameter sind dabei die maximale Änderung für beide Werte. Der Abstand zwischen den Änderungen

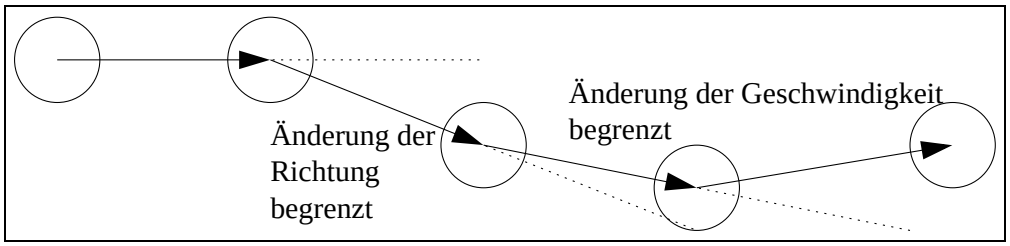

Abbildung 5: Bewegung abhängig von vorheriger Bewegung

kann anhand beliebiger Verteilungen erfolgen. Üblicherweise wird jedoch die negative

[4]Derzeit ist nur die Gleichverteilung umgesetzt

Exponentialverteilung verwendet, da diese einerseits leicht zu berechnen ist (Abschnitt 5.8.3). Außerdem zeigt die Erfahrung, dass sich viele Praxisereignisse mittels negativer Exponentialverteilung approximieren lassen.

5.2.4 Bewegung anhand von Gebietsdefinitionen

Eine flexible Art der Bewegungsmodellierung besteht in Gebietsdefinitionen. Dabei definieren die Punkte eines Polygons die Grenzen von Gebieten. Für diese Gebiete lässt sich eine Vorzugsrichtung, eine mittlere Abweichung von dieser Richtung, eine maximale Geschwindigkeit, eine mittlere Bewegungszeit und eine Pausenzeit definieren. Die Gebietsdefinition wird dabei in einer Textdatei vorgenommen.
Alle Elemente hinter dem Zeichen „#“werden als Kommentar aufgefasst. Jede Zeile steht für ein Gebiet. Die ersten Ziffern der Zeile stehen dabei für:

- Gebietstyp
  - 0=normale Gebiet mit Vorzugsrichtung
  - 1=Straßenähnliches Gebiet, Vorzugsrichtung bidirektional
  - 2=Kreuzung
- Geschwindigkeit in Meter je Sekunde
- mittlere Bewegungsdauer
- mittlere Pausendauer
- Richtung
- Streuung Richtung

Danach folgen paarweise die x- und y-Koordinaten der Punkte des umschließenden Polygons. Eine konzentrische Rechteckbewegung der Teilnehmer entgegen dem Uhrzeigersinn wird in folgender Form definiert:

```
#Gebietsdefinition
#Typ, Geschwindigkeit, mittlere Bewegungsdauer, mittlere Pausendauer, Richtung,Streuung Richtung
#erste Zeit
0 3 10 10 0      0.0            0 0            800 0          400 400
0 3 10 10 4.71   0.0            0 0            0 800          400 400
0 3 10 10 3.14   0.0            0 800          800 800        400 400
0 3 10 10 1.57   0.0            800 800        800 0          400 400
```

Die erste Zeile bedeutet dabei: der Gebietstyp ist 0 (normales Gebiet mit Vorzugsrichtung), die maximale Geschwindigkeit beträgt 3 m/s, Bewegungen dauern 10 Sekunden gefolgt von einer Pause von 10 Sekunden, Vorzugsrichtung ist 0 Grad Bogenmaß, mit einer mittlere Streuung von 0.0. Es existieren drei Punkte mit den Koordinaten (0,0) (800,0) und (400,400). Es wird also ein Dreieck definiert. Alle 4 Gebiete ergeben ein Szenario mit konzentrischen Rechteckbewegungen (Abbildung 6).

5.2.5 Szenario 1: konzentrische Bewegungen

Dieses Szenario, ist dem vorherigen Kapitel entnommen.

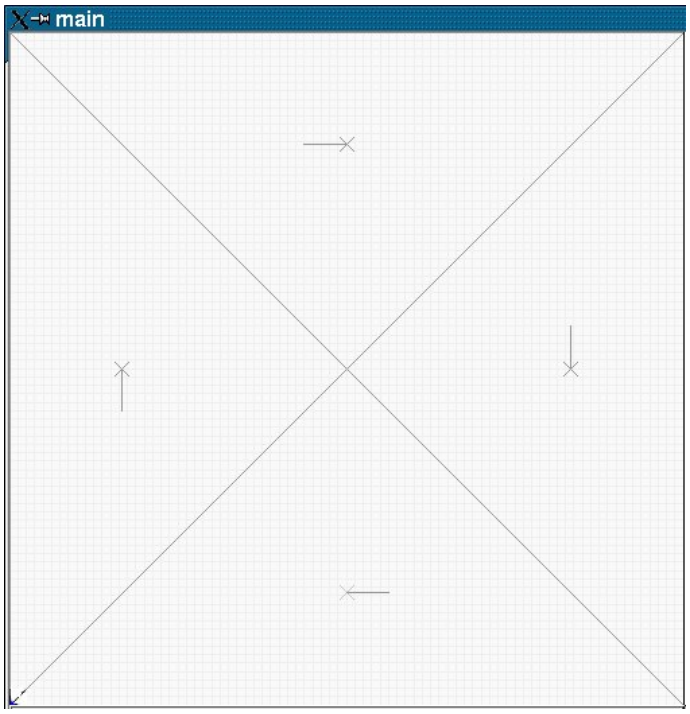

Abbildung 6: Vier Gebiete die konzentrische Rechteckbewegungen definieren

In diesem Szenario bewegen sich 20 Teilnehmer in einem Feld mit den Ausmaßen 800m x 800m. Die Bewegung ist dabei entgegen dem Uhrzeigersinn. Die Geschwindigkeit beträgt 5 Meter in der Sekunde. Die Dauer der Bewegung ist negativ exponentialverteilt mit einer mittleren Dauer von 10 Sekunden. Es wird dabei eine TCP-Verbindung,ohne Slow-Start, zwischen zwei zufällig gewählten Nodes initiiert. Der Startzeitpunkt der Verbindung ist ebenfalls zufällig und liegt zwischen Null und halber maximaler Simulationszeit. Durch den unterschiedlichen Bahnumfang, der abhängig vom Abstand zum Mittelpunkt ist, dauert eine Umrundung unterschiedlich lang. Dadurch ändert sich die Anzahl der Hops im Lauf der Simulation.

5.2.6 Szenario 2: Rennsport

Die Gebietsdatei diese Szenarios sieht folgendermaßen aus:

```
#Gebietsdefinition
#Typ,Geschwindigkeit, mittlere Bewegungsdauer, mittlere Pausendauer, Richtung,Streuung Richtung
#Mitte
0 10 1 0 0      0               30 30    470 30 470 470         30 470

#Strassen
0 15 1 0  3.142 0               30 0             500 0          500 30          30 30
0 15 1 0  4.713 0               470 30           470 500        500 500         500 30
0 15 1 0  0     0               470 500          0 500          0 470           470 470
0 15 1 0  1.571 0               0 470            30 470         30 0            0 0
```

Im Gegensatz zum vorherigen Szenario findet die Bewegung nur auf dem äußeren Ring statt (Abbildung 7). Die innere Fläche ist dabei groß genug um die Datenübertragung durch sie hindurch zu verhindern. Das Routing muss also evtl. sehr lange Wege finden, um eine Verbindung aufrecht zu erhalten. Die Geschwindigkeit der Teilnehmer ist dabei gleich verteilt zwischen 0 m/s und 15 m/s. Die Bewegung wird im Mittel einmal in der Sekunde angepasst.

5.2.7 Szenario 3: ETSI-ähnliches Manhattan-Modell

Dieses Szenario ist dem ETSI-Modell aus Abschnitt 3.2.7 entnommen. In Abbildung 8 wird gezeigt, in welcher Weise sich die Teilnehmer auf den Straßen zwischen den Häuserblöcken bewegen. Die Gebietsdatei dieses Szenario befindet sich im Anhang A. Die Geschwindigkeit auf den Strassen beträgt 3 Meter je Sekunde. In diesem Szenario

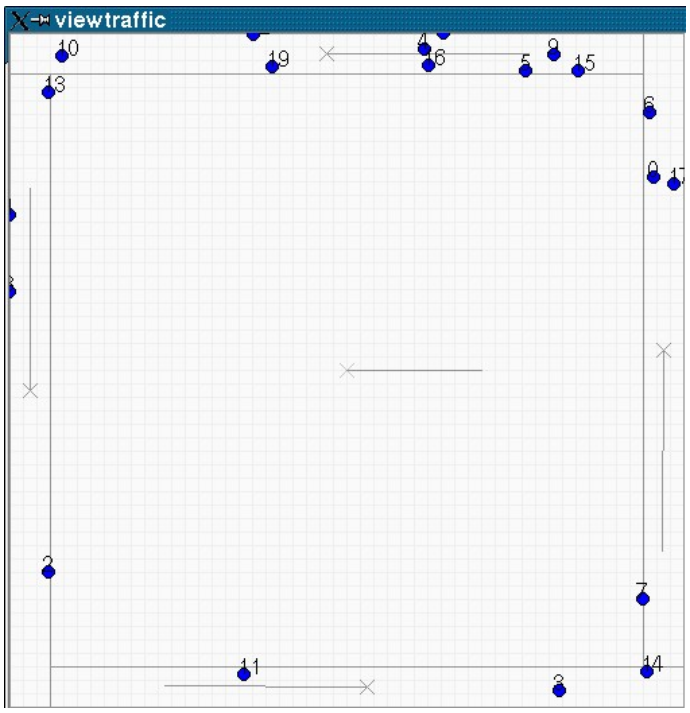

Abbildung 7: Rennsport-Szenario

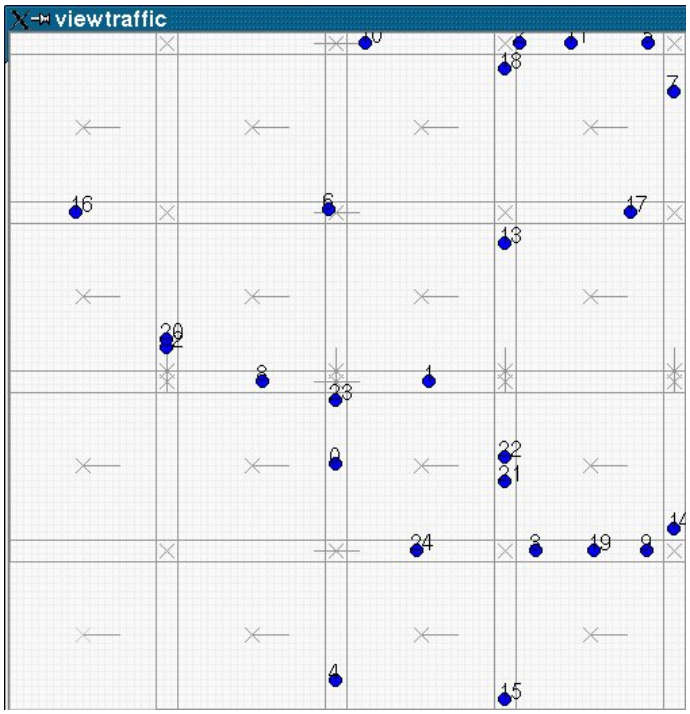

Abbildung 8: Manhattan-Szenario, ähnlich ETSI

wird wieder eine zufällige TCP-Verbindung aufgebaut. Die Wechselwahrscheinlichkeit an den Kreuzungen beträgt 50 Prozent. Trifft ein Teilnehmer den Rand des Simulationsgebietes, so kehrt er seine Richtung um.

5.3 manager.tcl

manager.tcl ist eine Oberfläche (Abbildung 9), welche es ermöglicht, sehr schnell Szenarien zu erstellen und diese aufzurufen. Sie ist in TCL/TK geschrieben und wird mit der Windowing Shell „wish“gestartet, welche in TCL/TK enthalten ist.

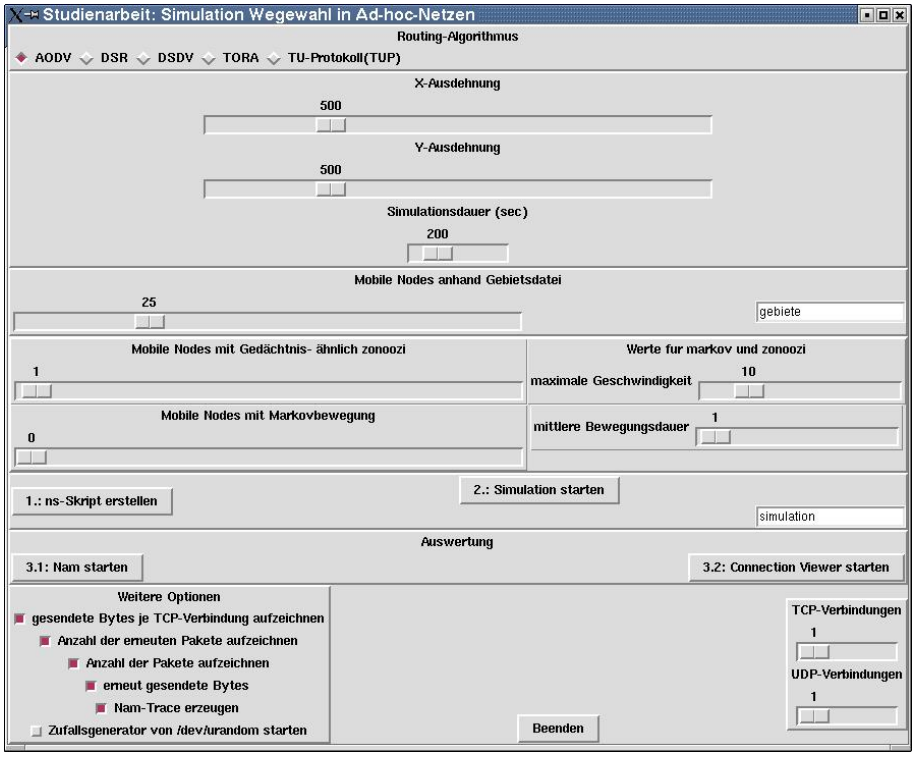

Abbildung 9: Oberfläche des manager.tcl

Als Option kann eine Parameterdatei übergeben werden, z.B. `manager.tcl rennen`.

5.4 viewtraffic

Dieses Programm dient dazu, die aktuellen Routen von TCP-Verbindungen bzw. UDP-Strömen darzustellen. Dafür liest es die Trace-Datei der Simulation komplett ein, um

diese auszuwerten. Hierin besteht ein Unterschied zum Network Animator (NAM). NAM liest immer nur die Teile des Namtraces ein, welche gerade benötigt werden. Da die Darstellung von den vorherigen Traces und Ereignissen abhängen, muss NAM mehr Ereignisse einlesen, als oftmals notwendig. Dies kann zu unangenehmen Wartezeiten führen. Das komplette Einlesen hat aber den Nachteil eines erhöhten Speicherplatzbedarfes. Parameter für viewtraffic sind `-a <Tracedatei>`, `-g <gebietsdatei>`, `-x <X-Ausdehnung>` und `-y <Y-Ausdehnung>`.

5.5 gentraffic

Dieses Tool erzeugt die Skripte für die Bewegung der einzelnen Nodes anhand der Szenariodefinition. Parameter dabei sind:

- -x X-Ausdehnung des Gebietes
- -y Y-Ausdehnung des Gebietes
- -g Name der Datei, die die Gebietsdefinitionen enthält
- -n Anzahl der Nodes die sich anhand der Gebietsdefinition bewegen
- -z Anzahl der Nodes die sich anhand eines Zonoozi-Modells bewegen
- -m Anzahl der Nodes die sich anhand eines Markov-Modells bewegen
- -t Simulationsdauer
- -f Anzahl der TCP-FTP-Verbindungen
- -c Anzahl der UDP-CBR-Verbindungen
- -v maximale Geschwindigkeit für die Markov- und Zonoozi-Nodes
- -d mittlere Bewegungsdauer für die Markov- und Zonnozi-Nodes

gentraffic wird üblicherweise von manager.tcl aufgerufen, indem alle diese Parameter grafisch eingestellt werden können.

5.6 Aufbau von gentraffic und viewtraffic

5.6.1 Klassendiagramm gentraffic

Abbildung 10 zeigt ein einfaches Klassendiagramm von gentraffic. Die Klassen Config und Random sind universelle Klassen. Random dient der Generierung von Zufallswerten und ist in Abschnitt 5.8 näher beschrieben.

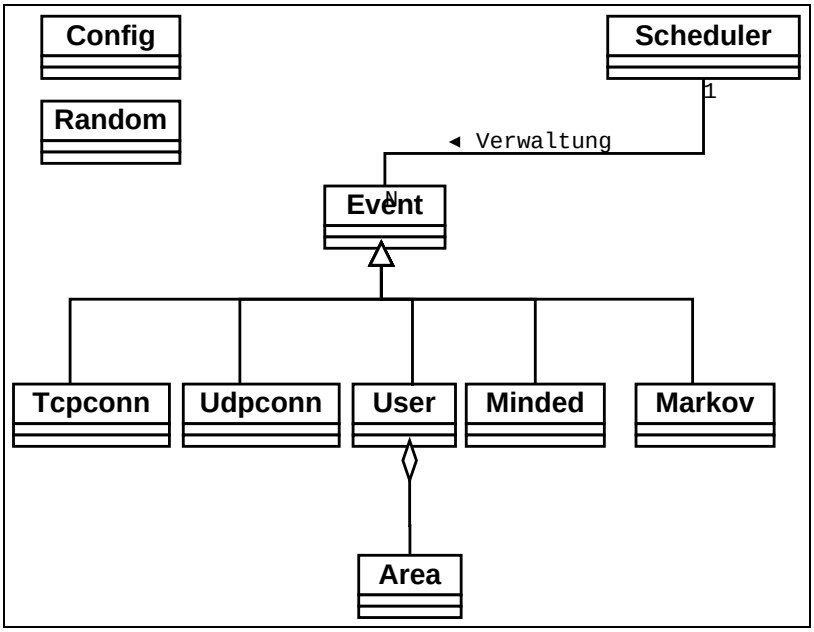

Abbildung 10: Klassendiagramm gentraffic

5.6.2 Klassendiagramm viewtraffic

Abbildung 11 zeigt ein einfaches Klassendiagramm von viewtraffic. Dabei ist die Klasse Config eine universelle Klasse, das fast überall genutzt wird.

5.6.3 Klasse Area

Die Klasse Area dient zum Einlesen der Gebietsdateien und liefert auf Anfrage die Parameter zu bestimmten Koordinaten. Methoden sind:

- Area(name): Der Konstruktor. Es wird die Gebietsdatei übergeben. Sie wird eingelesen und die Daten werden intern gespeichert

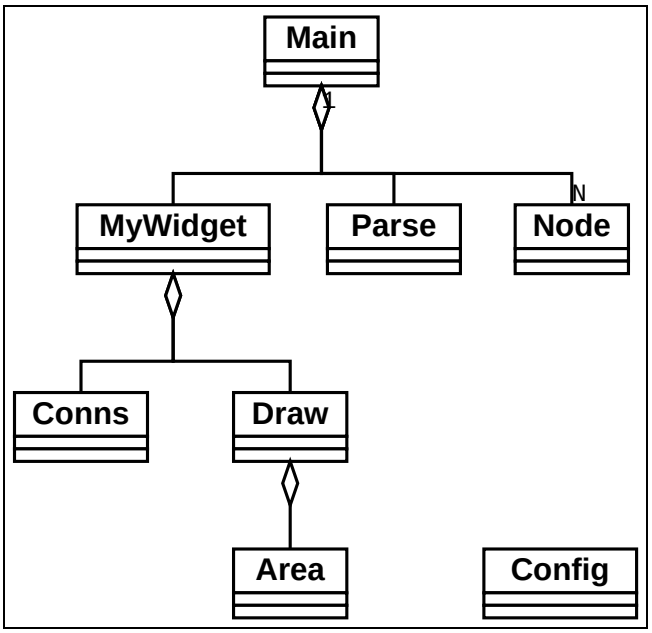

Abbildung 11: Klassendiagramm viewtraffic

- ~ Area(): Der Destruktor
- get (x,y,zeit): es wird das zu den Koordinaten passende Gebiet ausgegeben
- instance(): Die Klasse muss nur einmal initialisiert werden. Wenn bereits ein Object existiert, liefert instance einen Zeiger darauf
- getareas(zeit): gibt alle Gebiete dieses Zeitpunktes aus

5.6.4 Klasse Scheduler

Dies Klasse Scheduler dient dazu, die Ausgaben der einzelne Verkehrsgeneratoren zeitlich zu ordnen und diese geordnet auszugeben. Er kann dabei nur Objekte der Klasse Event (Abschnitt 5.6.5) verwalten. Methoden sind:

- Scheduler(zeit): Der Konstruktor. Parameter ist die Zeitdauer der Simulation.
- ~ Scheduler(): Der Destruktor
- start (): Der Scheduler wird gestartet. Das erste existierende Ereignis wird aufgerufen und aus der Liste gelöscht. Nach Beendigung dieses Ereignisses wird das

nächste aufgerufen

- schedule (event, zeit): Ein Ereignis wird zu einem Zeitpunkt eingeplant
- instance (): Gibt die Adresse eines existierenden Schedulers aus

5.6.5 Klasse Event

Dies ist die Ereignisklasse. Das Aufrufen wird von der Klasse Scheduler (Abschnitt 5.6.4) verwaltet.
Methoden:

- Event(zeit): Der Konstruktor. Als Parameter wird die Simulationsdauer angegeben
- ~ Event(): Destruktor
- run (zeit): Aufruf. Diese Funktion ist virtuell und wird von Scheduler aufgerufen. Von event abgeleitete Klasse müssen sie selbst implementieren. Hier werden üblicherweise die Berechnungen getätigt
- reschedule (zeit): Diese Methode dient dazu, dem Scheduler den nächsten Aufruftermin mitzuteilen.

5.6.6 Klasse Markov, Minded, User, TcpConn und Udpconn

Diese Klassen sind von der Klasse Event abgeleitet. Sie können deshalb alle vom Scheduler verwaltet werden. In diesen Klassen werden die verschiedenen Verkehrsmodelle berechnet und Ausgaben getätigt.

5.6.7 Klasse Parse

Diese Klasse liest die Trace-Datei ein und wertet sie aus. Methoden:

- Parse (Dateiname): Der Konstruktor. Als Parameter wird der Dateiname des Tracefiles übergeben.
- ~ Parse(): Der Destruktor.
- end (): Ja/nein-Aussage, ob es weitere Ereignisse gibt
- givenext (): Gibt die Werte des nächsten Ereignisses aus
- give_start (ID): Gibt die Startzeitpunkte der Verbindungen aus
- getcolor (ID): Gibt einen einmaligen Farbwert für eine Verbindung aus

5.6.8 Klasse Node

Die Klasse Node wird für jeden Node einmal instanziiert. Jede Instanz speichert die Ereignisse des betreffenden Nodes. Methoden:

- Node (): Der Konstruktor.
- ~ Node (): Der Destruktor.
- newentry (Ereignis): Ein Ereignis wird gespeichert.
- getentry (Zeit): Es wird der Zustand des Nodes zu einem bestimmten Zeitpunkt ausgegeben. Dabei werden die Ereignisse abgearbeitet.

5.6.9 Klasse MyWidget

Diese Klasse dient zur Darstellung der Oberfläche. Als Methoden existiert nur der Konstruktor. Es werden ein Conn-Objekt, ein Draw-Objekt, ein Schiebereglern und ein Knopf zum Beenden platziert.

5.6.10 Klasse Conns

- Conns(Parseobjekt): Der Konstruktor. Parameter ist ein Zeiger auf ein Parse-Objekt.
- setMaxTime (zeit): Legt die maximal anzeigbare Simulationsdauer fest.
- neu (Zeit): neu wird beim Ändern des Schiebereglers aufgerufen und aktualisiert den Wert der aktuellen Simulationszeit
- paintEvent(): Neudefinition der QT-Basismethode. Hier werden die Verbindungsbalken dargestellt.

5.6.11 Klasse Draw

- Draw (Nodesobjekte, Parseobjekt): Der Konstruktor.
- paintEvent(): Neudefinition der QT-Basismethode. Hier werden das Simulationsgebiet und die Nodes gezeichnet.
- neu(Zeit): neu wird beim Ändern des Schiebereglers aufgerufen.

5.7 Installieren der Tools

Die Datei tools.tgz ist mitteln `tar xvzf tools.tgz` zu entpacken. Danach muss das Makefile editiert werden. Die Variable QTDIR muss auf das QT-Verzeichnis gesetzt werden. Ist dieses unbekannt, kann mittels `make getqt` QT automatisch gefunden werden. Nach Anpassung des Makefiles werden die Tools mittels

```
make
make install
```

nach /usr/local/bin installiert. Üblicherweise befindet sich dieses Verzeichnis im Pfad und die Tools können aus beliebigen Verzeichnissen aufgerufen werden.

5.8 Erzeugen von Verteilungen

5.8.1 Grundlagen

Zufallsgeneratoren erzeugen üblicherweise Werte zwischen 0 und 1. Dafür werden rückgekoppelte Schieberegister oder Algorithmen wie Wichmann-Hill [WH81] verwendet. Der erzeugte Wert sei RN.

5.8.2 Gleichverteilung

Eine Gleichverteilung zwischen x_{min} und x_{max} kann aus dem Zufallsgenerator erzeugt werden:

$$Wert = RN \cdot (x_{max} - x_{min}) + x_{min} \tag{2}$$

5.8.3 negative Exponentialverteilung

Ein Generator für negativ exponential verteilte Werte ist ebenfall aus einem Zufallsgenerator ableitbar. Die Verteilungsfunktion lautet:

$$F(t) = 1 - e^{-\frac{t}{\tau}} \tag{3}$$

Der Wertebereich von $F(t)$ entspricht dem Wertebereich von RN. Es muss also nur $F(t)$ durch die RN ersetzt und nach t umgestellt werden, um einen Zufallsgenerator für negativ exponential verteilte Zeiten zu erhalten:

$$t = -\tau * log(RN) \tag{4}$$

5.8.4 Gauss-Verteilung

In [DR] finden sich JAVA-Implementierungen vieler weiterer Zufallsgeneratoren, beispielsweise folgender Algorithmus zur Bestimmung eines Wertes einer Gaussverteilung:

```
// generate N(0,1)
// E(X)=0 ; Var(X)=1
```

```
public synchronized double nextGaussian() {
  if (!haveNextGaussian) {
    double v1=nextUniform(),v2=nextUniform();
    double x1,x2;
    x1=Math.sqrt(-2*Math.log(v1))*Math.cos(2*Math.PI*v2);
    x2=Math.sqrt(-2*Math.log(v1))*Math.sin(2*Math.PI*v2);
    nextGaussian=x2;
    haveNextGaussian=true;
    return x1;
  } else {
    haveNextGaussian=false;
    return nextGaussian;
  }
}
```

6 Simulation und Bewertung

6.1 Vergleich der Routingverfahren

6.1.1 Kriterien

Die Bewertung von Routing-Protokollen sollte stets anwendungsbezogen stattfinden. Bezüglich der eingesetzten Szenarien sind dabei jedoch immer folgende Werte interessant.

- Performance
- Zuverlässigkeit der Verbindung
- Quality of Service
- Recoverygeschwindigkeit nach Verbindungsabbruch

6.1.2 Szenario 1: Kreisbewegungen, eine TCP-Verbindung

Die Datenrate für das Szenario aus Abschnitt 5.2.5 ist in Abbildung 12 dargestellt.
Hierbei ist deutlich zu erkennen, dass sich die Datenrate zwischen 70 Sekunden und 150 Sekunden bei AODV, DSR und TORA sehr ähnlich sind. Bei ca. 108 Sekunden findet bei diesen drei Protokollen ein Wechsel von drei auf vier Hops statt, welcher die Datenrate leicht sinken lässt (Abbildung 13). Dabei ist zu beachten, dass der NS-2 bereits berücksichtigt, dass Störleistungen geringer sein müssen als Empfangsleistungen und deshalb die Datenrate auch beim Wechsel von 3 auf 4 Hops noch leicht sinkt.
Bei ca 146 Sekunden bricht die Verbindung zusammen, da die Entfernung zwischen Node 8 und 9 zu groß wird.
DSDV kommt aufgrund der Pause zwischen Topologie-Updates nicht nach, die Routen schnell genug zu aktualisieren, deswegen bricht die Verbindung nach Verlust der ersten Route ab.
Zum Zeitpunkt 265 Sekunden schaffen TORA und DSDV einen Connect. Die Route von TORA hält aber nur sehr kurz, während DSDV diesmal eine stabilere Lösung findet (Abbildung 14).
Bei allen Routingverfahren ist die Datenrate bei gleicher Hopanzahl ähnlich bzw. identisch. Der Overhead des Routingprotokolls fällt also nicht ins Gewicht.
AODV und DSDV finden mach 410 Sekunden wieder eine Route, welche sie bis 455 Sekunden halten können. Danach bricht diese Route weg. Zum Zeitpunkt 465 Sekunden finden AODV und TORA eine Ersatzroute, die in ähnlicher Form auch von DSDV gefunden wird, allerdings erst zu einem späteren Zeitpunkt (Abbildung 15).
Bei der Gesamtdatenrate (Abbildung 12) hat AODV mit 4158 KBytes den höchsten Wert vorzuweisenweit und liegt weit vor DSR und TORA. DSDV hat die schlechteste Gesamtrate.

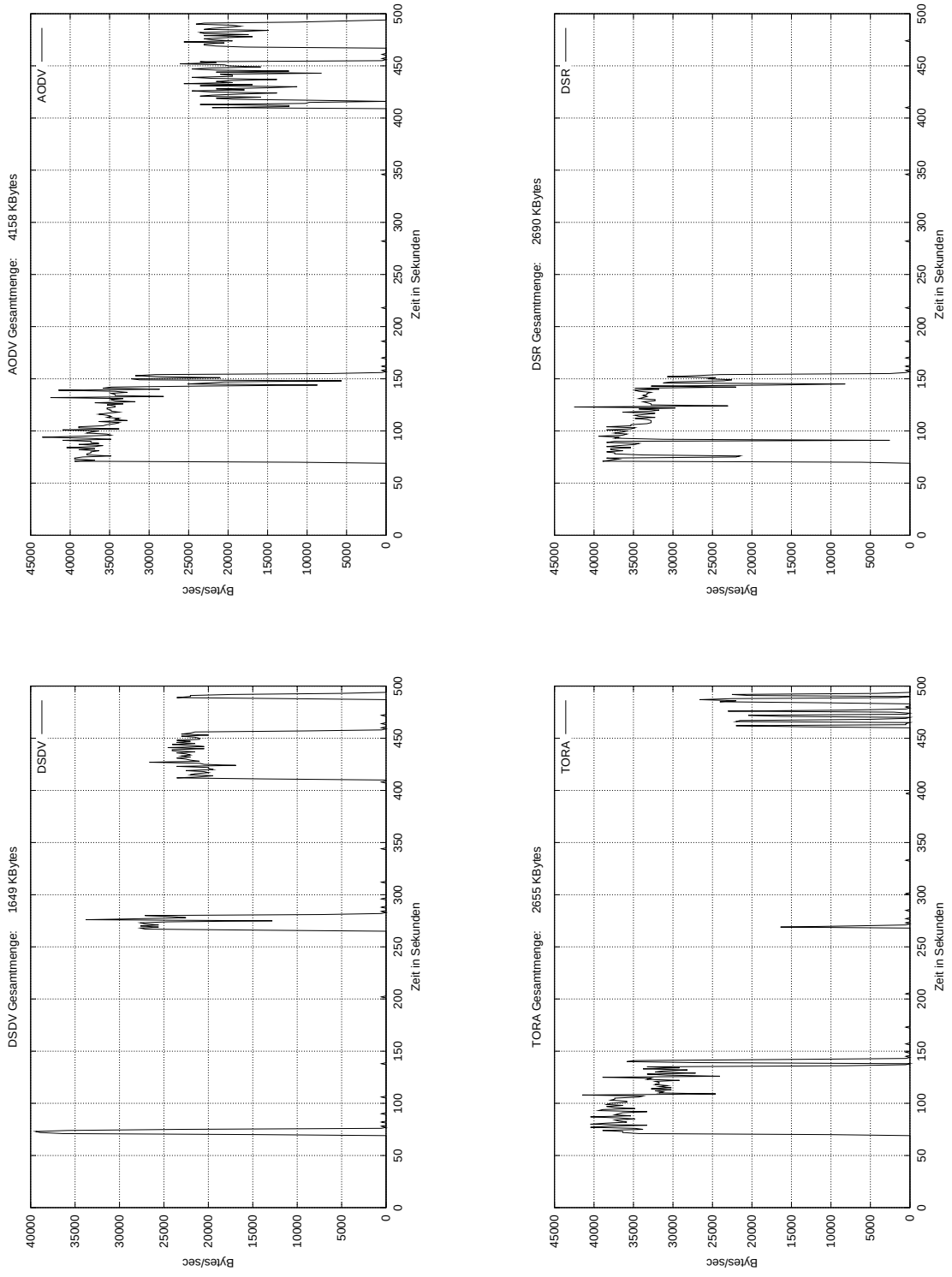

Abbildung 12: Datenraten des Szenarios „konzentrische Bewegung“

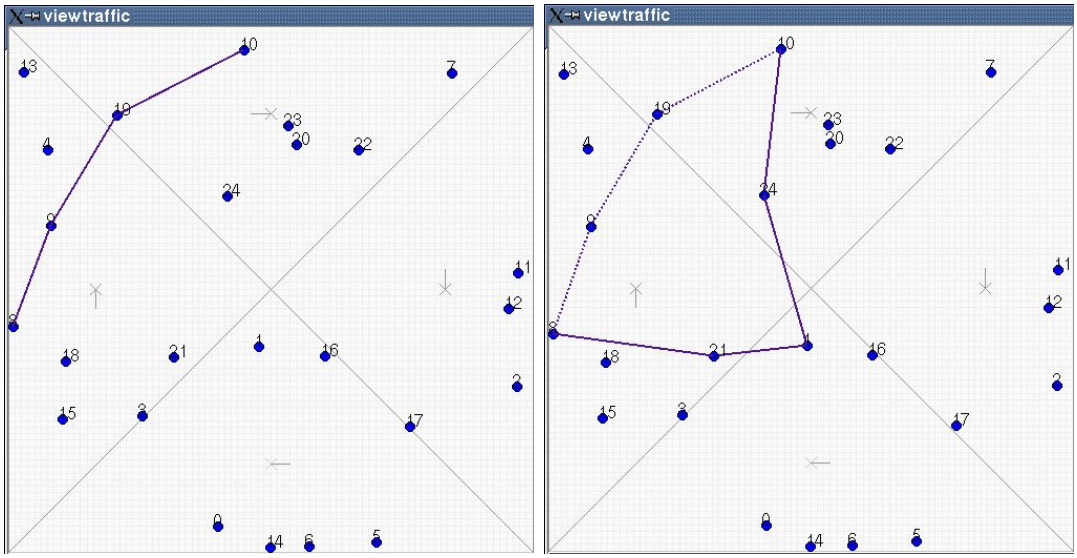

Abbildung 13: AODV, DSR und TORA: Routing bei 107 und 108 Sekunden

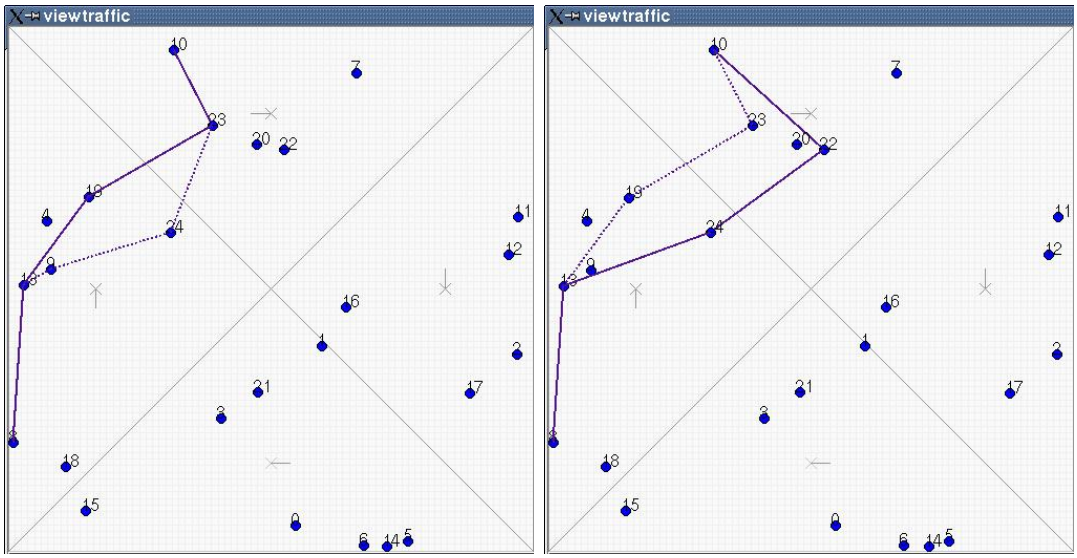

Abbildung 14: Routing bei 268 Sekunden. TORA und DSDV

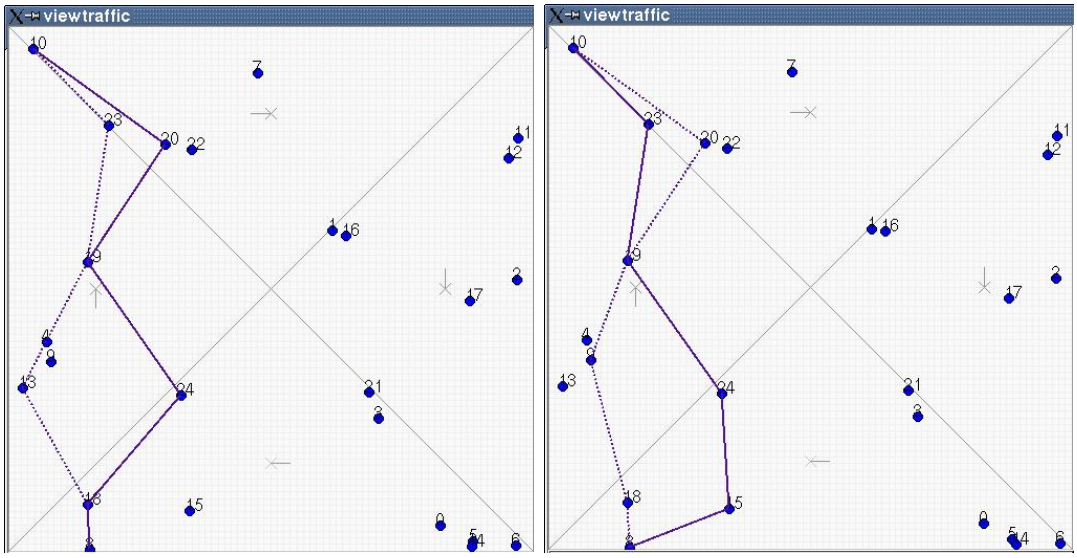

Abbildung 15: Route von AODV/TORA im Gegensatz zur Route von DSDV

6.1.3 Szenario 2: Rennsport

In Abbildung 16 sind die Übertragungsraten des Szenarios Rennen dargestellt. Die FTP- Übertragung erfolgt nach 126 Sekunden.
Alle vier verschiedenen Routingverfahren finden dabei sofort eine Route (Abbildung 17), welche jedoch kurz darauf wieder zusammenbricht.

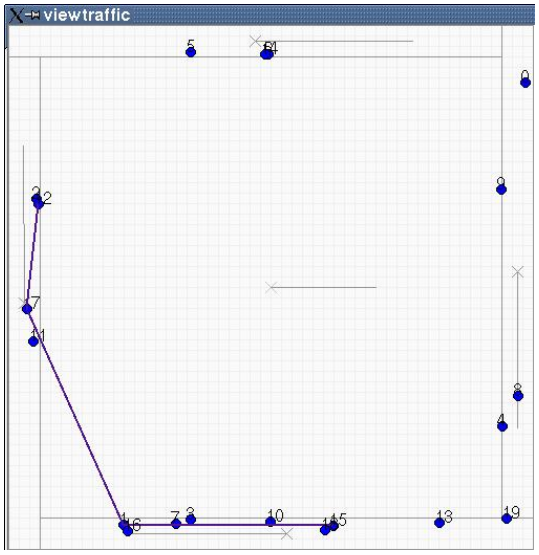

Abbildung 17: Eine mögliche Route

In diesem Szenario wechselt die Geschwindigkeit der Teilnehmer im Mittel nach einer Sekunde. Dadurch wird der Abstand zwischen den Nodes kleiner und größer, was dazu führt, dass die Routen relativ oft neu bestimmt werden müssen. Hierbei hat insbesondere DSDV Probleme, da es die Routen im Gegensatz zu den anderen Verfahren nicht bei Bedarf bestimmt, sondern im 15 Sekunden- Abstand Tabellen austauscht. Dies ist insbesondere deshalb ein Problem, da sich die Routen bei diesem Szenario teilweise komplett ändern müssen, wie z.B. in Abbildung 18 dargestellt.

Der Einbruch der Datenrate bei 410 Sekunden resultiert aus von einer Erhöhung der Hopanzahl. Die bisherige Route bestand aus sehr langen Verbindungen, bei denen der Störeffekt aufgrund der Entfernung nicht mehr zum Tragen kam. Die Teilverbindungen der neuen Route sind hingegen kurz genug, so dass sich entfernte Nodes noch stören und dadurch die Datenrate sinkt (Abbildung 19).
Auch bei diesem Szenario hat AODV die beste Gesamtdatenrate (Abbildung 16), DSR und TORA liefern schlechtere Werte. DSDV hat mit diesem Szenario große Probleme mit der schlechtesten Gesamtrate.

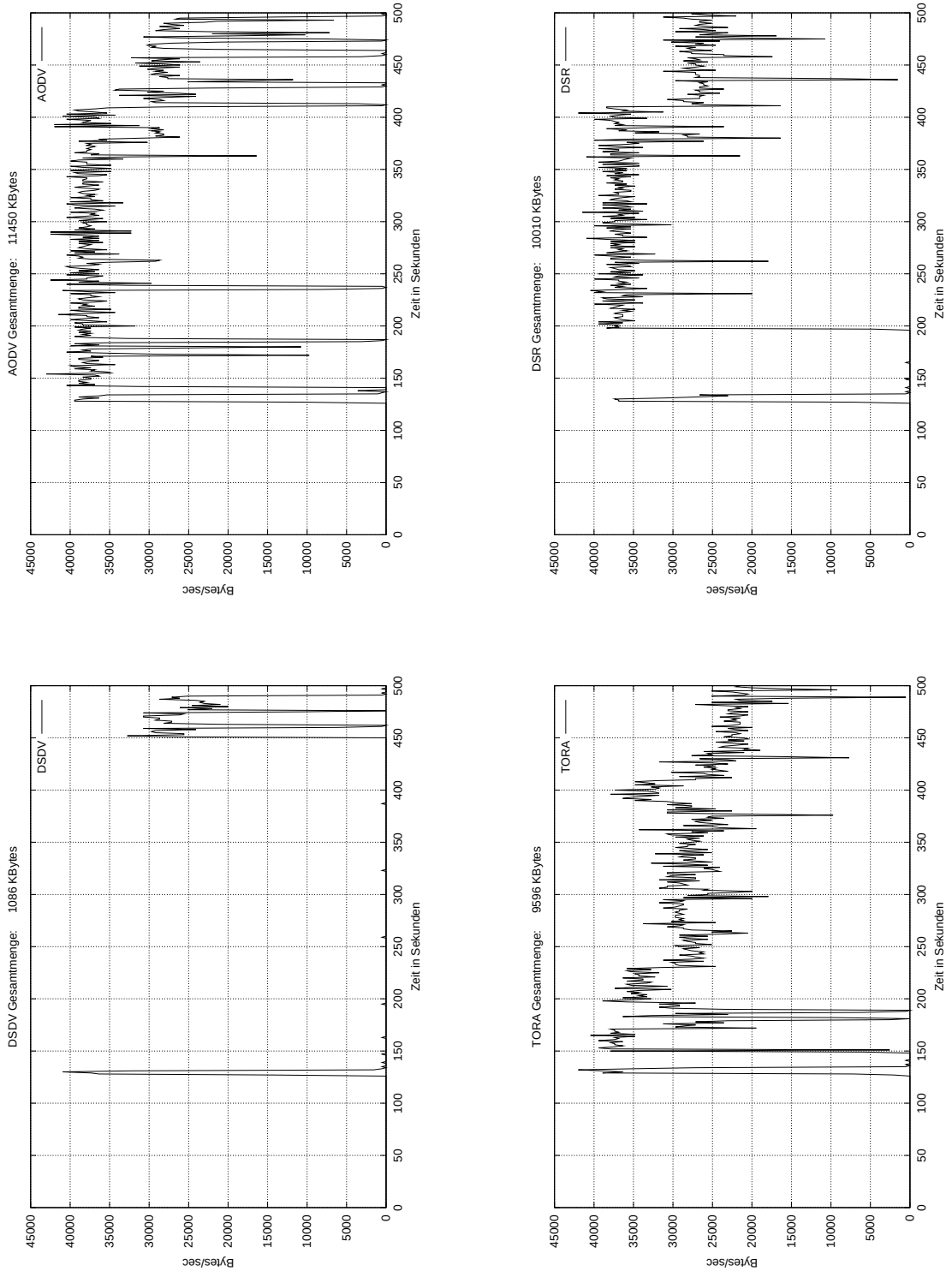

Abbildung 16: Datenrate des Szenarios „Rennsport“

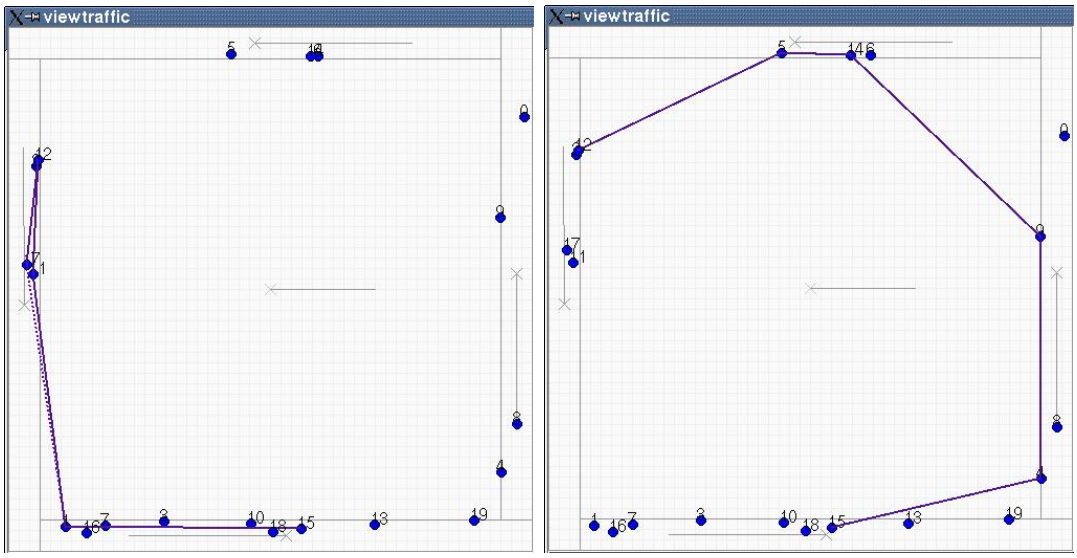

Abbildung 18: Wechsel der Route nach 133 Sekunden

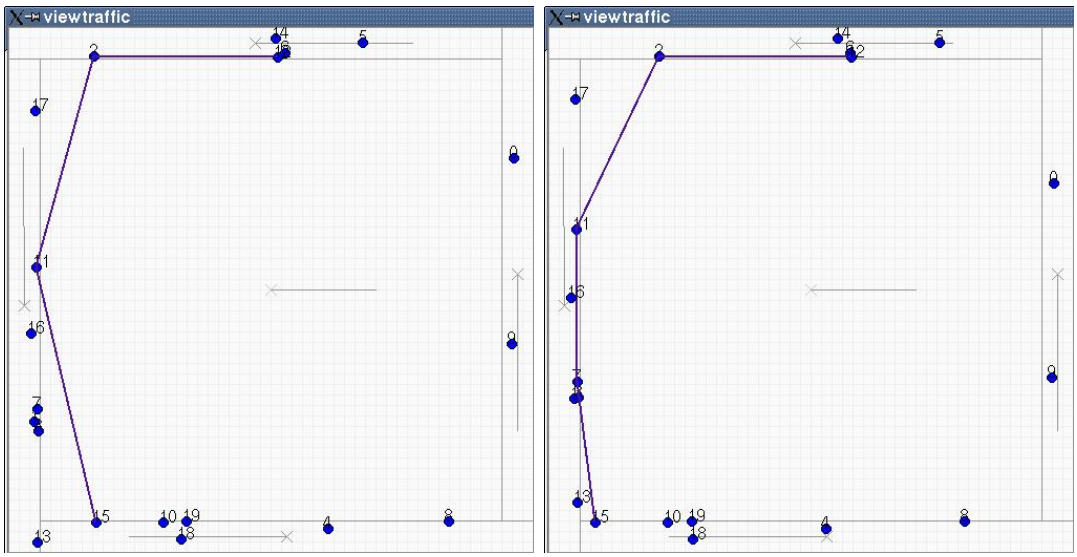

Abbildung 19: Hopanzahl beeinflusst Datenrate

6.1.4 Szenario 3: Manhattan

Abbildung 20 zeigt die Datenraten im Szenario „Manhattan“.

Bei 165 Sekunden finden alle Verfahren eine Route, welche aber nur kurz Bestand hat. Bis auf TORA sind alle Verfahren in der Lage eine Ersatzroute zu definieren.

Eine zweite Route zum Zeitpunkt 408 wird nur DSDV, AODV und TORA gefunden (Abbildung 21). DSR kann keine Route finden. Vergleichend zu den vorherigen Szenarien haben hier die Nodes nicht die gleiche Vorzugsrichtung, sondern bewegen sich unabhängig voneinander. Dies spiegelt sich auch in den etwas abweichenden Resultaten wider. Die höchste Gesamtdatenrate besitzt wieder AODV, gefolgt von DSDV und DSR. TORA schneidet bei diesem Szenario sehr schlecht ab.

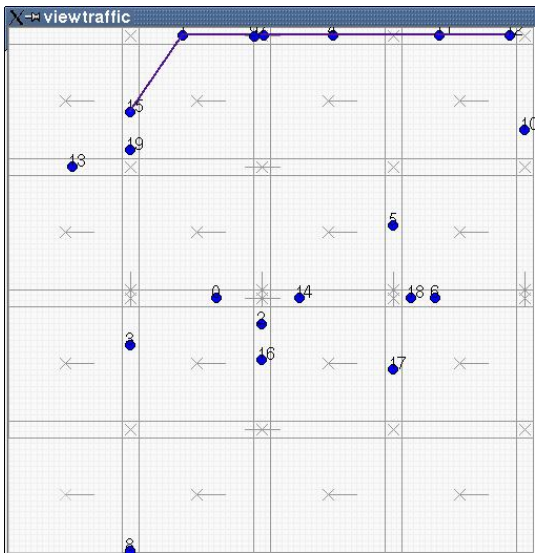

Abbildung 21: Route von DSDV, AODV und TORA

Es wird deutlich, dass die Wahl des Szenarios eine wichtige Rolle für die Wahl des Routingverfahrens spielt. Das Routingverfahren sollte nach den Anforderungen des Szenarios gewählt werden. Verfahren die in allen Situationen die beste Lösung finden, scheinen nur schwer zu realisieren sein.

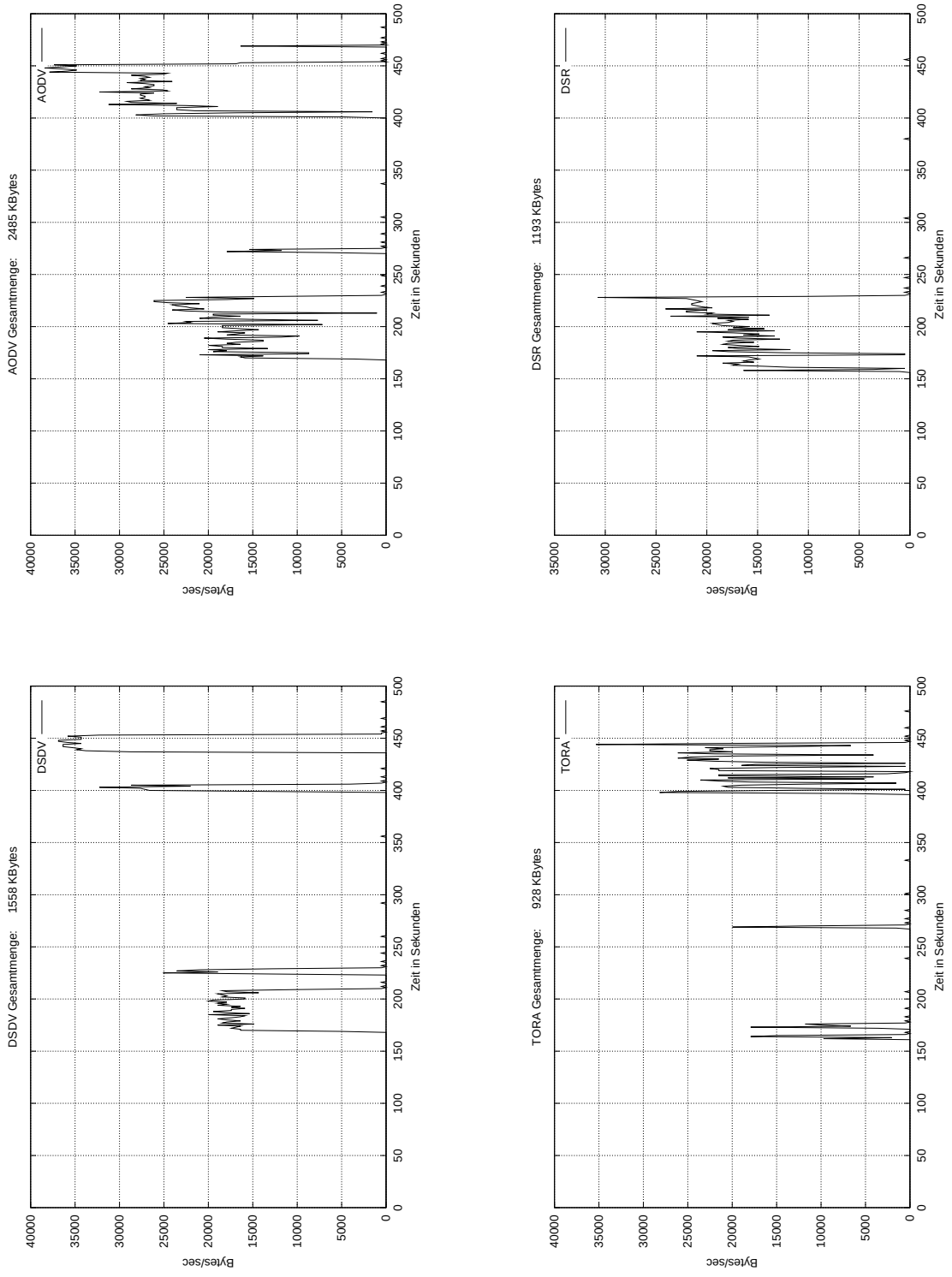

Abbildung 20: Datenraten des Szenarios „Manhattan“

7 Ausblick

Auf dem Gebiet der Ad-hoc-Netze wird immer noch sehr aktiv geforscht. Einige Aspekte dieser Arbeit sollten fortgesetzt bzw. weiter geführt werden.
Zukünftige Protokolle werden verstärkt auf Quality of Service Wert legen. Diese Protokolle können, sofern sie definiert sind, in den NS-2 integriert werden, um sie mit den vorgestellten Bewegungsmodellen und Szenarien zu simulieren. Eine beispielhafte Erweiterung des NS-2 wird im Anhang C vorgestellt.
Ebenfalls interessant ist die Definitionen neuer Szenarien. Das hier vorgestellte Gebietsmodell sollte in der Lage sein, die meisten Praxisszenarien annähernd umzusetzen. Zur Vereinfachung sollte aber die Entwicklung eines grafischen Editors für Gebietsmodelle in Betracht gezogen werden.
Die Gebietsdateien werden derzeit nur im Tool „viewtraffic“ dargestellt. Es wäre sinnvoll eine Darstellung dieser Gebiete ebenfalls für den NAM anzubieten.
Zudem ist zu überlegen, das Bewegungsmodell zu verfeinern. Für bestimmte Praxisszenarien kann eine Gruppenbildung interessant sein. Möglicherweise kann dies mit dem Gebietsmodell verbunden werden. Die Kopplung des Verbindungsmodells an die Bewegung kann ebenfalls für bestimmte Szenarien interessant sein. Beispielsweise könnten die Verbindungen nicht mehr nur zwischen zufälligen Nodes stattfinden, sondern die Existenz von langsamen oder stehenden Servern sollte bedacht werden.

8 Resümee

Ein wichtiger Teil der Aufgabenstellung bestand darin, einen geeigneten Simulator für Ad-hoc-Netze auszuwählen. Obwohl verschiedene Simulatoren existieren, eignen sich jedoch nur wenige für die Ziele dieser Arbeit. Die Wahl fiel aufgrund der gewählten Kriterien auf den Network Simulator 2.
Ein Aspekt bei der Auswahl war die mögliche Einbindung weiterer Routingprotokolle. Formale Spezifikationssprachen wie SDL, Estelle oder LOTOS werden dabei von keinem Simulator unterstützt. Deshalb wird in dieser Arbeit die Verfügbarkeit des Quelltextes

als Erweiterbarkeit angesehen. Dies zeigt allerdings auch, dass derzeit kein Simulator existiert der für alle Problemstellungen maßgeschneiderte Lösungen liefert.
Als weiterer Teil der Aufgabenstellung sollte die Wegewahl grafisch dargestellt werden. Da der Network Animator hierfür nur bedingt geeignet war, wurde ein das Tool viewtraffic entwickelt, welches speziell auf die Anzeige der Routen optimiert wurde.
Zusätzlich sollte ein Bewegungsmodells in den NS-2 integriert werden. Zu diesem Zwecke wurde das Tool „gentraffic “entwickelt, welches auf Basis von Gebietsdefinitionen Bewegungsmuster erzeugt und diese dem NS-2 zur Verfügung stellt. Mittels dieser Gebietsdefinitionen sind verschiedenste Szenarien denkbar. Es ist ebenfalls möglich, Nodes anhand eines Markov-Modells oder gedächtnisbehaftet zu bewegen. Eine Überlagerung der Modelle ist vorgesehen.
Die Simulationen haben gezeigt, dass je nach Szenario verschiedene Routing-Protokolle gut oder schlecht abschneiden. Auffallend ist auch, dass zu bestimmten Zeitpunkten kein Routingverfahren eine Route aufrecht erhalten konnte. Dies zeigt, dass Quality of Service in Funknetzen bisher nur ansatzweise realisiert werden kann.
Da zur Zeit auf der Mailing-Liste des NS-2 über die DSR-Implementation diskutiert wird, sollten bei einer neuen DSR-Version die Szenarien noch einmal simuliert werden.

Literatur

[BMJ+98] Broch, Josh ; Maltz, David A. ; Johnson, David B. ; hu, Yih-Chun ; Jetcheva, Jorjeta: A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In: *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking(MobiCom 98)* (1998)

[Bor01] Bornträger, Christian: Hauptseminar Ad-hoc-Netzwerke. In: *http://ikmcip1.e-technik.tu-ilmenau.de/~ webkn/STUD-DIPLOM/STUDREFERAT/ad-hoc-netze/HS-SS01-AdHocNetze.html* (2001)

[BP] Bhirud, Abhi ; Parikh, Bhavin: Ad Hoc Network Simulator. In: *http://www2.cs.pitt.edu/~ melhem/Mobility/*

[DR] Dorai-Raj, Sundar: RandomNumberGenerator.java. In: *http://www.stat.vt.edu/~ sundar/java/code/RandomNumberGenerator.html*

[Eur98] European Telecommunications Standards Institute: Universal Mobile Telecommunications System (UMTS);Selection procedures for the choice of radio transmission technologies of the UMTS. In: *TR 101 112 V3.2.0* (1998)

[IBM] IBM: BlueHoc: Open-Source Bluetooth Simulator. In: *http://www-124.ibm.com/developerworks/opensource/bluehoc/*

[Jug01] Jugl, Enrico: *Mobilitätsmodellierung und Einflüsse auf Systemparameter von Mobilfunksystemen.* Shaker Verlag, 2001

[man] Mobile Ad-hoc Networks (manet). In: *http://www.ietf.org/html.charters/manet-charter.html*

[PC] Paspallis, N. ; Chatzigiannakis, J.: Ad-hoc Mobile Network Simulator. In: *http://www.cs.ucsb.edu/~ nearchos/adhoc.html*

[Ric] Rice University: Monarch Project. In: *http://www.monarch.cs.rice.edu/*

[Sca] Scalable Network Technologies, Inc.: QualNet. In: *http://www.qualnet.com/products/QualNet/*

[Sch01] Schopp, Michael: *Mobilität in Kommunikationsnetzen - Konzepte, Modellierung und Leistungsbewertung.* Institut für Nachrichtenvermittlung und Datenverarbeitung, Stuttgart, 2001

[UCL] UCLA, Computer Science D.: Global Mobile Information Systems Simulation Library. In: *http://pcl.cs.ucla.edu/projects/glomosim/*

[Unia] University of California, Berkeley: The Ptolemy Project. In: *http://ptolemy.eecs.berkeley.edu/*

[Unib] University of Maryland, Computer Science D.: Maryland Routing Simulator. In: *http://www.cs.umd.edu/projects/netcalliper/software.html*

[Unic] University of Southern California: The Network Simulator - ns-2. In: *http://www.isi.edu/nsnam/ns/*

[WH81] Wichmann, B. A. ; Hill, I. D.: Algorithm AS 183: An efficient and portable pseudo-random number generator. In: *Applied Statistics 31 188-190* (1981)

A Gebietsdatei Manhattan-Szenario

```
#Gebietsdefinition
#Typ, Geschwindigkeit, mittlere Bewegungsdauer, mittlere Pausendauer, Richtung,Streuung Richtung
#erste Zeit

#Häuser

0       5 5 1 0 0               0 0             200 0           200 200         0 200
0       5 5 1 0 0               230 0           430 0           430 200         230 200
0       5 5 1 0 0               460 0           660 0           660 200         460 200
0       5 5 1 0 0               690 0           890 0           890 200         690 200

0       5 5 1 0 0               0 230           200 230         200 430         0 430
0       5 5 1 0 0               230 230         430 230         430 430         230 430
0       5 5 1 0 0               460 230         660 230         660 430         460 430
0       5 5 1 0 0               690 230         890 230         890 430         690 430

0       5 5 1 0 0               0 460           200 460         200 660         0 660
0       5 5 1 0 0               230 460         430 460         430 660         230 660
0       5 5 1 0 0               460 460         660 460         660 660         460 660
0       5 5 1 0 0               690 460         890 460         890 660         690 660

0       5 5 1 0 0               0 690           200 690         200 890         0 890
0       5 5 1 0 0               230 690         430 690         430 890         230 890
0       5 5 1 0 0               460 690         660 690         660 890         460 890
0       5 5 1 0 0               690 690         890 690         890 890         690 890

#Strassen
#links rechts
1       3 1 0 0         0               0 200           890 200         890 230         0 230
1       3 1 0 0         0               0 430           890 430         890 460         0 460
1       3 1 0 0         0               0 660           890 660         890 690         0 690
1       3 1 0 0         0               0 890           890 890         890 920         0 920

#oben unten

1       3 1 0 1.57      0               200 0           230 0           230 920         200 920
1       3 1 0 1.57      0               430 0           460 0           460 920         430 920
1       3 1 0 1.57      0               660 0           690 0           690 920         660 920
1       3 1 0 1.57      0               890 0           920 0           920 920         890 920

#Kreuzung

2       3 1 0 0         0               200 200         230 200         230 230         200 230
2       3 1 0 0         0               430 200         460 200         460 230         430 230
2       3 1 0 0         0               660 200         690 200         690 230         660 230
2       3 1 0 0         0               890 200         920 200         920 230         890 230


2       3 1 0 0         0               200 430         230 430         230 460         200 460
2       3 1 0 0         0               430 430         460 430         460 460         430 460
2       3 1 0 0         0               660 430         690 430         690 460         660 460
2       3 1 0 0         0               890 430         920 430         920 460         890 460


2       3 1 0 0         0               200 660         230 660         230 690         200 690
2       3 1 0 0         0               430 660         460 660         460 690         430 690
2       3 1 0 0         0               660 660         690 660         690 690         660 690
2       3 1 0 0         0               890 660         920 660         920 690         890 690
```

```
2       3 1 0 0         0               200 890         230 890         230 920         200 920
2       3 1 0 0         0               430 890         460 890         460 920         430 920
2       3 1 0 0         0               660 890         690 890         690 920         660 920
2       3 1 0 0         0               890 890         920 890         920 920         890 920
```

B Matlab-Skripte

B.1 prepare.m

Die Funktion prepare liest die Werte der Simulation ein.

```
global AODV DSR DSDV TORA
xlabel "Zeit in Sekunden"
ylabel "Bytes/sec"
load tcp_bytes_send_AODV;
AODV=tcp_bytes_send_AODV;
load tcp_bytes_send_DSR;
DSR=tcp_bytes_send_DSR;
load tcp_bytes_send_DSDV;
DSDV=tcp_bytes_send_DSDV;
load tcp_bytes_send_TORA;
TORA=tcp_bytes_send_TORA;
```

B.2 writeps.m

Diese Funktion erstellt Grafiken der Datenraten als Postscript-Dateien.

```
function writeps
global AODV DSR DSDV TORA
gset terminal postscript


grid "on"
```

```
gset output "AODV.ps"
num=rows (AODV)-1;
title (strcat("AODV", sprintf(" Gesamtmenge: %8.0f KBytes",(sum(AODV)/1024))));
plot (0:1:num,AODV,";AODV;");


gset output "DSR.ps"
num=rows (DSR)-1;
title (strcat("DSR", sprintf(" Gesamtmenge: %8.0f KBytes",(sum(DSR)/1024))));
plot (0:1:num,DSR,";DSR;");


gset output "DSDV.ps"
num=rows (DSDV)-1;
title (strcat("DSDV", sprintf(" Gesamtmenge: %8.0f KBytes",(sum(DSDV)/1024))));
plot (0:1:num,DSDV,";DSDV;");

gset output "TORA.ps"
num=rows (TORA)-1;
title (strcat("TORA", sprintf(" Gesamtmenge: %8.0f KBytes",(sum(TORA)/1024))));
plot (0:1:num,TORA,";TORA;");

gset terminal x11
```

C Neue Routing-Protokolle

Es lassen sich neue Routing-Protokolle in den NS-2 integrieren. Allerdings müssen dazu einige Anpassungen an den Quelltexten vorgenommen werden.
Im Unterordner `tcl/mobility` befinden sich für die mobilen Routingverfahren tcl-Dateien, welche die Standardwerte des Protokolls festlegen. Zusätzlich werden hier die Prozeduren `create-<protokoll>-routing-agent` und

`<protokoll>-create-mobile-node` festgelegt. `<protokoll>` ist dabei durch den Protokollnamen zu ersetzen. Diese Datei sollte ebenfalls für ein eigenes Protokoll angelegt werden. Man kann hierfür auf eine bestehende Datei zurückgreifen und diese modifizieren.
Weiterhin muss die Datei `tcl/lib/ns-lib.tcl` angepasst werden. Hier muss die Zeile `source ../mobility/<protokoll>.tcl` hinzugefügt werden. Ähnliche Zeilen sind bereits vorhanden und können als Basis diesen. Zusätzlich muss die Prozedur `create-<protokoll>-agent` definiert werden. Hier können wieder vorhandene Prozeduren als Schablone dienen.
Sofern die Dateien nach dem vorhandenen Schema erweitert wurden, muss noch ein Agent mit dem Namen `Agent/<protokoll>` definiert werden. Dies geschieht über eine eine entsprechende C++-Klasse, welche von der Basisklasse TclClass abgeleitet wird. DSDV wird beispielsweise mittels

```
static class DSDVClass:public TclClass
{
  public:
  DSDVCLass():TclClass ("Agent/DSDV");
  .
  .
```

definiert. Eigene Erweiterungen sind entsprechend zu treffen. Für ein eigenes Protokoll ist ein möglichst ein eigener Unterordner zu nutzen. der Ordner `dsdv` kann als Schablone dienen.
Als letztes muss noch das Makefile des NS-2 um die entsprechenden Dateien erweitert werden. Eine Beispielimplementierung des fiktiven Technische Unversitäts-Protokoll (TUP) findet sich als Patch anbei. TUP entspricht dabei DSDV. Der Patch ist im ns-Unterordner, z.B. ns-2.1b9 mittels

```
patch -p1 < tup.patch
```

zu applizieren. Danach ist mittels configure und make der ns neu zu übersetzen.