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

Studienarbeit Nr. 2303

UMTS-Implementierung für OMNeT++

Thorsten Frosch

Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. Dr. h.c. Kurt Rothermel
Betreuer:	Dipl.-Inf. Damian Philipp
begonnen am:	18. Oktober 2010
beendet am:	19. April 2011
CR-Klassifikation:	C.2.5, D.2.10

Abstract

Um Anwendungen und Systeme simulieren zu können, die UMTS-Netze nutzen, wird in dieser Arbeit eine Simulation für ein UMTS-Kommunikationssystem entwickelt. Dabei beschränkt sich die Modellierung auf reines UMTS, ohne Erweiterungen wie HSDPA oder HSUPA. Auf die exakte physikalische Modellierung der Datenübertragung wird verzichtet, um die Simulation möglichst skalierbar zu gestalten. Basierend auf empirischen Daten wird ein Modell entwickelt, das zum einen die Funkschnittstelle von UMTS modelliert und dabei andere Komponenten wie das UMTS Core Network vereinfacht darstellt. Als Simulationsumgebung wird OMNeT++ verwendet, ein quelloffener und erweiterbarer Netzwerksimulator. Es wird der Entwurf des System beschrieben und es werden Implementierungsdetails dargestellt. Bei der Implementierung wurde auf Komponenten des INET/MANET-Frameworks für OMNeT++ zurückgegriffen, das die Funktionalität eines TCP/IP-Stacks zur Verfügung stellt. Das System wird anschließend evaluiert und mit Hilfe von Messergebnissen aus Testsimulationen mit empirischen Messdaten verglichen. Es wird außerdem ein Skalierbarkeitstest der Simulation durchgeführt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

Das Universal Mobile Telecommunication System (UMTS) ist ein Standard für Mobilfunknetze, der Anfang des Jahrhunderts eingeführt wurde und die dritte Generation der Mobilfunknetze darstellt. UMTS bietet eine deutlich höhere Datenrate als seine Vorgänger und ermöglicht neue mobile Anwendungen wie etwa Videotelefonie.

Zuvor hatte die Einführung des GSM-Standards (Global System for Mobile Communication, früher: Group Spéciale Mobile) Anfang der 1990er-Jahre die Nutzung des Mobilfunknetzes revolutioniert. Er löste die analogen Mobilfunkstandards der ersten Generation ab und wird als erster Standard der zweiten Generation ("2G") gezählt. 1993 nutzen bereits über eine Million Menschen GSM-Netze. Seitdem stiegen die Nutzerzahlen stark an, sodass inzwischen knapp zwei Milliarden Menschen in über 200 Ländern weltweit diesen Standard nutzen. Es ist ein Standard für voll-digitale Mobilfunksysteme, der sowohl paket- als auch leitungsvermittelnde Übertragung ermöglicht, jedoch hauptsächlich für die Telefonie und für den Short-Message-Service (SMS) genutzt wird.

Für die Datenübertragung stehen in einem GSM-Netz zwischen 9,6 kBit/s und 14,4 kBit/s zur Verfügung. Da sich durch die immer größere Nutzung des mobilen Internets und des steigenden Datenvolumens durch größere Dateien und Medieninhalte die Bandbreite zunehmend als limitierender Faktor erwies, wurden Weiterentwicklungen des GSM-Standards veröffentlicht, der dieses Problem beheben sollte. Es wurde unter anderem der GPRS- und der EDGE-Standard veröffentlicht, die Datenübertragungen mit 171 kBit/s beziehungsweise bis zu 220 kBit/s ermöglichten.

In Deutschland wurden im Jahr 2000 die UMTS-Lizenzen für insgesamt über 50 Milliarden Euro versteigert und an verschiedene Mobilfunkanbieter vergeben. Durch diesen neuen Standard stieg die Datenrate wieder an: Nun waren im Downstream maximal 384 kBit/s möglich, im Upstream immerhin noch 64 kBit/s. Doch erst die Einführung von High Speed Downlink Packet Access (HSDPA) und High Speed Uplink Packet Access (HSUPA) ermöglichte es den Nutzern erstmals Daten im MBit-Bereich zu übertragen: Ein Download wurde mit bis zu 14,4 MBit/s möglich, ein Upload mit bis zu 5,8 MBit/s. Dadurch ergaben sich auch völlig neue Nutzungsbereiche des mobilen Internets, da neben dem normalen Surfen auch Anwendungen wie das Konsumieren hochauflösender Videoinhalte oder Videotelefonie ermöglicht wurde.

Um Anwendungen und Systeme simulieren zu können, die UMTS-Netze nutzen, wurde in dieser Arbeit nun eine Simulation für ein UMTS-Kommunikationssystem entwickelt. Dabei beschränkt sich die Modellierung auf reines UMTS, ohne Erweiterungen wie HSDPA oder HSUPA. Auf die exakte physikalische Modellierung der Datenübertragung wird verzichtet, um die Simulation möglichst skalierbar zu gestalten. Basierend auf den empirischen

Daten, die in [CGGPC06] und [TLL08] über die Verzögerung und Bandbreitenverteilung in UMTS-Systemen gesammelt wurden, ist ein Modell entwickelt worden, das zum einen die Funkschnittstelle von UMTS modelliert und dabei andere Komponenten wie das UMTS Core Network oder auch das Internet vereinfacht darstellt.

Als Simulationsumgebung wurde OMNeT++ [omn] verwendet, ein erweiterbarer und modularer Netzwerksimulator, der auf der Programmiersprache C++ basiert. In dieser Arbeit wurde zudem die INET/MANET-Erweiterung verwendet, die viele Funktionen und Mechanismen für eine IP-basierte Netzwerksimulation zur Verfügung stellt. So ist es mit diesem Framework möglich, auf bereits implementierte Protokolle des TCP/IP-Stacks zuzugreifen. Außerdem sind verschiedene Mobilitätsmodelle vorhanden, nach denen sich mobile Geräte auf der Simulationsfläche bewegen. Dies ist für diese Arbeit sehr nützlich, da Geräte mit UMTS-Funktionalität zumeist mobil sind und diese Mobilität auch in dieser Simulation berücksichtigt werden soll.

Abschließend wird das entwickelte System auf seine Korrektheit und Skalierbarkeit untersucht. Um die entwickelte Simulation zu verifizieren, werden Ergebnisse aus verschiedenen Simulationsszenarien mit den empirischen Messdaten aus den Arbeiten [CGGPC06] und [TLL08] verglichen. Zudem wurden selbst Messreihen mit einem UMTS-fähigen Smartphone durchgeführt, die dann ebenfalls mit den Simulationsergebnissen verglichen wurden. Außerdem dienen sie zu einer weiteren Verifizierung der in den als Grundlage verwendeten empirischen Daten der beiden erwähnten Arbeiten. Die Skalierbarkeit der Simulation wird anhand der Laufzeit verschiedener Szenarien getestet.

1.1 Gliederung

Die Arbeit gliedert sich in mehrere Abschnitte: In Kapitel 2 werden die Grundlagen von UMTS-Systemen erläutert und die in [CGGPC06] und [TLL08] vorgeschlagenen Modelle zur Modellierung von Verzögerung und Bandbreite vorgestellt. Im nächsten Kapitel wird das zu implementierende System vorgestellt. Dabei wird hier vorrangig auf den Entwurf und den Aufbau eingegangen. Außerdem werden dort die verwendeten Algorithmen dargestellt. In Kapitel 4 wird die Implementierung in OMNeT++ beschrieben. Abschließend wird in Kapitel 5 eine Evaluation des entwickelten Systems durchgeführt, indem Ergebnisse aus verschiedenen Simulationsszenarien mit den vorhandenen empirischen Messdaten verglichen werden. Zudem sind hier die Ergebnisse des Skalierbarkeitstests zu sehen. Abschließend erfolgt in Kapitel 6 eine Zusammenfassung der Arbeit und es wird ein Ausblick gegeben.

2 Grundlagen

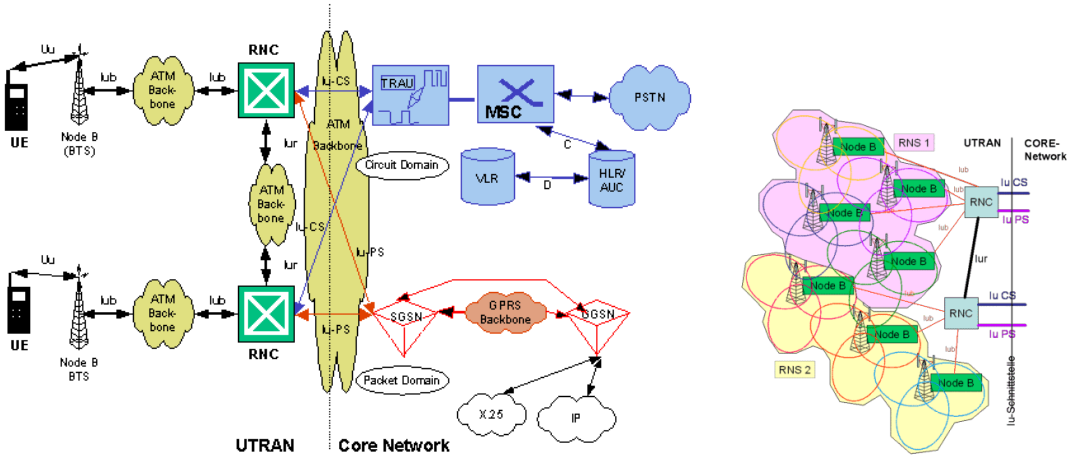
In diesem Kapitel werden die Grundlagen von UMTS-Kommunikationssystemen erläutert. Zudem wird eine Übersicht über zwei Arbeiten gegeben, die die Eigenschaften eines solchen Systems wie Verzögerungsverhalten einzelner Pakete oder die Bandbreitenverteilung auf die angeschlossenen Geräte empirisch untersucht haben.

2.1 UMTS

Das Universal Mobile Telecommunication System (UMTS) ist ein Standard der dritten Generation der Mobilfunknetze. Durch seine Einführung konnten deutlich höhere Datenraten als noch mit GSM oder seinen Erweiterungen wie EDGE erzielt werden.

Im Gegensatz zu GSM, das auf Frequency Division Multiplexing (FDM) und Time Division Multiplexing (TDM) basiert, verwendet UMTS die Technik des Wideband Code Division Multiple Access (W-CDMA). Diese nutzt als Unterteilung der einzelnen Signale einen binären Code, anhand dessen die überlagerten Signale wieder getrennt und gefiltert werden können. Uplink und Downlink sind dabei wiederum getrennt, d.h. die Uplinksignale liegen in einem anderen Frequenzbereich als die des Downlinks. Dieses Verfahren wird Frequency Division Duplex (FDD) genannt. Es existiert ein alternatives Verfahren, Time Division Duplex (TDD), das die Unterteilung von Up- und Downlink per Zuweisung von Zeitslots realisiert. In der Praxis kommt jedoch nur FDD zum Einsatz. Um den parallelen Betrieb von mehreren UMTS-Netzen durch unterschiedliche Mobilfunkbetreiber zu ermöglichen, wird das UMTS-Frequenzband weiter unterteilt und die Teilfrequenz den unterschiedlichen Providern zugewiesen. So existieren zum Beispiel in Deutschland sechs FDD-Frequenzbänder, die auf die vier Provider Deutsche Telekom, Vodafone, O2 und E-Plus verteilt werden.

Die Architektur von UMTS [3GPa] ähnelt der des GSM-Systems, um eine Kompatibilität der zwei Standards zu gewährleisten und den Umstieg für die Provider zu erleichtern und kostengünstiger zu gestalten. In Abbildung 2.1(a) ist eine Übersicht der Architektur zu sehen. Man kann sie in zwei Bereiche unterteilen: das Core Network und das UMTS Terrestrial Radio Access Network (UTRAN). Das Core Network entspricht dem von GSM und GPRS und verarbeitet paket- und leitungsvermittelnde Dienste getrennt. Das UTRAN besteht im Wesentlichen aus den Radio Network Controllern (RNC) und den mit ihnen verbundenen Node B. Alle an ein RNC angeschlossenen Node B samt deren Funkzellen bezeichnet man als Radio Network Subsystem (RNS). Ein Beispiel dafür ist in Abbildung 2.1(b) zu sehen. Die Node B sind die Funktürme, die meist je drei Zellen versorgen. Die RNC sind für die Verwaltung der Funkressourcen der angeschlossenen Zellen zuständig. Die Node B dagegen



(a) Gesamtarchitektur eines UMTS-Systems (b) Übersicht des UTRAN

Abbildung 2.1: Architektur eines UMTS-Systems (Quelle: [umt])

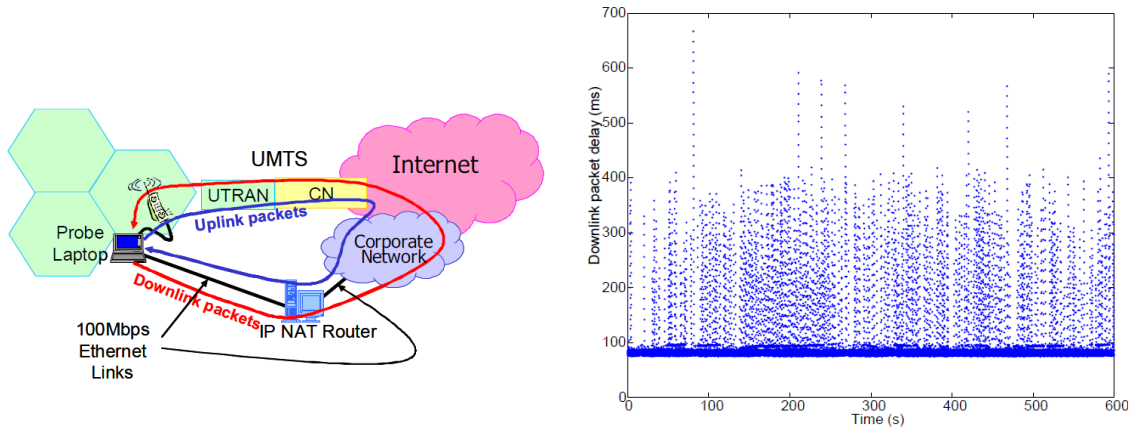
sind für die physikalische Übertragung zuständig. Die RNC sind mit den Node B und dem Core Network über ATM-Verbindungen verbunden.

Die Protokollarchitektur der Funkschnittstelle ist dem OSI-Modell entsprechend in Schichten eingeteilt und besteht aus dem Physical Layer, dem Data Link Layer und dem Network Layer [3GPb]. Der Link Layer kann wiederum in die Medium Access Control (MAC), die Radio Link Control (RLC) und den Packet Data Convergence Protokoll unterteilt werden. Die Teilschicht des RLC ist für die Fragmentierung und Defragmentierung der Pakete aus den oberen Schichten zuständig. Darüber hinaus bietet er den darüber liegenden Schichten zwei Arten der Übertragung an: zum einen die unbestätigte Übertragung und zum anderen die bestätigte Übertragung. Zweitere nutzt ein Sliding-Window-Protokoll mit Selective-Repeat-Mechanismus zur Realisierung seiner Dienste. Da die Funkübertragung aufgrund von Interferenzen und anderen Störungen extrem anfällig ist für Bitfehler und hohe Bitfehlerraten auftreten können, sind hier viele Retransmits zu erwarten. Diese sind durch den RLC zwar transparent für die oberen Schichten, allerdings wirken sie sich auf die Übertragungszeiten der einzelnen Pakete aus.

Die Übertragungen auf dem Physical Layer sind getaktet. Sie erfolgt über verschiedenen Transportkanäle, denen ein bestimmtes Time Transmission Interval (TTI) zugewiesen wurde. In diesem Intervall werden dann die Daten über die Funkschnittstelle übertragen. Die Taktung der Übertragung bietet unter anderem die Möglichkeit zur Fehlerkontrolle.

2.2 Empirische Daten

In diesem Abschnitt werden zwei Arbeiten vorgestellt, die empirische Daten zum Verzögerungsverhalten und der Bandbreitenzuteilung in UMTS-Kommunikationssystemen aufgezeichnet haben.



(a) Versuchsaufbau (b) Downlink-Verzögerungen

Abbildung 2.2: Versuchsaufbau und Resultat der Downlink-Messung (Quelle: [CGGPC06])

2.2.1 Verzögerungsmodell

In der Arbeit von Cano-Garcia et al. [CGGPC06] wurde das Verzögerungsverhalten eines UMTS-Systems untersucht. Dazu wurden empirische Messungen durchgeführt und anschließend ein Modell entwickelt, das das Verhalten beschreiben soll.

Der Versuchsaufbau ist in Abbildung 2.2(a) zu sehen. Als Zugang zum UMTS-Netz wurde ein Sony Ericsson V800 genutzt, das an einen Laptop per USB angeschlossen wurde. Dieser besaß außerdem eine Ethernetschnittstelle, die mit dem Netzwerk einer Universität verbunden war.

Es wurden dann UDP-Pakete mit einer Größe von 40 Byte von einem Interface über das UMTS-Netz zum anderen Interface gesendet, wobei der Abstand zwischen den Paketen 20 ms betrug. Der resultierende Datenstrom mit 16 kBit/s lag dabei deutlich unter der verfügbaren Bandbreite des Uplinks als auch des Downlinks des UMTS-Zugangs. Das Problem der Zeitsynchronisation wurde vermieden, da sich das Quell- und das Zielinterface auf dem selben Gerät befanden.

Diese Messung wurde an drei verschiedenen Tagen nachts für jeweils zehn Stunden durchgeführt. Ein Ausschnitt von 600 s der Downlink-Messung ist in 2.2(b) zu sehen. Auffällig dabei ist, dass die meisten Pakete eine Verzögerung von etwa 80 ms erfahren, jedoch auch sehr viele Verzögerungsspitzen zu beobachten sind, wobei der Abstand zwischen den einzelnen Paketen in diesen Spitzen konstant 20 ms beträgt. Dies ist genau der Abstand, mit dem die Pakete versendet wurden. Die ähnlichen Beobachtungen wurden ebenso beim Uplink-Verzögerungsverhalten gemacht.

Die Verzögerungsspitzen haben ihren Ursprung im RLC Sublayer, der dafür zuständig ist, die Pakete zuverlässig und in richtiger Reihenfolge zu übertragen. Wenn ein Paket fehlerhaft übertragen wurde, so wird die Übertragung wiederholt. Alle weiteren über die

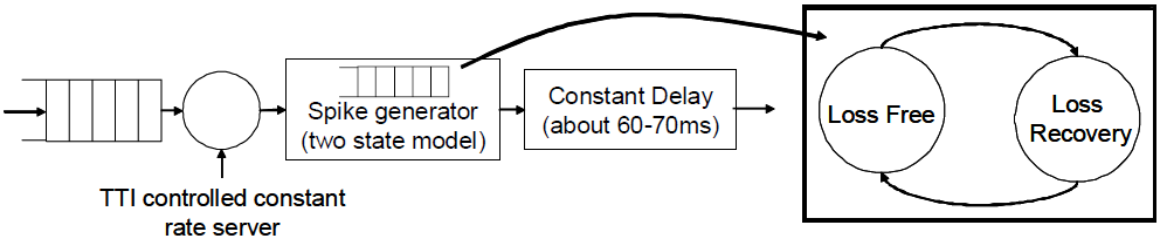


Abbildung 2.3: Vorgeschlagenes Verzögerungsmodell von Cano-Garcia et al. (Quelle: [CGGPC06])

Funkschnittstelle empfangene Pakete werden gepuffert und nicht an die höher liegenden Schichten weitergegeben, um die richtige Sequenz der Pakete zu bewahren. Wenn das fehlerhafte Paket schließlich korrekt übertragen wurde, wird es samt aller gepuffteren Pakete gleichzeitig weitergegeben. Das bedeutet, dass die Pakete dann gleichzeitig in den höheren Layern ankommen und ihre Verzögerung sich somit nur um den Sendeabstand zweier Pakete unterscheiden. Die Dauer der Verzögerungsspitzen ist dabei diskret, da die Übertragung bei UMTS in TTI getaktet ist und der RLC Layer für das Beheben eines Paketverlustes eine ganzzahlige Anzahl an TTI benötigt. Die Verteilung der Dauer einer Spitze wurde in [CGGPC06] über eine Wahrscheinlichkeitsfunktion definiert, ebenso wie die Zeit zwischen zwei Verzögerungsspitzen. Außerdem wurde festgestellt, dass kein Zusammenhang zwischen dem Auftreten eine Verzögerungsspitze im Uplink und Downlink besteht.

Basierend auf diesen Ergebnissen wurde ein Modell für die Paketverzögerung in UMTS-System erstellt, das in Abbildung 2.3 zu sehen ist. Alle zu sendenden Pakete passieren zunächst einen Begrenzer (im Original "TTI controlled constant rate server", im Folgenden "TTI Server"genannt), der jedes TTI eine von der verfügbaren Bandbreite abhängige Datenmenge sendet. Sobald alle Daten eines Pakets übertragen wurden, wird es zur nächsten Stufe weitergeleitet. Dies ist ein Zustandsautomat, der für die Generierung der Verzögerungsspitzen zuständig ist. Im ersten der beiden Zustände, dem Loss Free State, werden alle ankommenden Pakete ohne weitere Verzögerung weitergeleitet. Ab dem Wechsel in den Recovery State werden die Pakete in einem Puffer gespeichert und bei einem erneuten Zustandswechsel in den Loss Free State alle gleichzeitig an die nächste Stufe weitergeleitet. Die letzte Komponente des Modells ist die konstante Verzögerung, das alle Pakete um 60 ms bis 70 ms verzögern soll.

Das vorgeschlagene Modell soll getrennt für den Uplink und den Downlink verwendet werden. Allerdings wird nicht erwähnt, ob in einer Simulation jedes Mobilgerät ein solches Modell besitzen soll oder ob für alle Geräte ein gemeinsames Modell vorgesehen ist. Wie sich später herausstellt, wird das Verzögerungsverhalten entsprechend den Messwerten simuliert, wenn jedes Mobilgerät über eine Instanz des beschriebenes Modells verfügt.

2.2.2 Bandbreitenmessungen

In der Arbeit von Tan et al. [TLL08] wurde die Übertragungskapazität und von UMTS-Systemen untersucht. Der Versuchsaufbau sah vor, TCP-Verbindungen von mehreren UMTS-fähigen Mobiltelefonen zu stationären Servern im Internet aufzubauen und Daten zu übertragen. Es wurde dabei der Durchsatz der einzelnen Geräte, der Gesamtdurchsatz und die Round Trip Time (RTT) per Ping-Anfragen gemessen. Darüber hinaus wurden auch Untersuchungen über das Verhalten des UMTS-Systems bei Video- und Telefonverbindungen durchgeführt. Da sich diese Arbeit auf die Übertragung von IP-Datenpaketen beschränkt, wird in dieser Arbeit nicht näher darauf eingegangen. Die Messungen wurden bei drei verschiedenen Mobilfunkprovidern durchgeführt; dabei wurde auf gute Empfangsqualität der UMTS-Geräte geachtet.

Ergebnis dieser Messreihe war, dass die Gesamtbandbreite, die ein Node B pro Zelle zu vergeben hat, im Downlink zwischen 850 kBit/s und 1 MBit/s liegt, beim Uplink zwischen 750 kBit/s und 900 kBit/s. Der erreichbare Durchsatz pro UMTS-Gerät war vom Provider abhängig. So war beim ersten Provider die maximale mit UMTS mögliche Bandbreite verfügbar, solange das System nicht überlastet war. Andernfalls sank die Bandbreite pro Gerät gleichmäßig. Bei Provider 2 und 3 war die Bandbreite pro Gerät auf einen gewissen Wert begrenzt, obwohl der maximale Gesamtdurchsatz nicht erreicht wurde. Ebenso war dieses Verhalten bei den RTT zu beobachten: bei Provider 1 schwankten die Werte stark zwischen 200 ms und mehreren Sekunden, bei Provider 2 und 3 hatten sie diskrete Stufen von einer Sekunde und drei Sekunden. Allerdings war auch zu sehen, dass bei Provider 2 und 3 einzelnen Geräten die maximale Bandbreite zur Verfügung stand, was wie die diskreten RTT-Stufen auf eine bestimmte Verteilungsstrategie schließen lässt, die allerdings nicht näher untersucht wurde.

2.3 OMNeT++

OMNeT++ ist ein quelloffener Netzwerksimulator, der auf der Programmiersprache C++ basiert. Als Grundlage der grafischen Entwicklungsumgebung dient die weit verbreitete Eclipse IDE.

OMNeT++ unterstützt die modulare Entwicklung von Simulationskomponenten. So ist jede Simulation aus verschiedenen Modulen zusammengesetzt, die wiederum aus Teilmodulen bestehen können. Module werden per Network Description Language beschrieben. Jedem einfachen Modul gehört eine C++-typische Headerdatei (Dateiendung *.h) und eine Implementierungsdatei (*.cc) an, in denen die Protokolle und Algorithmen realisiert werden. Einfache Module können zu einem komplexem Modul verbunden werden. Die einzelnen Module können mit Links und Kanälen verbunden werden, die Eigenschaften wie Bandbreite und Verzögerung haben können. Am Ende jedes Links muss ein Gate stehen, das im Modul definiert wird und somit den Endpunkt eines Übertragungskanals darstellt. Dabei wird zwischen In-Gates, die nur Nachrichten empfangen, Out-Gates, die nur Nachrichten senden und Inout-Gates, die sowohl senden als auch empfangen können, unterschieden. Alternativ

zu der Nutzung von Übertragungskanälen besteht auch die Möglichkeit, Nachrichten direkt von einem Gate zum anderen zu übertragen, ohne dass eine Verbindung besteht, was zum Beispiel für die Modellierung einer Funkübertragung genutzt werden kann.

Die Simulationsparameter werden über eine Konfigurationsdatei eingestellt. Darin können Parameter bestimmt werden, die Module als Nutzereingabe erwarten. Wird die Simulation dann gestartet, so läuft diese in diskreten Ereignisschritten ab. Ein Ereignis der Simulation ist etwa das Versenden einer Nachricht oder der Ablauf eines Timers. Alle Ereignisse finden zu einem bestimmten Simulationszeitpunkt statt. Die Simulationsengine berechnet somit alle durch ein Ereignis ausgelöste Änderungen der Simulation und springt anschließend zum nächsten Ereignis. Das bedeutet, dass sich die Simulationszeit von der realen Zeit deutlich unterscheiden kann.

2.3.1 Mobilität

Die Mobilgeräte der in dieser Arbeit erstellten Simulation können sich auf Simulationsgrundfläche bewegen. Im INET/MANET-Framework für OMNeT++ sind bereits verschiedene Bewegungsmuster implementiert, die dann auch in der Simulation genutzt werden können. In der Evaluation der Simulation werden drei dieser Mobilitätsmuster betrachtet: die Null-Mobilität, die Random-Waypoint-Mobilität und die NS2-Trace-File-Mobilität.

Die Null-Mobilität ist kein echtes Bewegungsmuster, sondern mehr eine Dummy-Implementierung. Mit ihr wird das Mobilgerät nicht bewegt, sondern verharrt auf dem Startpunkt, der zu Simulationsbeginn angegeben wurde. Alternativ kann man auch zufällig einen Punkt auf der Simulationsfläche wählen lassen.

Das Bewegungsmuster der Random-Waypoint-Mobility beruht auf der Verbindung von Punkten auf der Simulationsgrundfläche, die abgelaufen werden. Von einer zufälligen Startposition ausgehend wird ein nächster zufällig gewählter Punkt auf der Simulationsgrundfläche bestimmt. Danach beginnt das Mobilgerät, sich mit linearer Geschwindigkeit zu eben gewählten Punkt zu bewegen. Dort angekommen, wird für eine bestimmte Zeit an diesem Ort gewartet, bis daraufhin der nächste zufällige Punkt bestimmt wird, der angelaufen werden soll.

Das dritte behandelte Bewegungsmuster ist die NS2-Trace-File-Mobilität. Sie ähnelt stark der Random-Waypoint-Mobility. Auch hier werden nacheinander Orte auf der Simulationsfläche mit linearer Geschwindigkeit angelaufen. Der Unterschied besteht darin, dass diese Punkte aus einer Datei eingelesen werden, zusammen mit dem Zeitpunkt, zu dem der Ort erreicht werden soll. Aus den Zeitangaben wird dann jeweils die Geschwindigkeit berechnet, mit der sich das Mobilgerät bewegen muss, um die Zeitvorgaben einzuhalten. Die Datei muss dabei im NS2-Trace-File-Format vorliegen, einem weit verbreitetem Format für Mobilitätsmodellierung. Mit dieser Art der Bewegungsmodellierung ist es zum Beispiel möglich, die Mobilgeräte anhand von Waypoints zu bewegen, die auf Grundlage einer Straßennetzkarte erstellt werden. Diese Art der Modellierung ist deutlich realitätsnaher als eine reine zufällige Wahl der Waypoints.

2.4 Verwandte Arbeiten

Neben den bereits betrachteten Arbeiten, die sich mit dem Verzögerungsverhalten und Bandbreiteneigenschaften von UMTS-Systemen beschäftigen, gibt es eine Reihe weiterer Arbeiten, die die Verzögerungseigenschaften untersuchen [LK05, BCCF01]. Allerdings wird dort vermehrt auf die physikalischen Eigenschaften der Übertragung eingegangen, wovon in dieser Arbeit abstrahiert werden soll. Daneben beschäftigen sich [AS08] und [CCE03] mit der Performance von TCP in UMTS-Systemen, wogegen in dieser Arbeit die allgemeine Leistungsfähigkeit von UMTS modelliert werden soll und nicht die von TCP. In der Arbeit von Marquez-Barja et al. [MBCCM10] wird die Performance von UMTS und anderen Funkübertragungssystemen im Netzwerksimulator ns-2 getestet. In dieser Arbeit wird dagegen das Verzögerungsverhalten und die Bandbreitenverteilung simuliert.

3 Systementwurf

In diesem Kapitel wird der Entwurf eines Modells eines UMTS-Kommunikationssystems für die Simulation beschrieben. Die Modellierung beschränkt sich auf die Funkschnittstelle des UMTS-Systems, Komponenten wie zum Beispiel das Core Network werden nicht modelliert. Zudem wird bewusst auf eine physikalisch exakte Modellierung verzichtet, um eine höhere Skalierbarkeit der Simulation zu erreichen. Im Folgenden werden die einzelnen Komponenten der Simulation im Hinblick auf ihre Entwurfskriterien dargestellt. Die Implementierung des Systems in OMNeT++ dagegen wird in Kapitel 4 beschrieben. Das Modell zur Modellierung der Verzögerung der Datenpakete wurde aus der Arbeit [CGGPC06] übernommen. Die Zuweisung der Bandbreite dagegen wurde aus den empirischen Ergebnissen in [TLL08] nachgebildet.

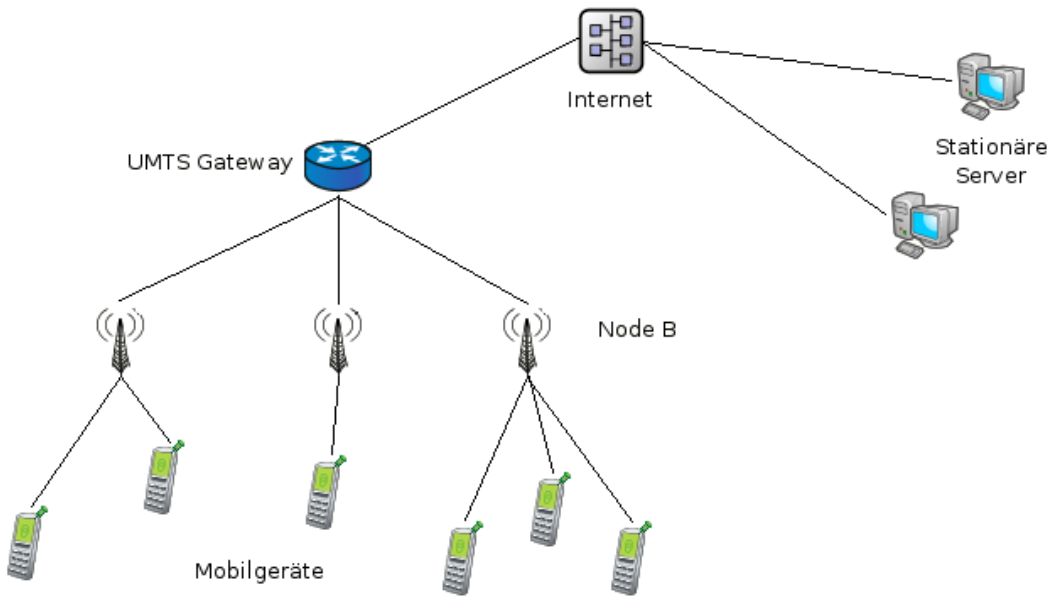


Abbildung 3.1: Gesamtaufbau des Systems

3.1 Aufbau

Das UMTS-Kommunikationssystem setzt sich aus mehreren Komponenten zusammen. Eine Übersicht ist in Abbildung 3.1 zu sehen. Die Mobilfunkgeräte und die Node B befinden sich auf einem Simulationsgebiet, dessen Größe vom Nutzer bestimmt werden kann. Das UMTS-Netz ist über den UMTS Gateway mit dem Internet verbunden. Der Gateway vereint alle RNC eines realen UMTS-Systems, da diese aus Nutzersicht transparent sind und deshalb auf eine genaue Modellierung verzichtet werden kann.

Das UMTS-Netz besteht aus mehreren Node B, zu denen sich mobile UMTS-fähige Geräte verbinden können. Im Gegensatz zu einem realen UMTS-System verwaltet jeder Node B hier nur eine Funkzelle und nicht mehrere, weshalb die Reichweite vereinfacht mit einem Kreis beschrieben werden kann. Die Anzahl der im System vorhandenen Node B hängt von der gewählten Platzierungsstrategie ab. Die Anzahl der mobilen Geräte lässt sich dagegen frei wählen. Alle Mobilgeräte besitzen bis auf den Physical Layer einen vollständigen TCP/IP-Stack. Die Datenübertragung soll dann zwischen Mobilfunkgeräten und kabelgebundenen Servern als auch ein zwischen zwei Mobilgeräten möglich sein.

Jedes Mobilgerät besitzt einen Association Client, der für die Verwaltung der Verbindung zum Node B zuständig ist. In jedem Node B ist ein Association Server vorhanden, der das Gegenstück zum Association Client darstellt und für das Verbindungsmanagement der zum Node B verbundenen Mobilgeräte zuständig ist. Der Bandwidth Assigner ist ebenfalls Teil eines jeden Node B und verteilt die verfügbare Bandbreite auf die verbundenen mobilen Geräte.

Die Simulation unterstützt die gleichzeitige Verwendung mehrerer UMTS-Provider, was bedeutet, dass sich auf der Simulationsfläche Node B befinden, die jeweils einem dieser Provider zugeordnet sind. Auch die Mobilgeräte besitzen einen Stammprovider, zu dessen Funkmasten sie sich bevorzugt verbinden werden. Sollte kein Node B des eigenen Providers verfügbar sein, so versucht sich das Mobilgerät auch zu Node B anderer Provider zu verbinden. Dieser Ansatz kann für weitere Anwendungen wie die Berechnung von Roamingkosten verwendet werden.

3.2 Mobile Geräte

Mobile Geräte besitzen die Möglichkeit, über ein UMTS-Interface eine Verbindung zum Internet herzustellen. Einem Mobilgerät können neben dem UMTS-Interface noch weitere Interfaces hinzugefügt werden, um zum Beispiel zusätzlich einen WLAN-Zugang herzustellen. Jedem Mobilgerät wird zu Beginn der Simulation genau eine IP-Adresse zugeordnet, die sich alle dort vorhandenen Interfaces teilen und die auch im Lauf der Simulation nicht mehr verändert wird. Die Ebene der Interfaces entspricht dem Link Layer eines TCP/IP-Stacks. Über den Interfaces sind die restlichen Layer des Stacks implementiert. Auf die Modellierung des Physical Layers verzichtet. So ist es also möglich, auf einem mobilen Endgerät eine oder mehrere UDP- und TCP-Anwendungen auszuführen, um damit beispielsweise einen

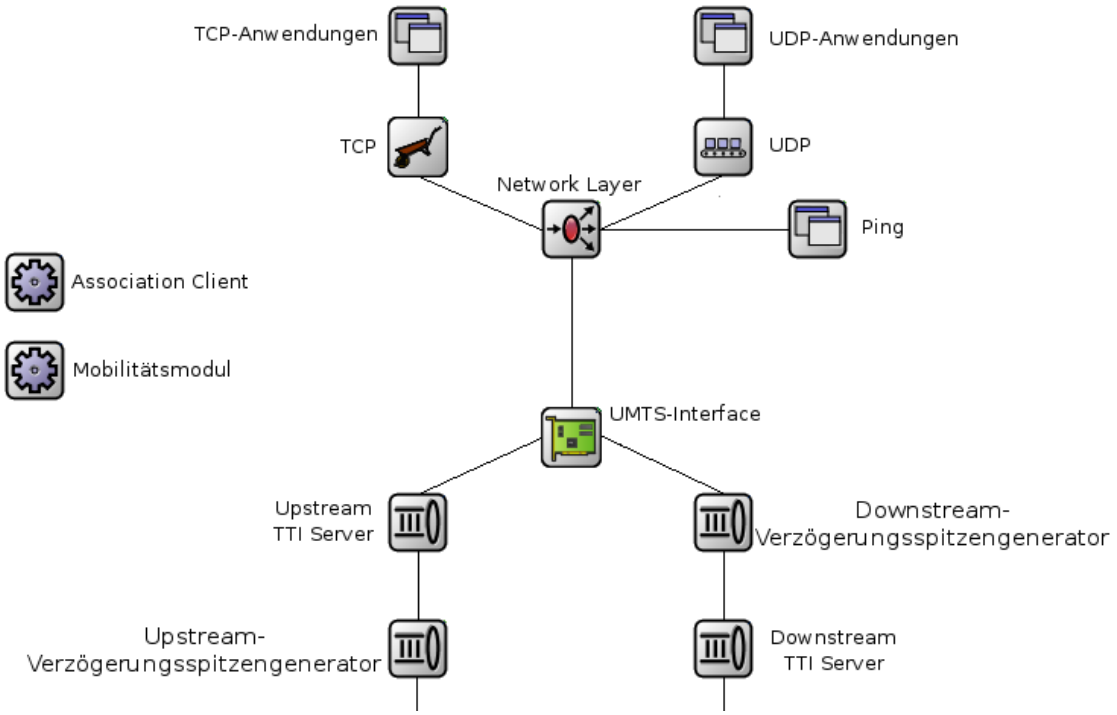


Abbildung 3.2: Aufbau eines mobilen Geräts

Browser mit HTTP-Datenverkehr zu simulieren. Es können beliebige eigene Anwendungen entwickelt werden, jedoch wird in dieser Arbeit nur auf eine HTTP-basierte Anwendung eingegangen. Neben den TCP- und UDP-Anwendungen besteht noch die Möglichkeit, über eine Ping-Anwendung die Round Trip Time (RTT) zu einem anderen Host zu messen.

Alle vom UMTS-Interface abgehenden Pakete passieren zunächst den Upstream TTI Server, der die erste Stufe des in [CGGPC06] vorgeschlagenen Verzögerungsmodells darstellt. Er ist dafür zuständig, die Übertragungszeit jedes einzelnen an den momentan verbundenen Node B zu sendendes Pakets zu modellieren. Nachdem die Pakete den TTI Server verlassen haben, passieren sie den Upstream-Verzögerungsspitzengenerator, der die in [CGGPC06] beschriebenen Verzögerungsspitzen generiert. Die Funktionsweisen des TTI Servers und des Verzögerungsspitzengenerators wurden bereits im Abschnitt 2.2.1 beschrieben. Nach dem Passieren des TTI Servers und des Verzögerungsspitzengenerators verlassen die Pakete das Mobilgerät und werden an den Node B gesendet. Analog zu der beschriebenen Funktionsweise in Upstream-Richtung müssen die eingehenden Downstream-Pakete ebenfalls einen TTI Server und einen Verzögerungsspitzengenerator passieren, die jedoch getrennt von denen des Upstreams sind.

Alle Komponenten des Mobilgeräts, die wie in Abbildung 3.2 zu sehen miteinander verbunden sind, sind über ideale Übertragungskanälen verbunden, die unbegrenzte Bandbreite und keine Verzögerung besitzen.

Zusätzlich zur Implementierung der einzelnen TCP/IP-Layer besitzt jedes mobile Gerät eine Mobilitätskomponente, die die Bewegung und somit die aktuelle Position auf der Simulationsfläche bestimmt. Es ist vorgesehen, dass eines der im INET/MANET-Framework vorhandenen Mobilitätsmuster verwendet wird. Des weiteren benötigt jedes Mobilgerät einen Association Client, der für den Aufbau, Erhaltung und Abbau einer Verbindung zu der sich in der Umgebung befindlicher Node B zuständig ist. Dessen Funktionsweise wird im Abschnitt 3.2.1 beschrieben.

3.2.1 Association Client

Der Association Client ist ebenfalls Bestandteil jedes mobilen Endgeräts und ist zuständig für das Verwalten der Verbindung zum Node B. Zu Beginn der Simulation sucht der Association-Client nach allen erreichbaren Node B in der Umgebung. Danach wird versucht, eine Verbindung zum geografisch nächstgelegenen Node B aufzubauen. Sollte keine Verbindung möglich sein, weil aufgrund fehlender Bandbreitenkapazitäten die Verbindungsanfrage abgelehnt wurde, werden die anderen erreichbaren Node B mit aufsteigender Entfernung abgefragt. Sollte nach einem Verbindungsversuch zu allen Node B in Reichweite keine Verbindung möglich sein, so wird ein Timer gestartet, nach dessen Ablauf erneut versucht wird, sich zu einem Node B zu verbinden, da möglicherweise Kapazitäten der eben abgefragten Node B frei geworden sind oder da sich das Mobilgerät nun näher an einem anderen Node B befindet.

Jeder Node B deckt einen kreisförmigen Bereich ab, innerhalb dessen sich ein Mobilgerät verbinden kann. Die Optimierungsgrenze ist durch einen Bruchteil der maximalen Reichweite gegeben und stellt einen Kreis um den Node B dar. Bewegt sich nun ein mobiles Gerät mit bestehender Verbindung in den Bereich zwischen Optimierungsgrenze und maximaler Reichweite, sucht der Association Client nach einem näher gelegenen Node B und versucht sich zu verbinden. Sollte tatsächlich ein geografisch näherer Node B gefunden werden, so wird ein Handover initiiert und die Verbindung zum alten Node B abgebaut und zum neuen Node B aufgebaut. Entfernt sich jedoch das mobile Gerät über die maximale Reichweite hinaus, ohne dass eine Verbindung zu einem anderen Node B aufgebaut werden konnte, so wird trotzdem die Verbindung zum bisherigen Node B abgebaut und danach ein Timer gesetzt, nach dessen Ablauf wieder nach neuen Node B in der Umgebung gesucht wird.

Die Reihenfolge, in der die Node B angefragt werden, hängt nicht nur vom geografischen Abstand ab, sondern auch vom Provider des Mobilgeräts und des betreffenden Node B. Zuerst werden die Node B des eigenen Providers abgefragt. Erst wenn keine Verbindung zu einem Funkmasten des eigenen Providers möglich ist, werden die Node B aller anderen Provider in Betracht gezogen. Allerdings findet kein Handover von einem Node B des eigenen Node B zu einem eines anderen Providers statt, solange das mobile Gerät die maximale Reichweite nicht verlassen hat.

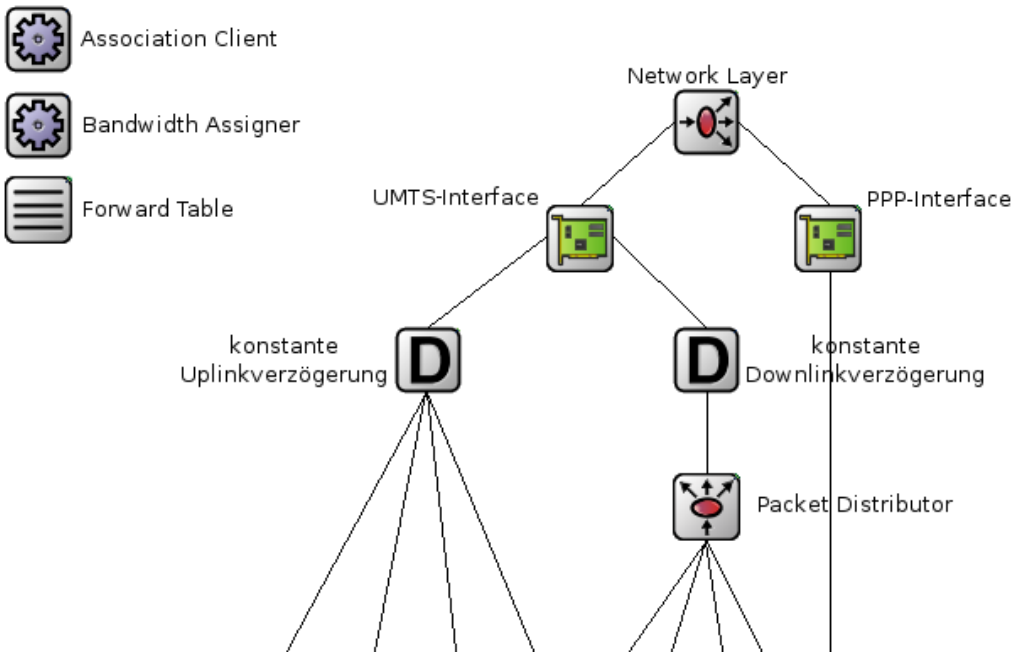


Abbildung 3.3: Aufbau eines Node B

3.3 Node B

Die verschiedenen Node B stellen den Zugangspunkt für die mobilen Endgeräte zum UMTS-Netz und damit zum Internet dar. Alle Node B sind direkt mit dem UMTS-Gateway verbunden. An diesen werden alle Pakete weitergeleitet, die nicht an lokal verbundene Mobilgeräte adressiert sind. Jeder Node B ist aus einzelnen Unterkomponenten aufgebaut: Der Association Server ist das Gegenstück zum Association Client, der sich in jedem mobilen Endgerät befindet und bereits im Abschnitt 3.2.1 beschrieben wurde, und ist für die Verwaltung der Verbindungen zu den Mobilgeräten zuständig. Darüber hinaus ist in jedem Node B ein Bandwidth Assigner aktiv, der jedem verbundenen mobilen Gerät entsprechend der gewählten Zuweisungsstrategie Bandbreite für Up- und Downstream zuweist. Die Funktionen des Bandwidth Assigner werden im Abschnitt 3.3.2 beschrieben. In jedem Node B sind die Layer zwei und drei eines TCP/IP-Stacks vorhanden. Auf dem Link Layer befinden sich zwei Interfaces: ein UMTS-Interface, das mit den verbundenen Mobilgeräten kommuniziert und ein PPP-Interface, das mit dem UMTS Gateway verbunden ist. Alle von Mobilgeräten oder dem UMTS Gateway ankommende Pakete werden über die Interfaces an den Network Layer weitergeleitet. Dieser entscheidet anschließend anhand einer Routingtabelle, über welches Interface ein Paket weitergeleitet werden muss, um zum Zielgerät zu gelangen. Das Routing von Downstream-Paketen in Richtung der Mobilgeräte geschieht durch den Packet Distributor. Dazu werden die Einträge der Forwardtabelle genutzt, die zu jedem verbundenen Mobilgerät das Gate speichert, über das es am Node B zu erreichen ist.

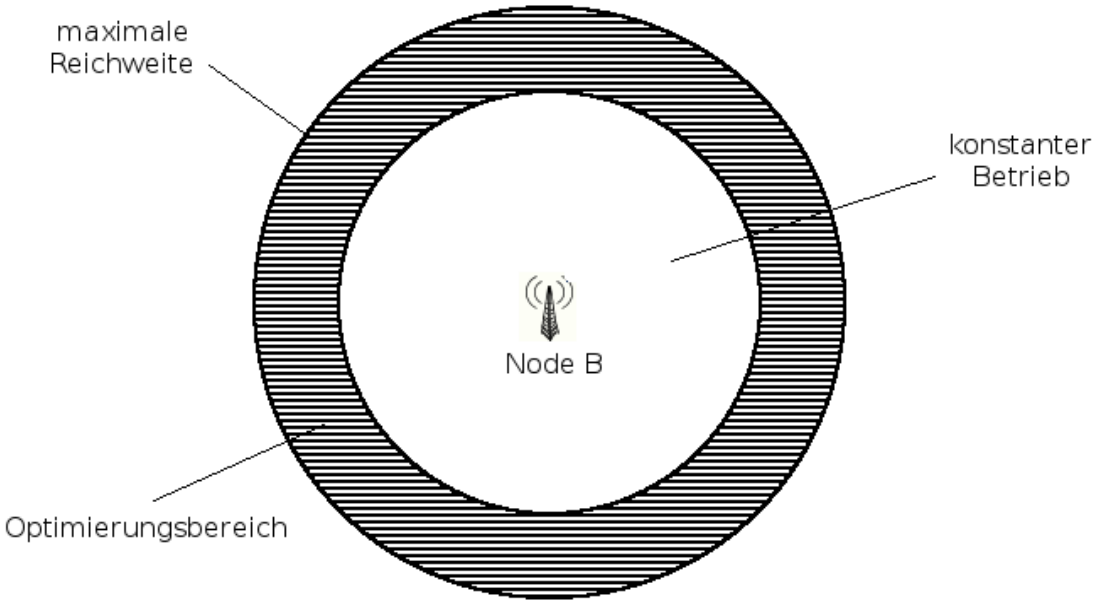


Abbildung 3.4: Node B mit maximaler Reichweite und Optimierungsbereich

Die Simulation unterstützt die Verwendung mehrerer UMTS-Provider. Jeder Provider besitzt eine eindeutige ID. Allen Node B auf der Simulationsfläche wird eine dieser IDs zugeteilt. Darüber hinaus besitzt jedes mobile Gerät auch einen Stammprovider, zu dessen Node B sich das Mobilgerät bevorzugt verbindet. Erst wenn keine Verbindung zu einem Funkmasten des Stammproviders hergestellt werden kann, werden die anderen erreichbaren Node B in der Umgebung in Betracht gezogen. Diese Funktionalität kann später zum Beispiel dazu genutzt werden, Roamingkosten zu modellieren.

Die Reichweite des Node B deckt einen kreisförmigen Bereich ab. Es gibt keinerlei Unterschiede in der Empfangsqualität, die abhängig von der Distanz zum Node B ist, da auf die Modellierung einer exakten physikalischen Übertragung verzichtet wurde. Pakete zwischen Mobilgeräten und Node B werden dementsprechend alle ohne Bitfehler übertragen. Die Modellierung der erhöhten Fehlerrate des Mobilfunkkanals im Gegensatz zu einer kabelgebundenen Übertragung spiegelt sich indirekt in einer erhöhten Verzögerung der Pakete wider. Der Verzögerungsspitzengenerator ist dafür zuständig, die benötigte Zeit für einen Retransmit bei fehlerhafter Übertragung zu modellieren.

In den Node B befindet sich außerdem der letzte Teil des in [CGGPC06] vorgeschlagenen Modells zur Modellierung der Verzögerung in UMTS-Systemen. Jedes von einem Mobilgerät ankommende Paket passiert zunächst eine Komponente, die jedes Paket um eine bestimmte konstante Zeit verzögert. Analog dazu wird die Paketlaufzeit von allen Downstream-Paketen durch ein Modul für konstante Verzögerung verlängert. Damit befindet sich in Downstream-Richtung die konstante Verzögerung in der Reihenfolge vor dem Verzögerungsspitzengenerator und dem TTI Server, doch da die Verzögerung konstant ist,

hat diese Umstellung keine weiteren Auswirkungen. Von einem Mobilgerät stammende Pakete werden anschließend an das UMTS-Interface weitergeleitet. Downstream-Pakete, die an ein lokal verbundenes Gerät adressiert sind, sendet der Packet Distributor an das entsprechende Mobilgerät.

Der Node B ist das Verbindungsstück zwischen Funkübertragung und drahtgebundener Übertragung. Dazu besitzt er ein UMTS-Interface und ein Point-To-Point-Interface, das mit dem UMTS Gateway verbunden ist. In realen UMTS-Systemen kommt hier ATM an Stelle des Point-To-Point-Protokolls zum Einsatz, allerdings ist es im INET/MANET-Framework nicht implementiert. Da sich die Protokolle von ATM und PPP sehr ähnlich sind, wurde auf eine Modellierung von ATM verzichtet und stattdessen PPP verwendet.

Ankommende Pakete werden über die Interfaces an den Network Layer weitergeleitet. Dort wird mit einer Routingtabelle ermittelt, über welches des beiden Interfaces die Zieladresse zu erreichen ist und das Paket dementsprechend weitergeleitet. Die Existenz des Link Layers und des Network Layers entsprechen der Realität, allerdings wird der Link Layer hier nicht weiter unterteilt und modelliert nur die effektive Übertragungszeit ohne Auftreten von Übertragungsfehlern. Der Einfluss einer wiederholten Übertragung bei Auftreten von Bitfehlern wird bereits durch den Verzögerungsspitzengenerator in einem Mobilgerät modelliert, sodass die Modellierung des RLC Sub Layers nicht mehr nötig ist.

3.3.1 Association Server

Der Association Server verwaltet die Verbindungen zu den verbundenen mobilen Geräten und stellt somit das Gegenstück zum bereits vorgestellten Association Client in den Mobilgeräten dar. Wenn ein Verbindungswunsch eines Mobilgeräts besteht, wird zuerst vom Bandwidth Assigner geprüft, ob noch Kapazitäten für eine weitere Verbindung bestehen. Sollte dies der Fall sein, baut der Association Server eine Verbindung auf und aktualisiert die Routingtabellen im Node B, im UMTS Gateway und in der Internetkomponente, damit das neu verbundene Gerät über den zugehörigen Node B zu erreichen ist. Analog dazu verläuft der Abbruch einer Verbindung: zuerst wird die physikalische Verbindung zum Mobilgerät abgebrochen und die beim Aufbau der Verbindung erstellten Einträge in den Routingtabellen werden wieder gelöscht.

3.3.2 Bandwidth Assigner

Jeder Node B besitzt einen Bandwidth Assigner. Er ist dafür zuständig, die in der Konfigurationsdatei festgelegte Bandbreite für Upload und Download, die je Node B für mobile Geräte zur Verfügung steht, auf die momentan verbundenen Mobilgeräte zu verteilen. Darüber hinaus entscheidet er auch je nach gewählter Zuweisungsstrategie, ob ein neues Mobilgerät sich zum Node B verbinden kann oder ob Bandbreitengarantien verletzt werden und somit die Verbindungsanfrage abgelehnt werden muss. In [TLL08] kann man anhand der gemessenen verfügbaren Bandbreite der einzelnen Mobilgeräte sehen, dass von

Algorithmus 3.1 Berechnung der Bandbreitenverteilung

```
procedure distributeBandwidth( )
    bandwidth per station ← Node B bandwidth / #stations
    assignEqually ← bandwidth per station ≥ guaranteed bandwidth
    if (assignEqually = TRUE) then
        if (bandwidth per station > maximum bandwidth) then
            bandwidth per station ← maximum bandwidth
        end if
        leftover bandwidth ← Node B bandwidth MOD #stations
        assign calculated bandwidth to all stations
        if (bandwidth per station < maximum bandwidth) ∧ (leftover bandwidth > 0) then
            distribute leftover bandwidth equally among stations
        end if
    else
        leftover bandwidth ← Node B bandwidth
        currentStation ← index of first connected station minus 1
        while (leftover bandwidth > guaranteed bandwidth) do
            assign guaranteed bandwidth to currentStation
            currentStation++
        end while
        if (leftover bandwidth > 0) then
            distribute leftover bandwidth equally among stations with index < currentStation
        end if
    end if
end procedure
```

Provider zu Provider unterschiedliche Zuweisungsstrategien verwendet werden. Zwei dieser Zuweisungsstrategien wurden in dieser Arbeit aufgegriffen und implementiert.

Die Strategie der gleichmäßigen Verteilung, bei der die gesamte Bandbreite gleichmäßig auf alle verbundenen Geräte verteilt wird, ist die erste der hier verwendeten Strategien. Es gibt keinerlei Bandbreitengarantien und es ist für jedes Mobilgerät in Empfangsreichweite des Node B möglich, sich mit diesem zu verbinden.

Die zweite hier implementierte Zuweisungsstrategie garantiert jedem verbundenen mobilem Endgerät eine Mindestbandbreite sowohl im Upstream als auch im Downstream. Wenn eine Verbindungsanfrage beim Node B eintrifft, allerdings nicht mehr genug Bandbreite verfügbar ist, um dem Mobilgerät die Mindestbandbreite zuweisen zu können, so wird der Verbindungswunsch abgelehnt. Dies kann allerdings durch einen Parameter gesteuert werden: Es ist möglich, dass sich das mobile Gerät zum Node B verbinden kann, jedoch keinerlei Bandbreite zugewiesen bekommt. Das kann in manchen Fällen sinnvoll sein, wenn zum Beispiel ein anderes zum Node B verbundenes Gerät die Verbindung abbricht und somit Bandbreite frei wird, die dann wiederum an das Endgerät, dem bisher keine Bandbreite zugewiesen wurde, weitergegeben werden kann.

Bei einem Verbindungsversuch eines Mobilgeräts fragt der zugehörige Association Client beim Bandwidth Assigner an, ob noch eine weitere Verbindung möglich ist. Es wird zuerst geprüft, ob der Simulationsparameter gesetzt wurde, der einem Mobilgerät jederzeit erlaubt sich zu verbinden. Wenn dies der Fall ist, kann sich ein Mobilgerät verbinden, obwohl es eventuell je nach Zuweisungsstrategie keine Bandbreite erhält. Andernfalls wird mit den Gleichungen 3.1 und 3.2 berechnet, ob genügend Uplink- und Downlinkbandbreite zur Verfügung steht.

$$\frac{Downlinkbandbreite}{Anzahl\ Mobilgeräte + 1} \geq garantierte\ Downlinkbandbreite \tag{3.1}$$

$$\frac{Uplinkbandbreite}{Anzahl\ Mobilgeräte + 1} \geq garantierte\ Uplinkbandbreite \tag{3.2}$$

Damit einem Mobilgerät erlaubt wird sich zu verbinden, müssen beide Bedingungen erfüllt sein. Es gibt zwei Möglichkeiten, dies zu gewährleisten. Daraus resultieren auch die zwei unterschiedlichen Zuweisungsstrategien: Zum einen kann festgelegt werden, dass keine Bandbreitengarantien vergeben werden, d.h. die garantierte Up- und Downlinkbandbreite 0 kBit/s beträgt. Dann ist es einem Mobilgerät immer möglich, sich zum Node B zu verbinden. Die andere Möglichkeit besteht darin, eine Bandbreitengarantie für alle verbundene Gerät abzugeben. Dann ist die Aufnahme eines neuen mobilen Endgeräts nur dann möglich, wenn die Garantien für alle verbundenen Geräte in sowohl im Upstream als auch im Downstream eingehalten werden können.

Wenn nun entschieden wurde, dass sich das neue Mobilgerät verbinden darf, wird eine Neuverteilung der Bandbreite durchgeführt. Es ist die gleiche Prozedur, die auch dann ausgeführt wird, wenn ein Mobilgerät die Verbindung zum Node B trennt. Die Berechnung der Verteilung ist für den Upstream und den Downstream identisch. Der Algorithmus 3.1 zur Bandbreitenverteilung stellt den Ablauf der Berechnung dar. Zuerst wird überprüft, ob genügend Bandbreite zur Verfügung steht, um die geforderten Garantien einzuhalten. Wenn dies der Fall ist, wird die gesamte Bandbreite gleichmäßig auf die verbundenen Mobilgeräte verteilt, ohne dass die maximale Bandbreite pro Gerät überschritten wird. Andernfalls wird so vielen mobilen Endgeräten wie möglich die garantierte Bandbreite zugewiesen. Unter diesen Geräten wird dann auch noch die, falls vorhanden, verbliebene Bandbreite aufgeteilt. Die Mobilgeräte, denen keine garantierte Bandbreite zugewiesen wurde, können somit keine Daten übertragen.

3.3.3 Node B-Platzierungsstrategien

Die Node B jedes Providers müssen auf der Simulationsgrundfläche platziert werden. Diese hat ihren Ursprung $(0, 0)$ in der linken oberen Ecke und erstreckt sich von dort nach rechts und nach unten, wie in Abbildung 3.5 zu sehen ist.

Es wurden zwei verschiedene Platzierungsstrategien ausgewählt und implementiert. Je nach Bedarf können weitere Strategien implementiert und der Simulation hinzugefügt werden.

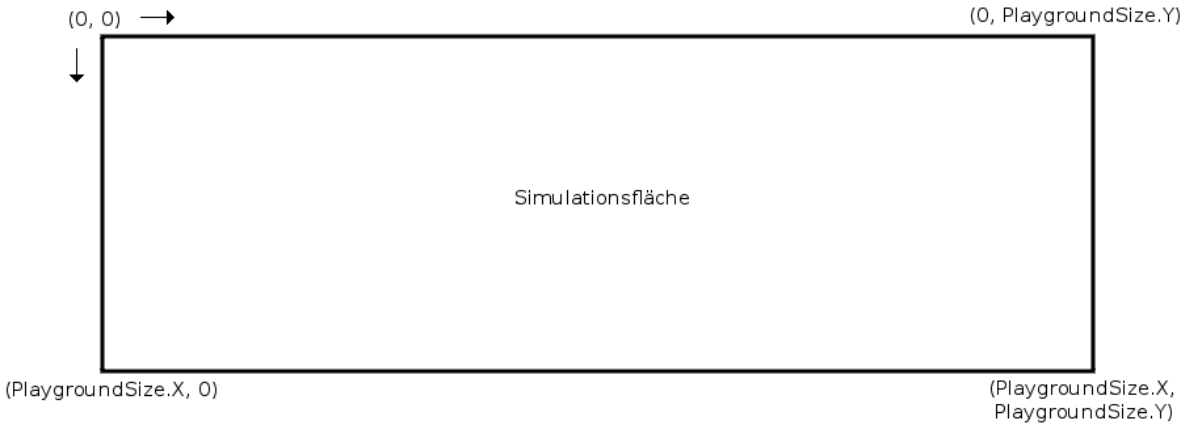


Abbildung 3.5: Koordinatensystem der Simulationsfläche

3.3.3.1 Gleichmäßige Platzierung

Diese Strategie platziert die Node B gleichmäßig in einem Wabenmuster. Dabei soll eine 100%ige Abdeckung der Simulationsgrundfläche erreicht werden, sodass kein Funkloch entsteht. Die maximale Reichweite und die Größe des Optimierungsbereichs der Node B können frei gewählt werden.

Es wird zuerst berechnet, wie viele Node B nötig sind, um die Grundfläche vollständig mit Waben zu bedecken, wobei die Wabengröße von der Reichweite der Node B abhängig ist. Zusätzlich wird jedem Node B bei der Platzierung noch eine zufällig ausgewählte Provider-ID zugewiesen. Ein anschauliches Beispiel ist in Abbildung 3.6 zu sehen. Das blaue Rechteck stellt die Simulationsgrundfläche dar. Die gebildeten Waben sind in rot eingezeichnet. Es ist ebenfalls zu sehen, dass am unteren Rand Unregelmäßigkeiten im Wabenmuster auftreten. Die reguläre Platzierung der Node B läge hier außerhalb der Grundfläche und wäre somit nicht zulässig. Deshalb wird hier das reguläre Muster aufgehoben und die betreffenden Node B werden soweit verschoben, dass sie sich nun auf der Grenze der Grundfläche befinden, um die gewünschte 100%ige Abdeckung der Simulationsgrundfläche zu gewährleisten.

Der Algorithmus 3.2 in Pseudocode dient zur gleichmäßigen Platzierung der Node B. Nicht abgebildet ist die Abfrage, ob bereits ein einziger Node B in der Mitte der Grundfläche ausreicht, um sie vollständig abzudecken. In diesem Fall wird der Algorithmus nicht ausgeführt.

Die einzelnen Node B sollen im Mittelpunkt der einzelnen Waben stehen. Aufgrund der Anordnung der Waben kann man sie in Zeilen und Spalten einteilen, wobei die Node B in der gleichen Spalte eine gemeinsame X-Koordinate und die Node B in der gleichen Zeile eine gemeinsame Y-Koordinate besitzen. Die Aufteilung der Node B in Zeilen und Spalten kann in Abbildung 3.6 beobachtet werden. Der Abstand der Mittelpunkte zweier Waben einer Spalte beträgt $radius \times \sqrt{3}$, der der Mittelpunkte zweier Waben einer Zeile $radius \times 1.5$. Diese zwei Konstanten werden gleich zu Beginn deklariert. Darüber hinaus werden drei boolesche Variablen eingeführt, die spezielle Positionen des aktuell zu platzierenden Node B angeben:

Algorithmus 3.2 Algorithmus zur gleichmäßigen Platzierung

```
1: procedure placeNodeBs( )
2:     DistanceLowerNodeB ← radius × √3
3:     DistanceRightNodeB ← radius × 1.5
4:     even, ReachedRightEnd, ReachLowerEnd ← FALSE
5:     CurrentPos.X ← 0 ∧ CurrentPos.Y ← 0
6:     while (¬ ReachedRightEnd) do
7:         if (CurrentPos.X ≤ 0) then
8:             CurrentPos.X ← 0.5 × radius
9:             even ← FALSE
10:        else if (CurrentPos.X + 0.5 × radius ≥ DistanceRightNodeB) then
11:            if (CurrentPos.X + 0.5 × radius ≥ PlaygroundSize.X) then
12:                ReachedRightEnd ← TRUE
13:            else
14:                CurrentPos.X ← PlaygroundSize.X
15:                even ← ¬ even
16:            end if
17:        else
18:            CurrentPos.X ← CurrentPos.X + DistanceRightNodeB
19:            even ← ¬ even
20:        end if
21:        if (even) then
22:            CurrentPos.Y ← 0
23:        else
24:            CurrentPos.Y ← 0.5× DistanceLowerNodeB
25:        end if
26:        ReachedLowerEnd ← FALSE
27:        if (¬ ReachedRightEnd) then
28:            addNodeB(CurrentPos, radius, random Provider ID)
29:        end if
30:        while (¬ ReachedLowerEnd) do
31:            if (CurrentPos.Y + DistanceLowerNodeB ≥ PlaygroundSize.Y) then
32:                if (CurrentPos.Y + 0.5× DistanceLowerNodeB ≥ PlaygroundSize.Y) then
33:                    ReachedLowerEnd ← trTRUE
34:                else
35:                    CurrentPos.Y ← PlaygroundSize.Y
36:                end if
37:            else
38:                CurrentPos.Y ← CurrentPos.Y + DistanceLowerNodeB
39:            end if
40:            if (¬ ReachedLowerEnd ∧¬ ReachedRightEnd) then
41:                addNodeB(CurrentPos, radius, random Provider ID)
42:            end if
43:        end while
44:    end while
45: end procedure
```

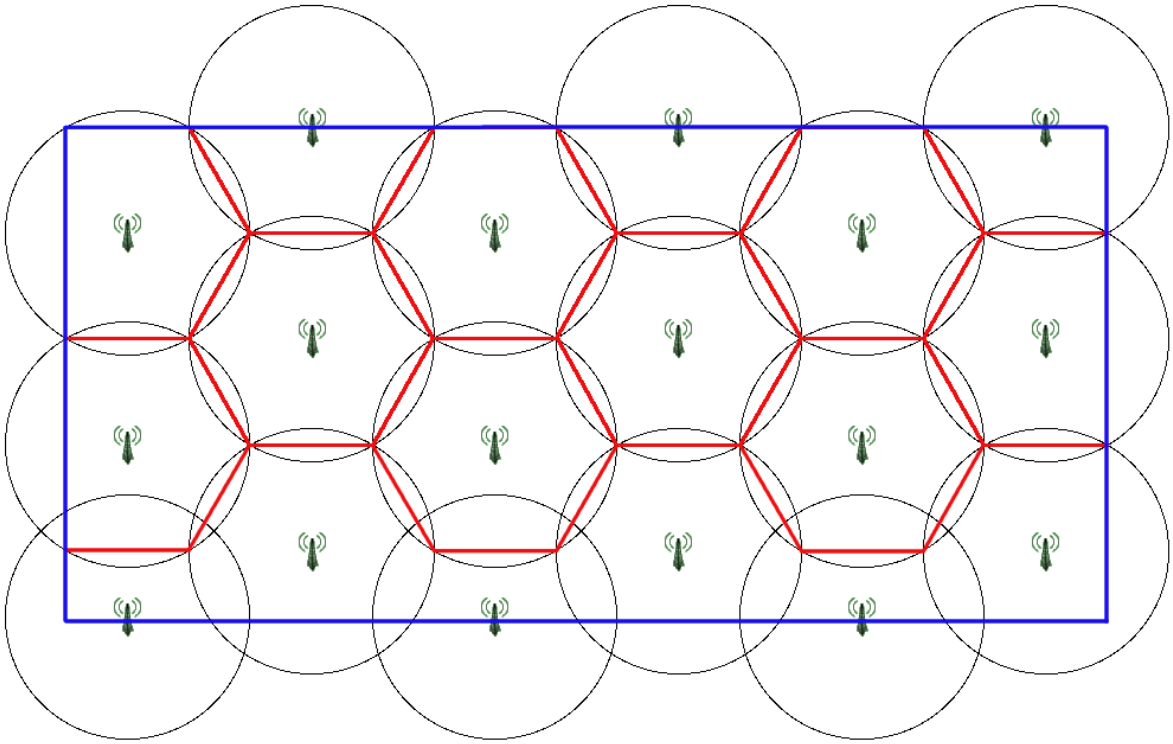


Abbildung 3.6: Gleichmäßige Platzierung der Node B (maximale Reichweite: 25 m; Grundfläche 200 m * 100 m)

ReachedRightEnd und ReachedLowerEnd erhalten den Wert TRUE, wenn der nächste zu platzierende Node B sich auf der rechten bzw. unteren Grenze der Simulationsfläche oder darüber hinaus befinden würde. Die Variable even erhält den Wert TRUE, wenn sich der oberste Node B der aktuellen Spalte sich auf der oberen Grenze der Grundfläche befinden soll, FALSE wenn nicht. Schließlich wird noch vor Beginn der Iterationen die aktuelle Position des zu platzierenden Node B definiert (Zeilen 2-5).

Anschließend beginnt die Iteration und die Node B werden Spalte für Spalte platziert. Zuerst wird in jedem Durchlauf die X-Koordinate der aktuellen Spalte bestimmt (Zeile 7-20). Die erste Spalte beginnt mit $x = 0.5 \times radius$. Danach wird jede Spalte um den bereits berechneten konstanten Abstand verschoben und der Indikator even wechselt zwischen TRUE und FALSE. Dieser Vorgang bricht ab, wenn die rechte Grenze der Simulationsgrundfläche erreicht wird. Dann wird überprüft, ob noch ein weiterer Node B nötig ist, um vollständige Abdeckung zu erhalten. Sollte dies nicht der Fall sein, wird ReachedRightEnd auf TRUE gesetzt und nach Auffüllen dieser Spalte bricht der Algorithmus ab.

Nachdem die X-Koordinate bestimmt wurde, muss noch in jeder Iteration die zugehörige Y-Komponente des zu platzierenden Node B bestimmt werden (Zeile 21-43). Dazu wird zuerst überprüft, ob es sich um eine Spalte handelt, in der der oberste Node B direkt auf die obere Grenze gesetzt werden muss (even ist TRUE, y = 0), oder ob er leicht versetzt platziert werden muss (even ist FALSE, $y = 0.5 \times DistanceLowerNodeB$). Falls die aktuelle

Algorithmus 3.3 Algorithmus zur zufälligen Platzierung

```
 1: procedure placeNodeBs( )
 2:     CurrentPos.X ← 0 ∧ CurrentPos.Y ← 0
 3:     radius ← 0
 4:     nodeBcreated ← 0
 5:     while nodeBcreated < number of Node B do
 6:         CurrentPos.X ← uniform(0, PlaygroundSize.X)
 7:         CurrentPos.Y ← uniform(0, PlaygroundSize.Y)
 8:         radius ← read from Config File
 9:         addNodeB(CurrentPos, Radius, random Provider ID)
10:         nodeBcreated ← nodeBcreated + 1
11:     end while
12: end procedure
```

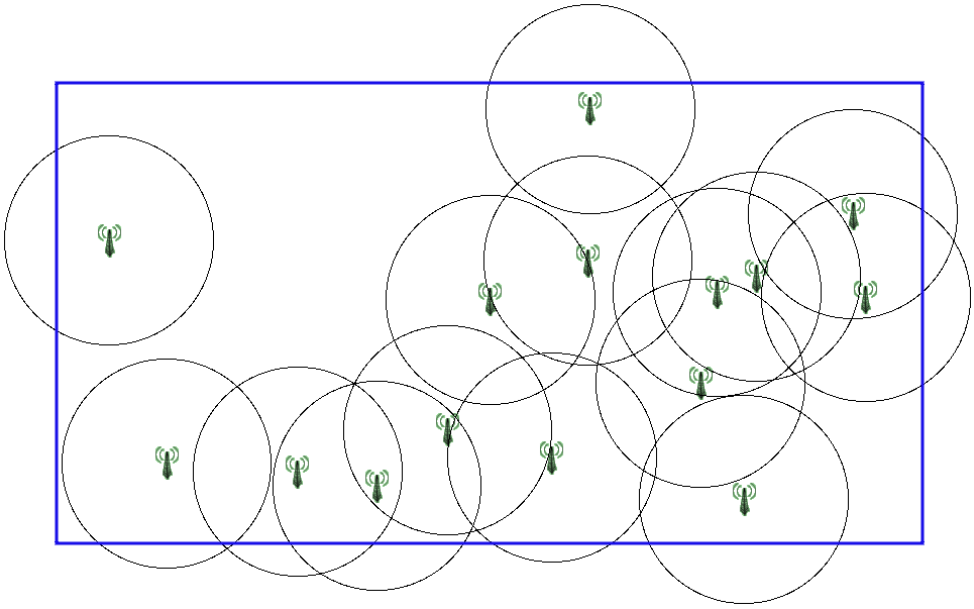
Spalte dann noch innerhalb der Grundflächengrenze liegt, wird die Funktion *addNodeB* aufgerufen, die den Node B dem System hinzufügt. Anschließend beginnt eine Iteration über die einzelnen Zeilen, die analog zu der Iteration über die Spalten abläuft. Am Ende jeder Iteration wird hier allerdings mit den aktuell berechneten Koordinaten wiederum die Funktion *addNodeB* aufgerufen, sofern sich die Position noch innerhalb der Grundfläche befindet.

3.3.3.2 Zufällige Platzierung

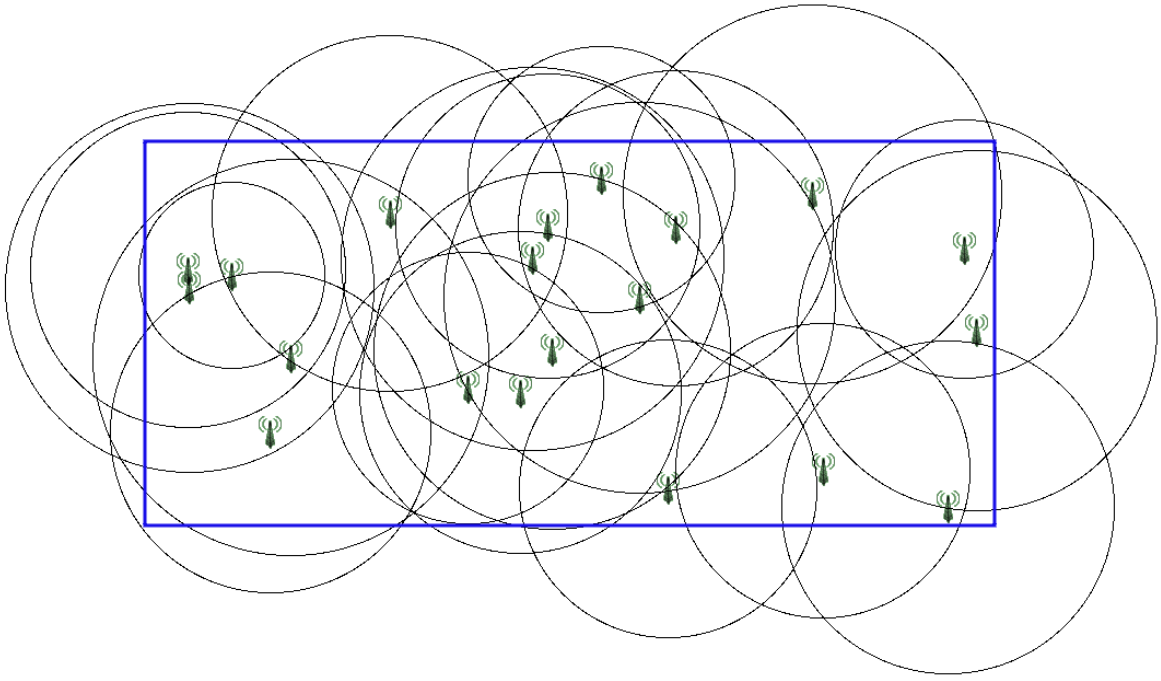
Diese Strategie platziert die Node B zufällig auf der Simulationsfläche. Dabei kann die Anzahl der zu platzierenden Node B und die zugehörigen Radien der Erreichbarkeitskreise vom Nutzer bestimmt werden. Es werden dann nach und nach Node B erstellt, ihnen eine zufällige Provider-ID zugewiesen und mit zufällig gleichverteilten Koordinaten platziert. In Abbildung 3.7(a) und 3.7(b) sind zwei beispielhafte Simulationsflächen mit zufälliger Node B-Platzierung zu sehen. Hierbei fällt auf, dass durch die fehlende Struktur und die zufällige Platzierung der Node B keine Gewährleistung für eine 100%ige Abdeckung der Simulationsgrundfläche gegeben werden kann. Deshalb ist diese Platzierungsstrategie zum Beispiel dazu geeignet, das Verhalten des UMTS-Systems bei Auftreten von Funklöchern zu beobachten.

Der Algorithmus zur zufälligen Platzierung der Node B ist in 3.3 zu sehen. Er ist deutlich weniger komplex als der Algorithmus zur gleichmäßigen Platzierung.

Zu Beginn werden die Variablen der aktuellen Node B-Position und der Anzahl an bereits erstellen Node B mit 0 initialisiert. Anschließend werden nacheinander die Node B platziert und durch Aufruf der Funktion *addNodeB* dem System hinzugefügt. Dabei wird für jeden Node B die Position zufällig gleichverteilt auf der Simulationsgrundfläche gewählt. Der Platzierungsvorgang endet, wenn die vom Nutzer gewählte Anzahl an Node B erstellt wurde.



(a) 15 Node B; maximale Reichweite: 25 m; Grundfläche 200 m * 100 m



(b) 20 Node B; maximale Reichweite: gleichverteilt in (20 m, 50 m); Grundfläche 200 m * 100 m

Abbildung 3.7: Zufällige Platzierung der Node B

3.4 UMTS Gateway

Der UMTS Gateway stellt das Verbindungsstück zwischen den im System vorhandenen Node B und dem Internet dar. Im Gegensatz zu realen UMTS-Systemen wird die Verbindung direkt und nicht über das Core Network hergestellt. Der UMTS Gateway fasst die Funktionalität aller RNC eines UMTS-Systems zusammen. Er besitzt $(AnzahlNodeB + 1)$ PPP-Interfaces, wobei das erste für eine Verbindung zur Internetkomponente verwendet wird. Wegen der Ähnlichkeit von PPP zu ATM kommt auch hier wie bei der Verbindung der Node B mit dem Gateway PPP anstatt ATM zum Einsatz, da dieses bereits im INET/MANET-Framework implementiert ist. Jedes weitere Interface ist mit einem Node B verbunden, sodass der UMTS Gateway mit allen Node B kommunizieren kann.

3.5 Internet

Das Internet ist ebenfalls eine Teilkomponente der Simulation. Es ist nicht wie in Realität aus einem Verbund vieler einzelner Router aufgebaut. Die Internetimplementierung dieser Simulation besteht aus einem Link Layer mit mehreren Point-To-Point-Interfaces und einem Network Layer. Alle ankommenden Pakete passieren zunächst noch eine Verzögerungskomponente, die jedes Paket für eine bestimmte Zeitspanne verzögert. Dies modelliert die Verzögerung eines Pakets im realen Internet. Das erste PPP-Interface ist mit dem UMTS Gateway verbunden, jedes weitere kann dazu genutzt werden, einen stationären Host zu verbinden. Alle ankommenden Pakete erreichen schließlich den Network Layer, der dann anhand einer Routingtabelle entscheidet, über welches der Interfaces es weitergeleitet werden muss, um die Zieladresse zu erreichen. Diese Weise der Implementierung einer Internetkomponente ist recht einfach gehalten, da sie nicht Hauptaugenmerk der Arbeit war. Sie kann durch die Modularität von OMNeT++ leicht durch eine realistischere und komplexere Implementierung ersetzt werden, die das reale Verhalten des Internets besser modelliert.

3.6 IP-Vergabe & Routing

Um eine Kommunikation sowohl zwischen zwei Mobilgeräten als auch zwischen Mobilgeräten und stationären Servern zu ermöglichen, benötigen alle beteiligten Geräte eine IP-Adresse, auf deren Basis ein Hop-By-Hop-Routing implementiert wird. Sollte ein Gerät mehrere Interfaces besitzen, so wird allen Interfaces dieselbe IP-Adresse zugewiesen. Im Gegensatz zur Realität wird die Zuweisung bereits zu Beginn der Simulation durchgeführt, wenn noch keinerlei Verbindung besteht.

Die möglichen Kommunikationswege im UMTS-Netz sind in Abbildung 3.8 zu sehen. Der erste mögliche Weg ist das Senden eines Pakets von Mobilgerät zu Mobilgerät, wenn beide zum gleichen Node B verbunden sind. Das betreffende Paket wird zuerst zum Node B weitergeleitet, zu dem die Mobilgeräte verbunden sind. Dort wird anhand der Routingtabelleneinträge überprüft, ob die Zieladresse einem ebenfalls lokal verbundenen Gerät

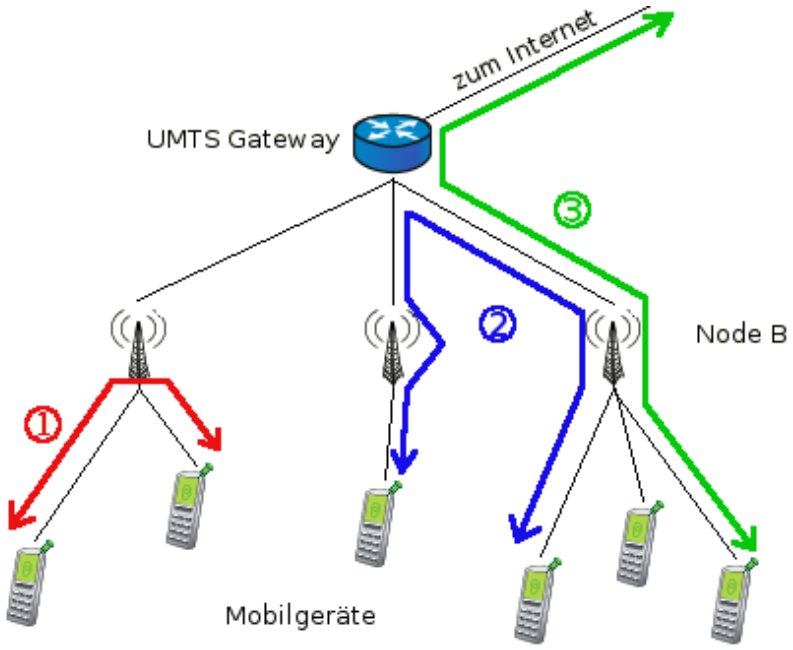


Abbildung 3.8: Mögliche Kommunikationswege im UMTS-System

zugewiesen ist. Sollte das der Fall sein, leitet der Packet Distributor des Node B das Paket an das Zielgerät weiter. Wenn die Zieladresse nicht lokal verbunden ist, wird das Paket standardmäßig an den UMTS Gateway gesendet. Dort wird dann wiederum überprüft, ob sich die Zieladresse im UMTS-System befindet und gegebenenfalls an den entsprechenden Node B und von dort an das Ziel weitergeleitet. Dies ist der zweite Fall, der in der Abbildung 3.8 eingezeichnet ist. Sollte die Zieladresse einem an das Internet angeschlossenem Host gehören, leitet der UMTS Gateway das Paket an das Internet weiter, was die dritte Möglichkeit der Kommunikation darstellt. Analog dazu, nur in umgekehrter Richtung, verläuft das Routing, wenn ein Paket aus dem Internet am UMTS Gateway eintrifft.

Ein Paket, das in einem Node B eintrifft und dessen Zieladresse in der Routingtabelle keinen Eintrag besitzt, wird standardmäßig an den UMTS Gateway weitergeleitet. Analog dazu verfährt der UMTS Gateway und sendet ein Paket mit unbekannter Zieladresse an das Internet. Wurde das Paket allerdings bereits über die Standardroute empfangen und es existiert kein Eintrag in der jeweiligen Routingtabelle, wird es verworfen.

4 Implementierung

In diesem Kapitel wird die Implementierung des UMTS-Modells beschrieben. Der verwendete Netzwerksimulator OMNeT++ nutzt die Programmiersprache C++. In Abbildung 4.1 ist ein Klassendiagramm der verwendeten Module mit den wichtigsten Funktionen zu sehen.

Das bereits bestehende INET/MANET-Framework diente als Grundlage. Es stellt die Implementierung eines TCP/IP-Stacks zur Verfügung. Weiterhin sind bereits diverse Mobilitätsmuster und einige Hilfsklassen wie FIFO-Queues vorhanden, die in dieser Arbeit genutzt werden. Es wurde versucht, möglichst viel aus dem Framework zu verwenden, um eine hohe Kompatibilität zu anderen Projekten zu erreichen, die ebenfalls auf diesem Framework aufbauen.

Im Folgenden werden nun die NED-Dateien der einzelnen Komponenten in einer Übersicht und die wichtigsten Funktionen der Module vorgestellt.

4.1 NetConfigurator

Der NetConfigurator ist in der Initialisierungsphase nach der Platzierung der Node B aktiv und vergibt an jedes Gerät im System eine IP-Adresse, das Interfaces besitzt. In Tabelle 4.1 sind die Parameter des NetConfigurators zu sehen, die über die Konfigurationsdatei eingestellt werden können.

Der NetConfigurator füllt die Routingtabellen entsprechend der Topologie der Geräte. Im INET/MANET-Framework existieren bereits mehrere Versionen des NetConfigurators, die dann als Grundlage für die hier implementierte Version dienten. Die Initialisierungsphase ist in einzelne Stufen eingeteilt; der NetConfigurator arbeitet dabei ausschließlich in Phase zwei. Seine Arbeitsschritte können dabei wiederum in fünf Unterphasen unterteilt werden:

Extrahieren der Topologie Alle Geräte mit IP-fähigen Interfaces, im vorliegenden System die Mobilgeräte, alle Node B samt UMTS Gateway und die Internetkomponente, werden in ihrer NED-Datei-Definition mit einem Label namens "node"deklariert. Der NetConfigurator besitzt eine Instanz der OMNeT++-eigenen Klasse "cTopology". Diese durchsucht alle erstellten Module der Simulation nach diesem Label und erstellt aus ihnen eine Topologie. Zwei Knoten werden durch eine Kante verbunden, wenn sie durch eine direkte Verbindung Kontakt zueinander haben, wie es hier zum Beispiel zwischen den Node B und dem UMTS Gateway der Fall ist.

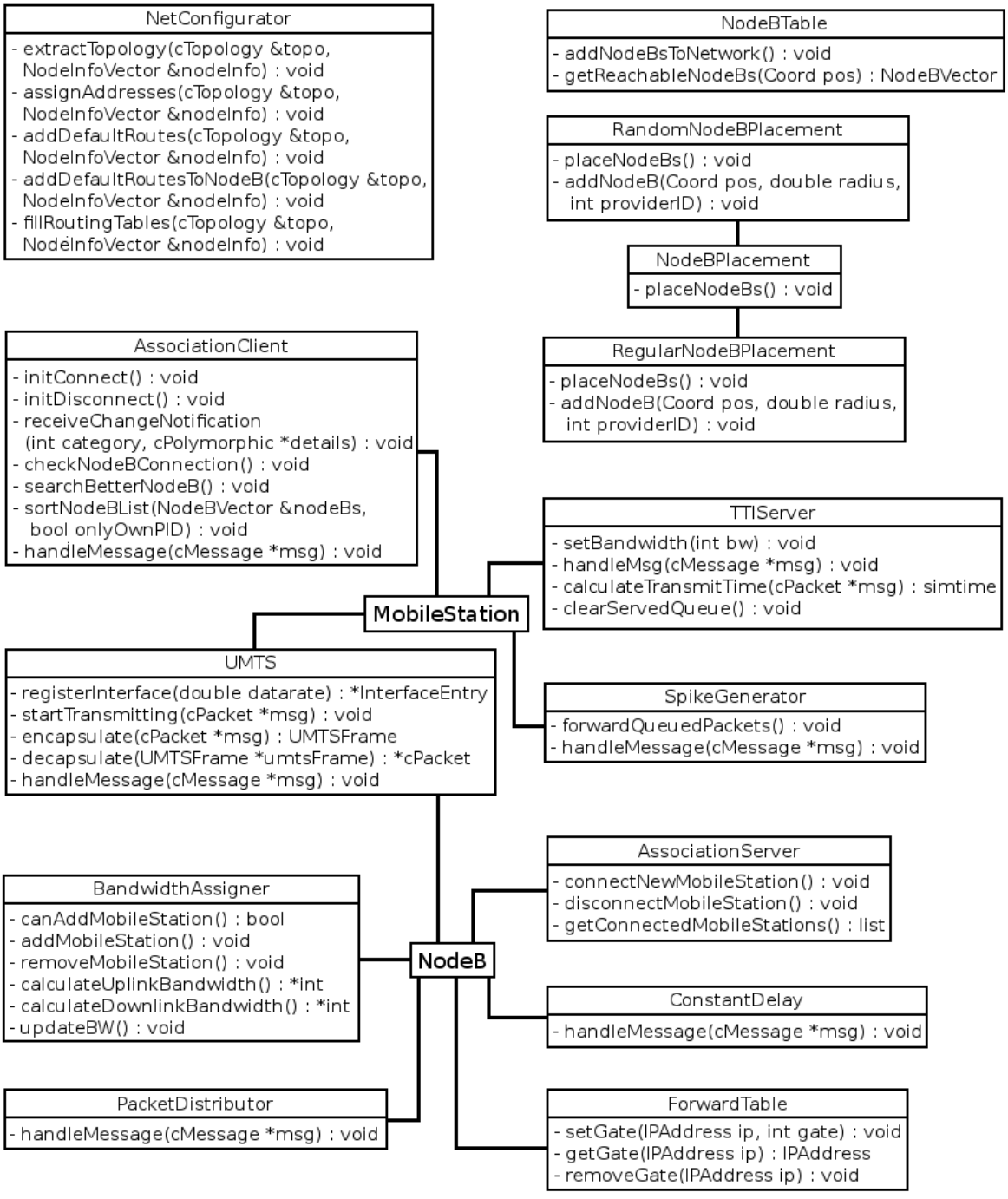


Abbildung 4.1: Klassendiagramm

Zuweisung der IP-Adressen Jeder Knoten im System besitzt eine Interfacetabelle, in der Informationen zu seinen Interfaces gespeichert werden. Nachdem die Topologie extrahiert wurde, werden die Interfacetabellen aller Knoten abgearbeitet und dabei jedem Interface eine IP-Adresse zugewiesen. Dabei erhalten alle Interfaces eines Geräts die gleiche IP-Adresse. Die Netzadresse und die Netzmaske können über die Konfigurationsdatei bestimmt werden. Dabei ist darauf zu achten, dass die Netzmaske so gewählt wurde, dass genügend IP-Adressen zur Verfügung stehen und allen Geräten eine Adresse zugewiesen werden kann.

Standardroute für Single Interface Hosts eintragen Nachdem die IP-Adressen zugewiesen sind, wird in allen Hosts mit einem einzigen Interface (Loopback Interfaces ausgenommen) eine Standardroute eingetragen. Diese besagt trivialerweise, dass alle Pakete über die einzige Verbindung des Interfaces gesendet werden sollen.

Default Route für Node B eintragen In die Routingtabelle jedes Node B wird eine Default Route eingetragen. Diese besagt, dass alle Pakete über das PPP-Interface an den UMTS Gateway gesendet werden, falls keine andere Route gefunden wurde. Das ist nötig, wenn ein Paket eines verbundenen Mobilgeräts an ein Gerät geschickt werden soll, das nicht mit dem gleichen Node B verbunden ist wie der Sender des Pakets.

Routingtabellen berechnen Nachdem die Default-Routen eingetragen wurden, müssen noch die statischen Routen eingetragen werden. Dazu werden mit Hilfe des Algorithmus von Dijkstra die kürzesten Wege zwischen den Knoten berechnet und jeweils der Knoten, der auf dem kürzesten Weg zum Zielhost den nächsten Hop darstellt, als Route für die Zieladresse eingetragen. Der Dijkstra-Algorithmus musste nicht mehr implementiert werden, da er bereits als Teil der cTopology-Klasse vorhanden ist.

Parameter	Standardwert
String NetworkAddress	"192.168.0.0"
String Netmask	"255.255.0.0"

Tabelle 4.1: Parameter des NetConfigurators

4.2 NodeBTable

Die Node B Table hat die Aufgabe, eine zentrale Tabelle über alle im System vorhandenen Node B samt Informationen wie Reichweite oder Position zu speichern. Alle Module, die im Laufe der Simulation Informationen über die Node B benötigen, können dann auf diese Tabelle zugreifen.

Das Modul zur Platzierung der Node B erstellt in der NodeBTable während der Initialisierungsphase für jeden Node B einen Eintrag. Dieser enthält die Koordinaten des Node B auf der Simulationsgrundfläche, seine maximale Übertragungsreichweite und seine Provider-ID. Nachdem alle Node B platziert wurden, ruft das Platzierungsmodul die Funktion *addNodeBsToNetwork* der NodeBTable-Klasse auf. Diese erhöht als erstes die Anzahl der

Gates des UMTS Gateways für PPP-Interfaces entsprechend der Anzahl an Node B, ebenso wie die des Network Layers innerhalb des Gateways. Anschließend werden nacheinander die Node B-Module und das zugehörige PPP-Interface im UMTS Gateway erstellt. Danach werden die Verbindungen und Kanäle zwischen Node B, Gateway und PPP-Interface aufgebaut, bevor die Initialisierungsmethoden der Module aufgerufen werden. Diese Reihenfolge ist zwingend notwendig, da die PPP-Interfaces zum Initialisierungszeitpunkt einen vollständig initialisierten Kanal zu einem anderen Interface benötigen, um Daten senden und empfangen zu können. Zum Schluss wird noch ein Pointer zu jedem Node B im entsprechenden Eintrag in der Node B-Tabelle gespeichert, damit andere Module bei Bedarf direkt darauf zugreifen können.

Im späteren Verlauf der Simulation kommt noch eine zweite Funktion der NodeBTable-Klasse zum Einsatz, die *getReachableNodeBs*-Funktion. Sie wird vom Association Client der Mobilgeräte aufgerufen, als Parameter wird die aktuelle Position des Mobilgeräts übergeben. Die Funktion durchläuft anschließend linear die komplette Node B-Tabelle und überprüft für jeden Eintrag, ob sich das Mobilgerät in Reichweite des aktuellen Node B befindet. Alle Node B, auf die das zutrifft, werden in einem Vektor gespeichert und dann an den Association Client zurückgegeben.

4.3 NodeBPlacement

Die Auswahl, welches Platzierungsmodul zum Einsatz kommen soll, wird über die Konfigurationsdatei getroffen. Dort werden auch die Parameter eingestellt, die das Platzierungsmodul benötigt. Eine Übersicht der Parameter ist in Tabelle 4.2 zu sehen. Bei der zufällig gleichverteilten Platzierung ist der Radius mit dem Modifikator *volatile* angegeben. Das bedeutet, dass der Parameter bei jedem Zugriff erneut aus der Konfigurationsdatei gelesen und nicht zwischengespeichert wird. Wenn der Radius als Zufallsfunktion angegeben wird, erhält man dadurch unterschiedliche Werte für die maximale Reichweite der Node B, da bei jedem Auslesen des Parameters ein neuer Wert generiert wird. Bei der gleichmäßigen Platzierung dagegen wird der Radius nur einmal ausgelesen, da dort alle Node B die gleiche maximale Reichweite haben.

Die Klasse *NodeBPlacement* ist ein Modul-Interface, das die Klassen *RandomNodeBPlacement* und *RegularNodeBPlacement* implementieren. Dafür muss die Funktion *placeNodeBs* implementiert werden, die Node B auf der Simulationsgrundfläche platziert. Der jeweilige Algorithmus zur Platzierung ist im Abschnitt 3.3.3 zu sehen. Durch die Funktion *addNodeB*, die beide Klassen implementieren, wird ein Eintrag für ein platzierten Node B in der NodeBTable erstellt.

4.4 UMTS-Interface

Das UMTS-Interface kommt in den Node B und den Mobilgeräten zum Einsatz. Im INET/MANET-Framework sind bereits mehrere Interfaces implementiert und stehen für

RandomNodeBPlacement		RegularNodeBPlacement	
Parameter	Standardwert	Parameter	Standardwert
volatile double radius	-	*double* radius	-
int numberOfNodeB	-		

Tabelle 4.2: Parameter des Platzierungsmoduls

den Einsatz in den Hosts zur Verfügung. Unter anderem sind Interfaces für Ethernet, PPP oder IEEE 802.11 vorhanden. Allerdings ist kein UMTS-Interface vorhanden, sodass es für diese Arbeit implementiert werden musste.

Während der Initialisierungsphase wird durch den Aufruf der Funktion *registerInterface* ein Eintrag in der InterfaceTable erstellt. Danach ist das Interface bereit, Pakete zu empfangen.

Ein vom Network Layer ankommendes Paket wird in einen UMTS-Frame gekapselt und ohne weitere Verzögerung in Richtung des Ausgangsgates des Hosts weitergeleitet. In den anderen Interfaces des INET/MANET-Frameworks ist es üblich, hier zuerst noch das Paket um die Zeit der theoretischen Übertragungsdauer zu verzögern. Da die Verzögerung in dieser Simulation auf andere Art und Weise modelliert wird, ist dies hier nicht nötig. Sollte allerdings das Ausgangsgate des Hosts nicht verbunden sein, so wird das Paket verworfen. Dies kann passieren, wenn von den höheren Layern ein Paket gesendet wurde, das Mobilgerät allerdings währenddessen die Verbindung zum Node B abgebrochen hatte. Analog dazu ist das Vorgehen bei einem Paket, das vom Eingangsgate des Hosts eintrifft: Der Payload wird aus dem UMTS-Frame entpackt und dann direkt an den Network Layer weitergeleitet.

4.5 Mobile Geräte

In Abbildung 4.2 ist die NED-Datei eines mobilen Geräts zu sehen. Neben den bereits im Entwurfkapitel vorgestellten Komponenten sind hier einige weitere Module vorhanden: Das NotificationBoard ist ein Konzept, das in sämtlichen Hosts im INET/MANET-Framework angewandt wird. Es realisiert ein kleines Publish/Subscribe-System, bei dem Module eines Hosts Ereignisse anmelden, die sie in Zukunft auslösen werden. Ein Beispiel dafür wäre die Zuweisung einer neuen IP-Adresse in einem Interface oder ein Positionsupdate der Mobilitätskomponente. Andere Module können sich wiederum für eben solche Ereignisse registrieren und werden benachrichtigt (es wird die Funktion *receiveChangeNotification* im registrierten Modul aufgerufen), sobald das Ereignis eingetreten ist. Dieses Vorgehen hat den entscheidenden Vorteil, dass Module, die auf bestimmte Veränderung warten, nicht per Polling den aktuellen Stand abfragen müssen, sondern bei einer Veränderung benachrichtigt werden, was deutlich effizienter ist.

In jedem Mobilgerät ist auch eine RoutingTable vorhanden. In ihr werden die Routen gespeichert, die der NetConfigurator in der Initialisierungsphase erstellt hat. In der ebenfalls in jedem Host vorhandenen InterfaceTable werden die vorhandenen Interfaces verwaltet. Neben

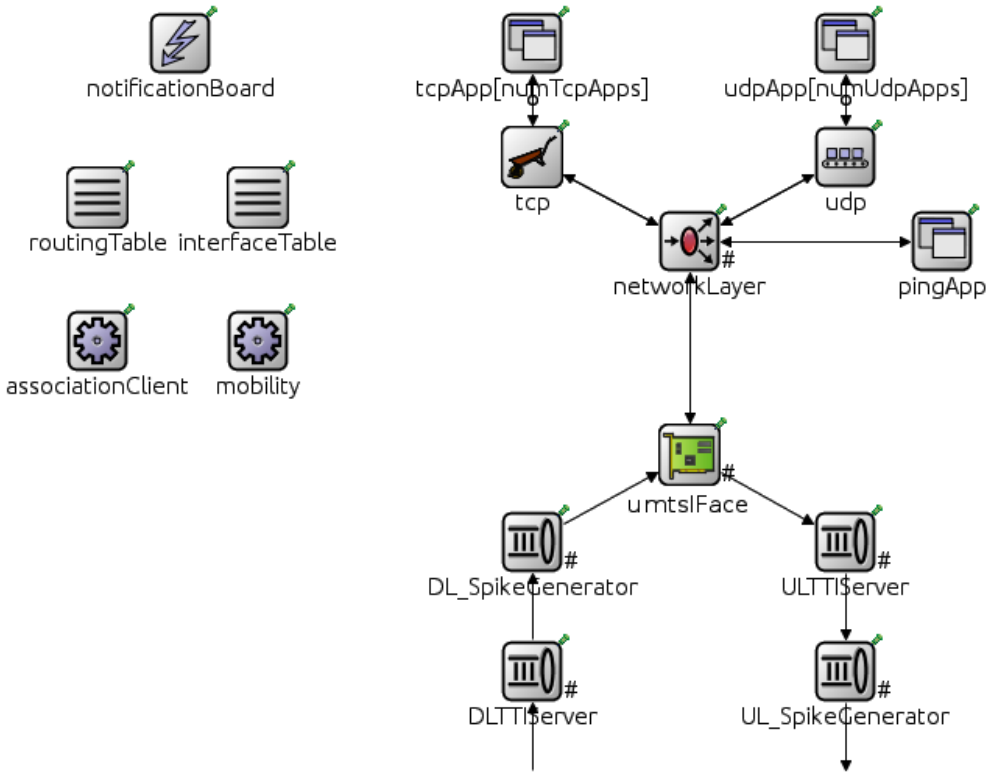


Abbildung 4.2: NED-Datei eines Mobilgeräts

dem Namen werden hier auch spezifische Informationen wie die IP-Adresse oder das Fehlen einer Broadcast-Funktion gespeichert. Unter anderem greift auch hier der NetConfigurator auf die Informationen zu und setzt die IP-Adresse.

Die Layer drei bis fünf des TCP/IP-Stacks sind direkt aus dem INET/MANET-Framework übernommen und werden deshalb hier nicht näher erläutert. Zusätzlich ist noch ein Ping-Modul des INET/MANET-Frameworks vorhanden, mit dem Ping-Anfragen gesendet werden können.

4.5.1 TTI Server

Der TTI Server ist der erste Teil des Verzögerungmodells. Die Verweildauer eines Pakets im TTI Server ist unter anderem abhängig von der aktuell verfügbaren Bandbreite; diese wird vom Bandwidth Assigner des Node B, zu dem das Mobilgerät verbunden ist, durch Aufruf der Funktion *setBandwidth* zugewiesen. Alle eingehende Pakete werden in einer FIFO-Queue gespeichert, die bereits im INET/MANET-Framework enthalten ist. Die Kapazität dieser Eingangsqueue kann über die Konfigurationsdatei festgelegt werden. Sollte ein Paket an einer bereits vollen Eingangsqueue ankommen, so wird nach dem Prinzip des Tail Drop vorgegangen und das Paket verworfen. Für das jeweils erste Paket der Eingangsqueue wird

die Zeit berechnet, die für die Übertragung theoretisch notwendig ist. Es ergibt sich für die Berechnung der Übertragungszeit T_t mit Paketgröße p und verfügbarer Bandbreite b_{max}:

$$T_t = p \ / \ b_{max}$$

Nachdem das Paket die Übertragungszeit an erster Stelle der Eingangsqueue verbracht hat, wird es an eine weitere Queue, die Ausgangsqueue, übergeben. Während sich Pakete in einer der beiden Queues befinden, ist ein Timer aktiv, dessen Laufzeit ein TTI beträgt. Nach Ablauf dieser Zeit wird die Funktion *clearServedQueue* aufgerufen, die alle Pakete, die sich in der Ausgangsqueue befinden und theoretisch bereits übertragen sind, über das Ausgangsgate weitergeleitet. Dies dient zur Modellierung der Taktung der UMTS-Übertragung auf die Zeitspanne eines TTI.

Die Parameter des TTI Server, die über die Konfigurationsdatei eingestellt werden können, sind in Tabelle 4.3 zu sehen.

Parameter	Standardwert
int capacity	50
int TTI_length	-

Tabelle 4.3: Parameter des TTI Servers

4.5.2 Verzögerungsspitzengenerator

Die Verzögerungsspitzengeneratoren stellen den zweiten Teil der Verzögerungsmodellierung dar. Ein Verzögerungsspitzengenerator besteht aus einem Puffer und einem Automaten mit zwei Zuständen. Zu Beginn der Simulation befindet sich der Automat im Loss-Free-Zustand, in dem alle eingehenden Pakete einfach ohne Verzögerung weitergeschickt werden. Es wird die Zeit der Verweildauer in diesem Zustand bestimmt, die aus der Konfigurationsdatei gelesen wird. Es ist vorgesehen, dass dort kein fester Wert, sondern eine Verteilungsfunktion angegeben ist. Deshalb wird bei jedem Erreichen des Zustands der Wert erneut ausgelesen und die Verweildauer bestimmt, weshalb die zugehörigen Parameter den Modifikator *volatile* besitzen, was in Tabelle 4.4 zu sehen ist. Nach Ablauf dieser Zeit wechselt der Automat in seinen zweiten Zustand, den Recovery-Zustand. Dort werden alle ankommenden Pakete im Puffer gespeichert. Es wird erneut die Verweildauer aus der Konfigurationsdatei ausgelesen, die separat von der des Loss-Free-Zustands angegeben werden kann. Beim erneuten Wechsel in den Loss-Free-Zustand werden alle gespeicherten Pakete direkt und ohne weitere Verzögerung über das Ausgangsgate weitergeleitet.

4.5.3 Association Client

Der Association Client ist für das Verbindungsmanagement in einem mobilen Endgerät zuständig. Alle Aktionen, ob Verbindungsaufbau oder Verbindungsabbau, werden durch ihn initiiert. Eine Übersicht der Funktionen des Association Client ist in Algorithmus 4.1 zu

Parameter	Standardwert
volatile double spike_IAT	-
volatile double spike_duration	-
int TTI_length	-

Tabelle 4.4: Parameter des Verzögerungsspitzengenerators

sehen. Der Association Client beginnt seine Aktivität bereits in der Initialisierungsphase. In Stufe eins werden noch diverse Variablen initialisiert und das Modul registriert sich beim Notification Board für Positionsupdates der Mobilitätskomponente.

Sobald die Simulation Initialisierungsstufe sieben erreicht hat, beginnt der Association Client einen Verbindungsversuch durch Aufruf der Funktion *InitConnect*. Dort wird zuerst von der NodeBTable eine Liste eingeholt, die alle von der aktuellen Position des Mobilgeräts erreichbaren Node B in der Umgebung enthält. Diese werden dann nach aufsteigender Entfernung zur Position des Mobilgeräts sortiert, wobei die Node B mit fremder Provider-ID zunächst nicht berücksichtigt und aussortiert werden. Mit dieser sortierten und gefilterten Liste wird nun die Funktion *tryToConnect* aufgerufen, die der Reihe nach die Bandwidth Assigner der Node B in der Liste anfragt, ob eine Verbindung möglich ist. Wenn dies der Fall ist, wird der Association Server des betreffenden Node B mit dem Aufbau der Verbindung beauftragt und der Verbindungsaufbau im Association Client ist abgeschlossen. Wenn die *tryToConnect*-Funktion erfolglos sein sollte, wird sie nochmals mit einer neuen Liste aufgerufen, die dann alle Node B enthält und nicht nur die mit identischer Provider-ID. Sollte auch hier kein Node B gefunden werden, zu dem eine Verbindung möglich ist, so wird der Verbindungsversuch abgebrochen und nach Ablauf eines Timeouts ein neuer Versuch gestartet. Dieses Timeout kann über die Konfigurationsdatei gewählt werden.

Da das Modul des Association Client sich für Positionsupdates beim Notification Board registriert, wird es laufend über die aktuelle Position des Mobilgeräts informiert. Bei jedem Update wird die Funktion *checkNodeBConnection* aufgerufen. Wenn das mobile Endgerät momentan nicht verbunden ist, so wird dort nichts unternommen, da ein neuer Verbindungsversuch bereits durch das Timeout der *InitConnect*-Funktion gesteuert wird. Wenn das Gerät verbunden ist und sich nun durch das Positionsupdate außerhalb der Reichweite des momentan verbundenen Node B befindet, so wird die Verbindung beendet. Befindet sich das Mobilgerät im Optimierungsbereich des Node B, so wird die Funktion *searchBetterNodeB* aufgerufen, die versucht, sich zu einem Node B mit gleicher Provider ID zu verbinden, der sich zugleich näher am Mobilgerät befindet als der, zu dem es gerade verbunden ist. Um einen zu großen Overhead durch das ständige Suchen nach einem näheren Node B zu vermeiden, kann festgelegt werden, dass nur nach einer bestimmten Anzahl an Positionsupdates im Optimierungsbereich nach besseren Node B gesucht werden soll.

Alle vom Association Client verwendeten Parameter sind in Tabelle 4.5 zu sehen.

Algorithmus 4.1 Funktionen des Association Clients zum Verbindungsmanagement

```
procedure initConnect( )
    list ← sortList(reachable Node B with own ProviderID)
    connected ← tryToConnect(list)
    if (connected = FALSE) then
        list ← sortList(all reachable Node B)
        connected ← tryToConnect(list)
    end if
    if (connected = FALSE) then
        schedule connection retry
    end if
end procedure

procedure tryToConnect(NodeBList list)
    success ← FALSE
    while (list has elements ∧ success = FALSE) do
        if (connection to first Node B in list is possible) then
            if (mobile station is connected) then
                initDisconnect( )
            end if
            inform AssociationServer to set up connection
            success ← TRUE
        else
            remove first Node B in list
        end if
    end while
     return success
end procedure

procedure searchBetterNodeB( )
    list ← sortList(reachable nearer Node B with own ProviderID)
    tryToConnect(list)
end procedure

procedure checkNodeBConnection( )
    if (¬ connected) then
        do nothing
    else if (mobile station is out of range of currently connected Node B) then
        disconnect from current Node B
    else if (mobile station is in optimization area ∧ enough position updates) then
        searchBetterNodeB( )
    end if
end procedure
```

Parameter	Standardwert
double reconnectDelay	-
double optimizationRatio	-
int checkAfterXUpdates	-

Tabelle 4.5: Parameter des Association Clients

4.5.4 Mobilität

Um die bereits implementierten Mobilitätsmuster des INET/MANET-Frameworks verwenden zu können, mussten einige Anpassungen an der Simulation vorgenommen werden. So verlangt jedes Mobilitätsmodul nach einem ChannelControl-Modul, dem es nach jedem Positionsupdate die neue Position mitteilt. Außerdem werden dort Pointer zu jedem im System befindlichen Mobilgerät gespeichert und die Funkübertragungen zwischen den Hosts abgewickelt. Da in dieser Simulation die Verbindungen zwischen Node B und Mobilgerät über direkte, verzögerungsfreie Kanäle realisiert wurden und auch die Information über die Erreichbarkeit der Node B über die NodeBTable gewährleistet ist, ist das ChannelControl-Modul überflüssig. Dennoch muss es im System vorhanden sein, um die Kompatibilität zu wahren. Die Mobilitätsmodule senden also trotzdem jedes Positionsupdate an dieses Modul, allerdings ohne jeden weiteren Nutzen für UMTS. Dies bedeutet zwar einen gewissen Overhead, dafür können allerdings alle Bewegungsmuster des INET/MANET-Frameworks ohne Modifikation mit der UMTS-Komponente verwendet werden.

4.6 Node B

Der Aufbau der NED-Datei eines Node B ist mit dem der Abbildung 3.3 nahezu identisch; Dort nicht abgebildet sind das NotificationBoard und die RoutingTable.

Wenn der Node B ein anderes Verbindungsprotokoll als PPP nutzen soll, um sich mit dem UMTS Gateway zu verbinden, so muss eine neue NED-Datei für den Node B erstellt werden, in der das PPP-Interface ersetzt wird. Außerdem muss in der Funktion *addNodeBsToNetwork* der NodeBTable die Erstellung der Interfaces im Gateway so geändert werden, dass dort ebenfalls das geänderte Interface zum Einsatz kommt.

4.6.1 Association Server

Der Association Server setzt die vom Association Client initiierten Verbindungsstatusänderungen um. Er erstellt bei Verbindungsaufbau einen Datenübertragungskanal zwischen Mobilgerät und Node B, macht dies bei Verbindungsabbau wieder rückgängig und ist für die Korrektheit der Einträge der ForwardTable und der RoutingTable im Node B, im UMTS Gateway und in der Internetkomponente zuständig.

Wenn der Association Client eines Mobilgeräts den Association Server mit dem Aufbau einer Verbindung beauftragt, wird die Funktion *connectNewMobileStation* aufgerufen. Dort wird zunächst dem Mobilgerät ein freies Inout-Gate zugewiesen. Falls kein freies Gate verfügbar ist, wird ein neues Gate angelegt. Anschließend wird das mobile Gerät durch einen verzögerungsfreien Übertragungskanal über das eben bestimmte Gate mit dem Node B verbunden. Innerhalb des Node B-Moduls muss der Downstream-Teil des Gates noch mit dem Packet Distributor und der Upstream-Teil mit dem Uplink-Verzögerungsspitzengenerator verbunden werden. Danach ist der physikalische Verbindungsaufbau abgeschlossen. Nun wird ein Eintrag in der ForwardTable des Node B angelegt, der die IP-Adresse des neu verbundenen Mobilgeräts auf den Index des Gates abbildet. Ebenso werden die Routingtabellen im UMTS Gateway und in der Internetkomponente aktualisiert, indem dort eine neuer Eintrag für die zu dem Mobilgerät gehörende IP-Adresse angelegt wird. Zuletzt wird noch der Bandwidth Assigner über das neue Mobilgerät informiert, damit dieser die Bandbreite neu verteilen kann.

Der Abbruch einer Verbindung läuft dann analog zum Verbindungsaufbau: zuerst werden die Übertragungskanäle zwischen Mobilgerät und Node B abgebaut, ebenso die Verbindung innerhalb des Node B zwischen Gate und Packet Distributor beziehungsweise Uplink-Verzögerungsspitzengenerator. Außerdem müssen wieder die Einträge in der ForwardTable und den RoutingTables gelöscht werden.

Neben den Routinen zum Verbindungsauf- und Abbau besitzt der Association Server die Funktion *getConnectedMobileStations*, die eine Liste der momentan verbundenen Mobilgeräte zurückgibt. Diese wird vom Bandwidth Assigner genutzt, um bei einer Neuverteilung der Bandbreite auf die verbundenen mobilen Geräte zugreifen zu können.

4.6.2 Bandwidth Assigner

Der Bandwidth Assigner verteilt nach dem im Abschnitt 3.3.2 beschriebenen Algorithmus die verfügbare Bandbreite auf die verbundenen Geräte eines Node B. Die Funktion *canAddMobileStation* führt die Berechnungen der Gleichungen 3.1 und 3.2 aus und überprüft damit, ob sich ein weiteres Gerät zum Node B verbinden kann. Außerdem implementieren die Funktionen *calculateUplinkBandwidth* und *calculateDownlinkBandwidth* die Verteilung der verfügbaren Bandbreite auf die verbundenen Geräte. Durch die Funktion *updateBW* wird die berechnete Bandbreite pro Gerät diesen dann zugewiesen.

Alle vom Bandwidth Assigner verwendeten Parameter sind in Tabelle 4.6 zu sehen.

4.6.3 ForwardTable

Die ForwardTable wird in den Node B zum Verwalten der verbundenen Mobilgeräte verwendet. Im Gegensatz zu einer RoutingTable, die auf dem Network Layer IP-Adressen auf Interfaces abbildet, ist die ForwardTable auf dem Link Layer für die Abbildung von IP-Adressen auf Gates zuständig. Jeder Eintrag der Tabelle besteht aus einer IP-Adresse,

Parameter	Standardwert
int UL_Bandwidth	-
int DL_Bandwidth	-
int guaranteed_UL_Bandwidth	-
int guaranteed_DL_Bandwidth	-
int maximum_UL_Bandwidth	-
int maximum_DL_Bandwidth	-
bool connectIfFull	-

Tabelle 4.6: Parameter des Bandwidth Assigner

der ein bestimmtes Gate zugeordnet wird. Somit haben alle Module, die auf die jeweilige ForwardTable Zugriff haben, die Information, über welches Gate eine bestimmte IP-Adresse erreichbar ist. Sie dient dazu, ein Paket, das an eine bestimmte Zieladresse geschickt werden soll, über das entsprechende Gate an das Zielgerät weiterzuleiten.

Die Einträge werden in einer Map gespeichert, die von der Standard-C++-Bibliothek zur Verfügung gestellt wird. Die Klasse der ForwardTable besitzt drei Methoden, um die Einträge der Map zu modifizieren: eine zum Hinzufügen eines Mappings von IP-Adresse auf eine Gatenummer, eine zur Abfrage, welches Gate für das Erreichen einer bestimmten IP-Adresse verwendet werden muss und eine zum Löschen eines Eintrags der Tabelle.

4.6.4 ConstantDelay

Das Modul für die konstante Verzögerung stellt die letzte Komponente des Verzögerungsmodells dar. Es sendet jedes ankommende Paket mit einer konstanten Verzögerung nochmals an sich selbst. Beim zweiten Eintreffen des Pakets wird es über das Ausgangsgate weitergeleitet. Bei der Initialisierung wird die Zeitdauer aus der Konfigurationsdatei gelesen und anschließend jedes ankommende Paket um eben diese Zeit verzögert.

Parameter	Standardwert
double delay	-

Tabelle 4.7: Parameter des ConstantDelay

4.6.5 PacketDistributor

Der PacketDistributor dient zum Routing der Pakete im Node B an eines der verbundenen Mobilgeräte. Ankommende Pakete wurden vom Network Layer aufgrund eines Eintrags in der RoutingTable an ihn weitergeleitet und nun muss noch das Paket über das richtige Gate an das Gerät mit der Zieladresse geschickt werden. Deshalb besteht die einzige Aufgabe des PacketDistributors darin, das zu der Zieladresse gehörige Gate von der ForwardTable zu erfragen und anschließend das Paket dorthin weiterzuleiten. Wenn kein Eintrag zu der Zieladresse in der ForwardTable existiert, wird das Paket verworfen. Dieser Fall kann zum

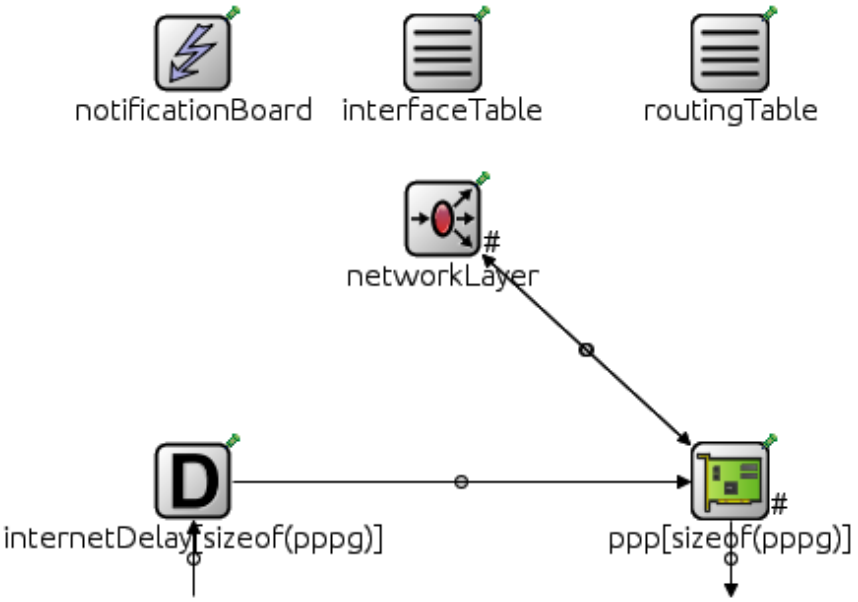


Abbildung 4.3: Übersicht der NED-Datei der Internetkomponente

Beispiel auftreten, wenn das Mobilgerät mit der Zieladresse die Verbindung zum Node B beendet, während sich das Paket auf dem Weg vom Network Layer zum PacketDistributor befindet.

4.7 UMTS Gateway

Der UMTS Gateway ist mit allen Node B im System verbunden und hat ein Upstream-Gate zum Internet. Er dient als Verbindungsstück zwischen dem UMTS-Bereich des Systems und dem Internet. Um die Pakete richtig weiterzuleiten, wurde der StandardRouter des INET/MANET-Frameworks als Vorlage genommen und geringfügig geändert. So besitzt der Gateway bei Simulationsstart genau ein PPP-Interface, das mit einem PPP-Interface der Internetkomponente verbunden. Durch die *addNodeBsToNetwork*-Funktion der Node B Table wird pro Node B im System ein weiteres PPP-Interface erstellt. Mit Hilfe des Network Layers und der RoutingTable wird dann jedes ankommende Paket über das entsprechende Interface in Richtung Zielgerät weitergeleitet. Die Übertragungskanäle zwischen UMTS Gateway, Node B und der Internetkomponente besitzen im Gegensatz zu den Kanälen zwischen Mobilgeräten und Node B eine festgelegte Bandbreite und eine Verzögerung, die über die Konfigurationsdatei festgelegt werden können.

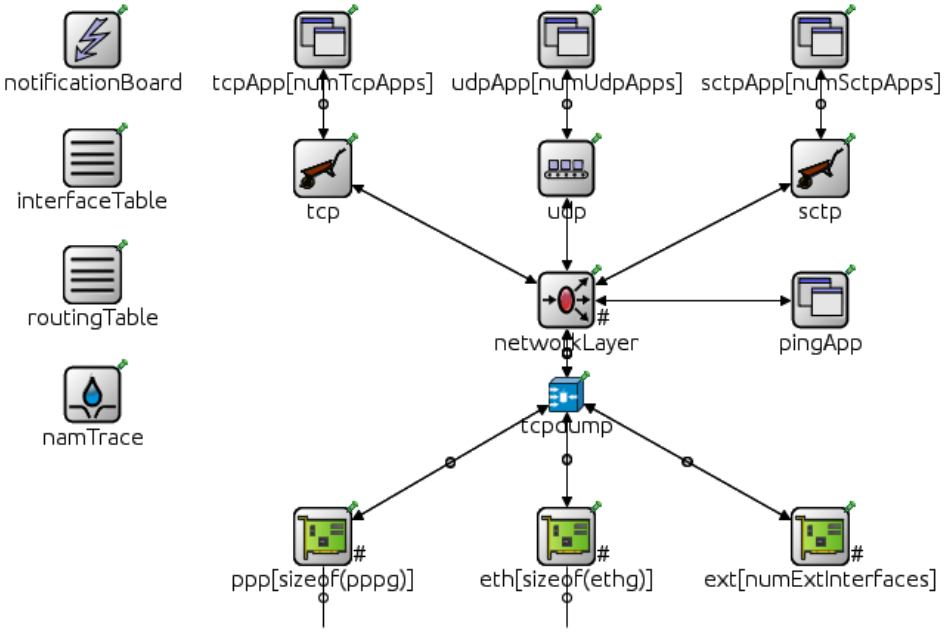


Abbildung 4.4: Übersicht der NED-Datei eines stationären Servers

4.8 Internet

Die Internetkomponente soll das Verhalten des realen Internets nachbilden. Allerdings wurde hier ein sehr einfaches Modell verwendet, da es nicht das Hauptaugenmerk der Arbeit war. Es ist mit dem UMTS Gateway über ein PPP-Interface verbunden. Zusätzlich können stationäre Hosts per PPP-Verbindung angeschlossen werden. Wie in Abbildung 4.3 zu sehen ist, besteht das Internetmodul aus einem ConstantDelay-Modul pro PPP-Gate und einem dazugehörigen PPP-Interface. Die Funktionsweise des Verzögerungsmoduls ist identisch mit dem, das in den Node B verwendet wird. Auch hier kann über die Konfigurationsdatei die Verzögerung bestimmt werden. Das erste PPP-Interface wird zu Beginn mit dem UMTS Gateway verbunden, die restlichen sind mit den stationären Hosts verbunden. Jedes ankommende Paket, egal ob von UMTS Gateway oder von stationären Servern, passiert zunächst das zum Gate gehörende Verzögerungsmodul, bis es über das PPP-Interface zu dem Network Layer gelangt. Dort wird entsprechend der Routingtabelleneinträge das Paket an das passende Interface weitergeleitet und verlässt danach wieder die Internetkomponente.

Wenn die Internetkomponente durch ein anderes Modell ersetzt werden soll, so ist darauf zu achten, dass dieses nach wie vor ein Interface besitzt, um die Verbindung mit dem UMTS Gateway herzustellen und dass es als erstes Interface nach dem Loopback-Interface in der InterfaceTable geführt wird.

4.9 Stationäre Server

Für die Evalution wurden stationäre Server benötigt. Dazu wurde der StandardHost des INET/MANET-Frameworks benutzt, da er die gewünschten Funktionen beherrscht. Eine Übersicht der NED-Datei ist in Abbildung 4.4 zu sehen. Es sind auch Module vorhanden, die bei der Evaluation nicht verwendet wurden.

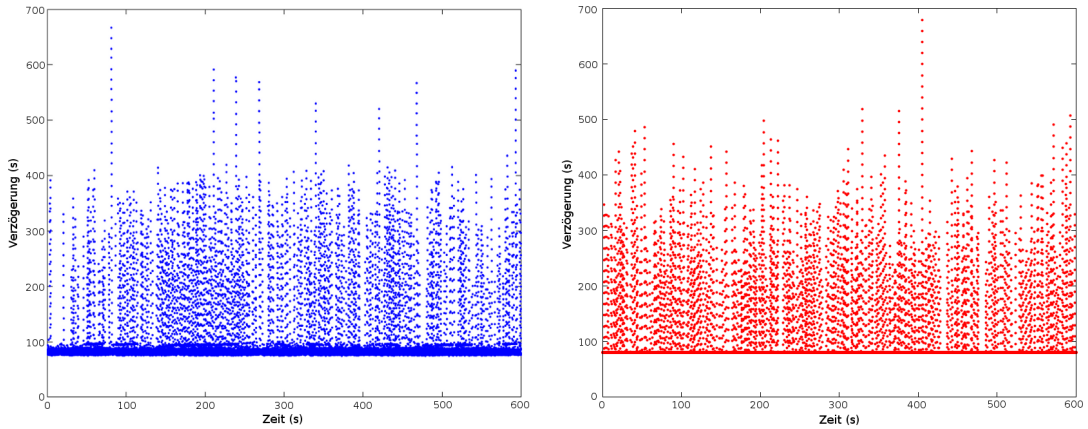
Der stationäre Server implementiert einen TCP/IP-Stack, wobei im Link Layer lediglich ein PPP-Interface zur Verbindung mit dem Internet genutzt wurde. Daneben gibt es auch wieder die Hilfsmodule wie das NotificationBoard oder die InterfaceTable. Die verwendeten Programme des Application Layers können ebenso wie die des Mobilgeräts über die Konfigurationsdatei bestimmt werden. Die stationären Server sind ebenso wie der UMTS Gateway über einen Übertragungskanal mit festgelegter Bandbreite und Verzögerung mit dem Internet verbunden.

5 Evaluation

Um die Implementierung des entwickelten Modells zu validieren, wurden Testläufe durchgeführt, die die Messungen aus [CGGPC06] und [tan2008] nachstellen. Die daraus erlangten Messergebnisse werden dann mit den empirischen Messergebnissen der beiden Arbeiten verglichen.

Neben den Vergleichstests wurde auch Skalierbarkeitsuntersuchungen durchgeführt. Dazu wurden verschiedene Testszenarien erstellt. Die Simulation eines Szenarios wurde beendet, nachdem sie eine Stunde Prozessorzeit in Anspruch genommen hatte. Anschließend wurden die erreichten Simulationszeiten notiert, anhand derer dann die Auswirkungen von einzelnen Simulationsparametern auf die Skalierbarkeit untersucht wurden.

Die Messreihen der beiden Arbeiten, die als Grundlage der Modellentwicklung dienten, wurden in Spanien und in China aufgezeichnet. Um die Resultate zu verifizieren und um zu überprüfen, ob die Ergebnisse auch auf UMTS-Provider in Deutschland zutreffen, wurde eine eigene Messreihe aufgestellt. Unter verschiedenen Bedingungen wurde mit einem UMTS-fähigen Smartphone die Verzögerung und Bandbreite gemessen.



(a) Originale Messreihe (Quelle: [CGGPC06])

(b) Simulierte Messreihe

Abbildung 5.1: Downlink-Verzögerung

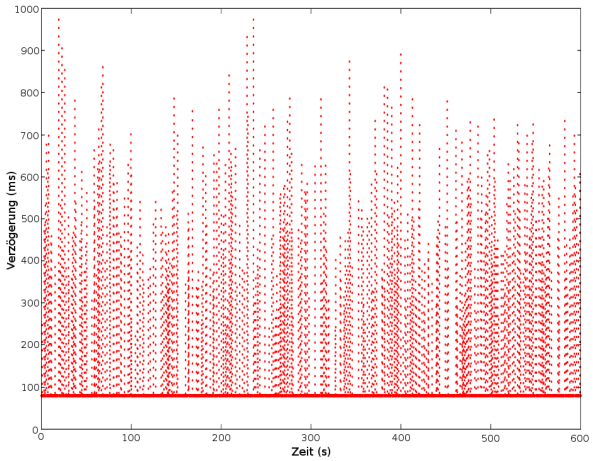


Abbildung 5.2: Uplink-Verzögerung der simulierten Messreihe

5.1 Verzögerungsverhalten

Um das Verzögerungsverhalten der erstellten UMTS-Simulation zu testen, wird der Versuchsaufbau aus [CGGPC06] nachgestellt (siehe Kapitel 2.2). Dazu wird eine sehr kleine Simulationsgrundfläche gewählt, auf der dann genau ein Node B platziert wird. Anschließend wird ein mobiles Gerät in Reichweite des Node B hinzugefügt, das sich durch Verwendung der Null-Mobilität nicht von seiner Startposition entfernt. Außerdem wird ein stationärer Server mit dem Internet verbunden. Der Übertragungskanal zwischen Server und Internet wird mit 2 ms Verzögerung bei einer Bandbreite von 10 MBit/s festgelegt, der Kanal zwischen Internet und UMTS Gateway als auch der zwischen UMTS Gateway und Node B mit 1 ms Verzögerung bei einer maximalen Bandbreite von 100 MBit/s. Auf dem Mobilgerät und dem Server wird jeweils eine UDP-Anwendung ausgeführt. Abhängig vom Ziel der Messung, ob nun der Downlink oder der Uplink untersucht werden soll, wird auf einem der beiden Hosts ein Paketgenerator gestartet, der dann im Abstand von 20 ms Pakete mit 40 Byte Größe sendet. Die Simulation wird für eine Zeitspanne von 600 Sekunden ausgeführt, da dies dem Ausschnitt der Schaubilder in [CGGPC06] entspricht. Es müssen einige Parameter bestimmt werden: Die Verteilung der Zeitdauer zwischen zwei Verzögerungsspitzen und die der Dauer einer Spitze wurde aus dem vorgeschlagenen Modell übernommen. Dazu wurden die Lognormal-Verteilung, die exponentielle Verteilung und die negative Binomialverteilung verwendet. Die restlichen verwendeten Parameter sind in Tabelle 5.1 gelistet.

Das Resultat der Downlink-Simulation ist in Abbildung 5.1 zu sehen. Der Median aller Verzögerungen beträgt 80 ms, der Durchschnitt 98 ms. Dies ist vergleichbar mit den in [CGGPC06] gemessenen Werte, die zwischen 72 ms und 83 ms für den Median und 94 ms bis 108 ms für den Durchschnitt betragen. Die Verzögerungsspitzen sind ebenfalls in der gleichen Größenordnung wie die der empirischen Messung. Zudem beträgt der Abstand der einzelnen Pakete einer Spitze 20 ms, was wie gefordert dem Sendeintervall der Paketquelle entspricht.

Parameter	Wert
Downlink TTI	10 ms
Uplink TTI	20 ms
Zeit zwischen zwei Verzögerungsspitzen im Downlink	lognormal(6.8472, 0.69133)s
Zeit zwischen zwei Verzögerungsspitzen im Uplink	exponential(1330.5004)s
Dauer einer Verzögerungsspitze im Downlink	negbinomial(12.0654, 0.36057)s
Dauer einer Verzögerungsspitze im Uplink	negbinomial(12.0654, 0.36057)s
Konstante Downlink-Verzögerung im Node B	46 ms
Konstante Uplink-Verzögerung im Node B	36 ms
Konstante Verzögerung des Internets	20 ms

Tabelle 5.1: Verwendete Simulationsparameter

Das Resultat der Uplink-Simulation ist in Abbildung 5.2 zu sehen. Die Werte für den Uplink sind vergleichbar mit denen des Downlinks, was ebenso in der originalen Messung beobachtet wurde.

Die Verzögerung eines Pakets, das nicht Teil einer Verzögerungsspitze ist und somit dem Großteil aller Pakete entspricht, setzt sich aus einzelnen Teilverzögerungen zusammen. Die Auflistung aller Teilverzögerungen für ein Paket in Uplink- als auch in Downlink-Richtung, das in der beschriebenen Simulation gesendet wurde, ist in Tabelle 5.2 zu sehen. Dabei wird die Zeit für die Übertragung eines Pakets über einen Kanal mit festgelegter Bandbreite nicht berücksichtigt, da diese weniger als 1 ms beträgt. Die gesamte konstante Verzögerung eines Downlink-Pakets beträgt 70 ms, die eines Uplink-Pakets 60 ms. Dies ist genau der Bereich, der von Cano-Garcia et al. vorgeschlagen wurde. Die Werte für die konstante Verzögerung in den Node B liegen unter denen des vorgeschlagenen Modells. Die Internetkomponente dieser Simulation verzögert ebenfalls jedes Paket um eine konstante Zeit. Da diese allerdings nicht Teil des von Cano-Garcia et al. vorgeschlagenen Verzögerungsmodells ist, ist muss der Wert der konstanten Verzögerung in den Node B entsprechend verringert werden. Sollte zum Beispiel die Implementierung des Internets geändert werden und mit ihr der Wert für die konstante Verzögerung, so ist darauf zu achten, dass der Wert für die konstante Verzögerung im Node B entsprechend angepasst wird.

Die eben vorgestellten Parameterwerte sind auch in den folgenden Simulationen für die Verzögerungsmodellierung verwendet worden.

5.2 Bandbreitenzuteilung

Die Bandbreitenverteilung wurde den empirischen Messdaten von Tan et al. [TLL08] nachgebildet. Um die Korrektheit der Bandbreitenzuweisung der erstellten Simulation zu testen, wurde ein Versuchsaufbau erstellt, der dem aus [TLL08] entspricht. Auf der Simulationsfläche befinden sich drei stationäre Server und drei Mobilgeräte, von denen zwei gleich zu Beginn der Simulation mit dem einzigen Node B verbunden sind. Das dritte mobile Gerät bewegt sich nach 200 Simulationssekunden in die Reichweite des Node B und stellt

Downlink		Uplink	
Ort	Wert	Ort	Wert
Kanalverzögerung Server → Internet	2 ms	Verzögerung TTI Server	20 ms
Konstante Verzögerung Internet	20 ms	Konstante Verzögerung Node B	36 ms
Kanalverzögerung Internet → Gateway	1 ms	Kanalverzögerung Node B → Gateway	1 ms
Kanalverzögerung Gateway → Node B	1 ms	Kanalverzögerung Gateway → Internet	1 ms
Konstante Verzögerung Node B	46 ms	Konstante Verzögerung Internet	20 ms
Verzögerung TTI Server	10 ms	Kanalverzögerung Internet → Server	2 ms
Gesamtwert	80 ms	Gesamtwert	80 ms

Tabelle 5.2: Übersicht der Teilverzögerungen eines 40-Byte-Pakets

ebenfalls eine Verbindung her. Alle Mobilgeräte bauen bei bestehender UMTS-Verbindung TCP-Verbindungen zu einem der Server auf und starten den Download einer Datei, die groß genug ist, um den Downlink-Kanal für den Rest der Simulation auszulasten. Es stellte sich heraus, dass fünf TCP-Verbindungen notwendig sind, um den Downlink auszulasten, was konsistent mit den Angaben in [TLL08] ist, wo die Anzahl mit "mehreren TCP-Verbindungen" beziffert wird. Nach Ablauf von 450 Simulationssekunden bewegt sich das dritte Mobilgerät wieder aus der Reichweite des Node B heraus und bricht den Download ab. Die beiden verbliebenen Mobilgeräte führen den Download fort, bis die Simulation nach 600 Simulationssekunden endet.

Der Downlink-Durchsatz der einzelnen Mobilgeräte sowie der Gesamtdurchsatz des Downlink-Kanals des Node B sind in Abbildung 5.3 zu sehen. Jeder Node B hatte in dieser Simulation eine Bandbreite von 850 kBit/s zur Verfügung, die auf die angeschlossenen Mobilgeräte verteilt werden konnte, wobei keine Bandbreitengarantien vergeben wurden. Zu Beginn der Simulation konnte an beide angeschlossenen Mobilgeräte die maximale Downlinkbandbreite von 384 kBit/s vergeben werden. Nachdem sich das dritte mobile Gerät bei Sekunde 200 verbunden hatte, wurde erneut die Bandbreite gleichmäßig verteilt, wodurch jedes Gerät eine Bandbreite von rund 283 kBit/s erhalten hat, was den leichten Rückgang des Durchsatzes der ersten beiden Mobilgeräte erklärt. Gleichzeitig steigt aber auch der Gesamtdurchsatz des Downlinks des Node B auf seinen maximalen Wert an. Nach Verbindungsabbruch des dritten Geräts erreicht der Durchsatz der beiden ersten Mobilgeräte den gleichen Wert wie zu Beginn der Simulation und auch der Gesamtdurchsatz geht wieder leicht zurück.

Wie in den empirischen Vergleichsdaten kann bei Auslastung des Downlinks ein Anstieg der Paketverzögerungszeiten beobachtet werden, die deutlich über den Zeiten liegen, die im unbelasteten Zustand erreicht werden (siehe Abbildung 5.4). Während des Downloads wird gleichzeitig alle 100 ms eine Ping-Anfrage an den Download-Server gesendet. Die Antwort erreicht die Mobilgeräte nach einer Zeitspanne von 0,3 s bis 1,5 s. Die empirischen Werte lagen ebenfalls in diesem Bereich, obwohl dort Verzögerungsspitzen von mehreren Sekunden auftraten, die hier in dieser Simulation nicht beobachtet werden können. Außerdem fällt auf, dass bei höherer Auslastung des Downlinks des Node B die Antwortzeiten der Ping-Anfragen ebenfalls steigen. Dies ist zu beobachten, als sich nach 200 Sekunden das dritte Mobilgerät verbindet und den Downlink des Node B voll auslastet. Grund dafür ist der höhere Füllstand

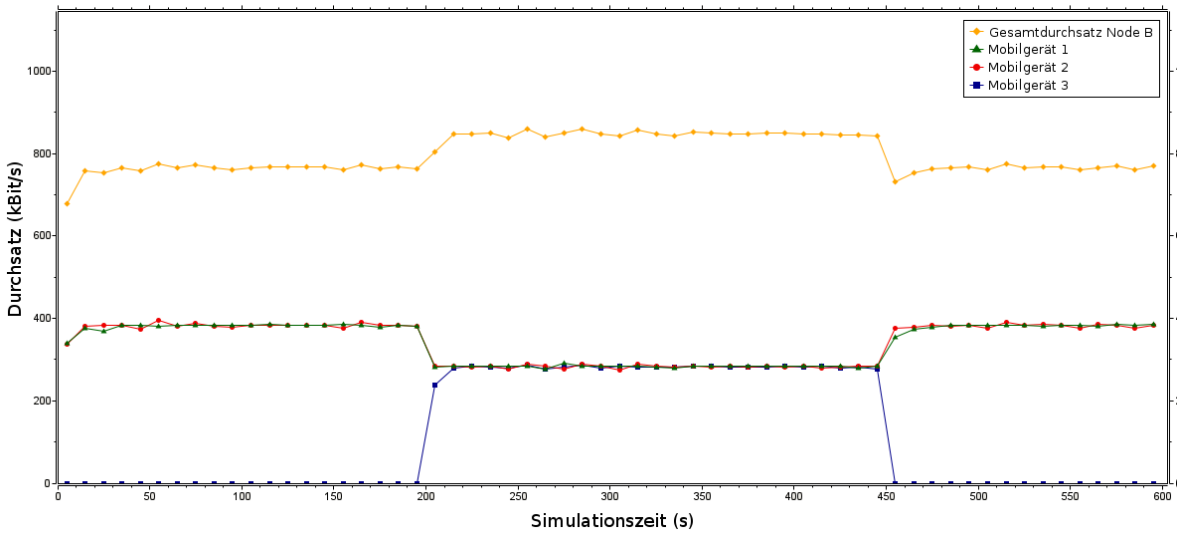


Abbildung 5.3: Durchsatz des Downlink-Kanals bei 3 Mobilgeräten ohne Bandbreitengarantien

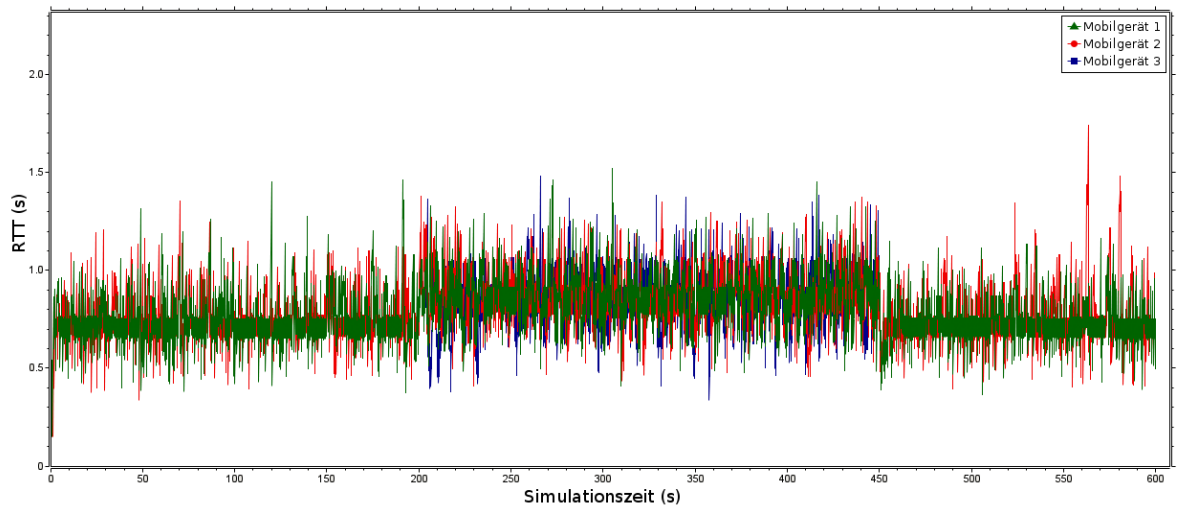


Abbildung 5.4: RTT des Downlink-Kanals

der FIFO-Schlange im UMTS-Interface des Node B, was eine höhere Wartezeit der einzelnen Pakete in der Schlange bedeutet. Der Bereich der Ping-Zeiten ist sehr von der gewählten Kapazität der TTI Server in den Mobilgeräten abhängig. Umso niedriger diese gewählt wird, desto geringer ist die Zeit, die die Pakete bei hoher Auslastung des Kanals in der Schlange verbringen müssen. Dies erhöht zwar ebenfalls die Anzahl der Pakete, die aufgrund der erreichten maximalen Kapazität verworfen werden müssen. Jedoch ist dies kein Problem für die Messung, da bei den verworfenen Paketen keine Antwort des Servers erfolgt. Analog dazu erhöhen sich die Ping-Zeiten bei größerer Kapazität der TTI Server. Es stellte sich

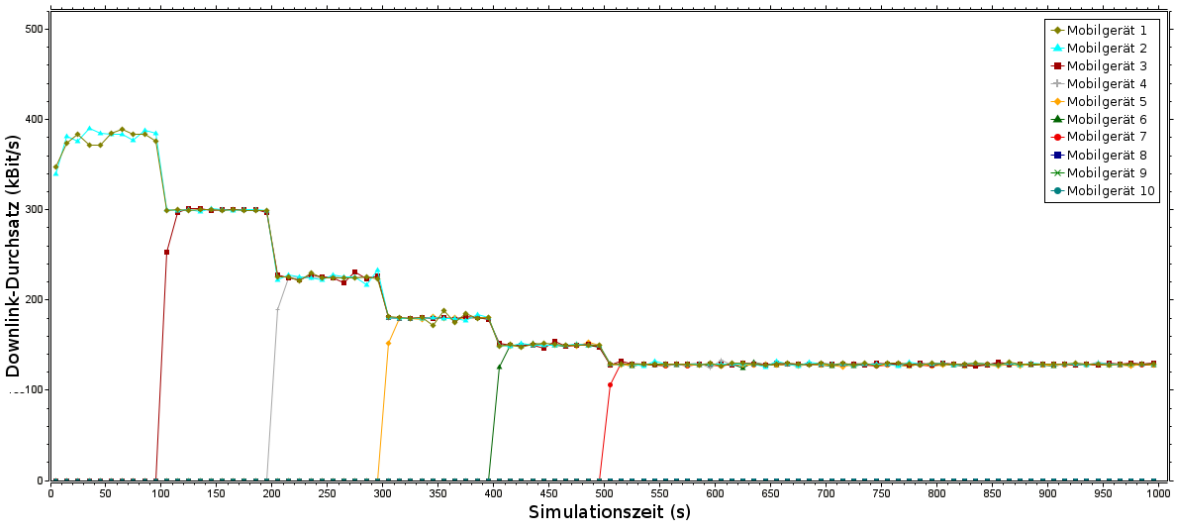


Abbildung 5.5: Durchsatz des Downlink-Kanals bei 10 Mobilgeräten mit Bandbreitengarantien

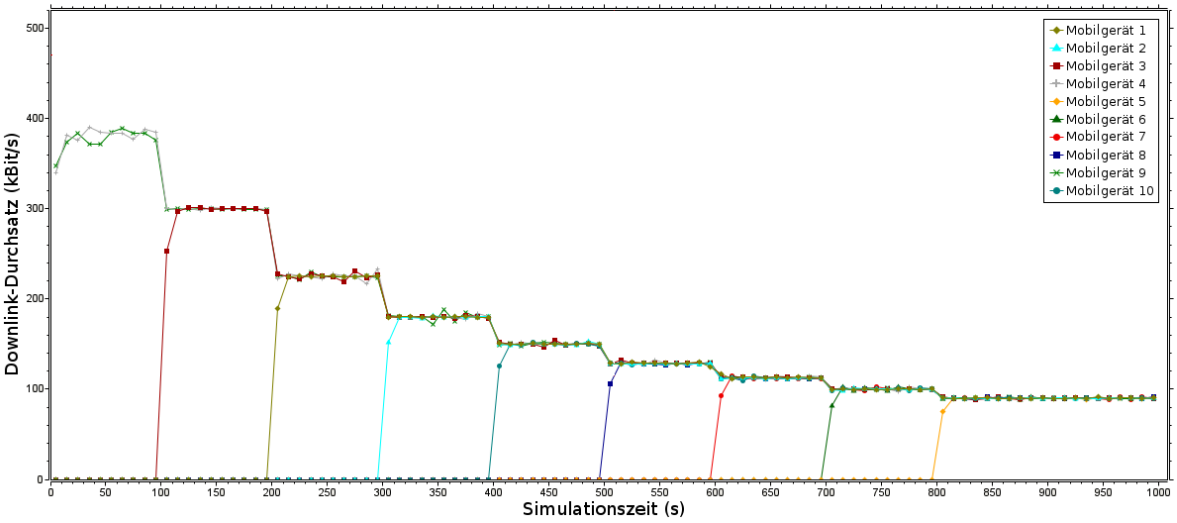


Abbildung 5.6: Durchsatz des Downlink-Kanals bei 10 Mobilgeräten ohne Bandbreitengarantien

heraus, dass bei einer Kapazität von 100 Paketen der Bereich der Ping-Zeiten mit dem der empirischen Vergleichsdaten übereinstimmt.

Nach der Simulation ohne Bandbreitengarantien für die verbundenen Mobilgeräte wurde eine zweite Simulation durchgeführt, bei der einem Node B 900 kBit/s Downlinkbandbreite zur Verfügung steht und jedem verbundenem Mobilgerät eine Mindestbandbreite im Downlink von 125 kBit/s garantiert wurde. Der Simulationsaufbau ist der gleiche wie der der Verzögerungsuntersuchung in Abschnitt 5.1, allerdings diesmal mit 10 Mobilgeräten, von

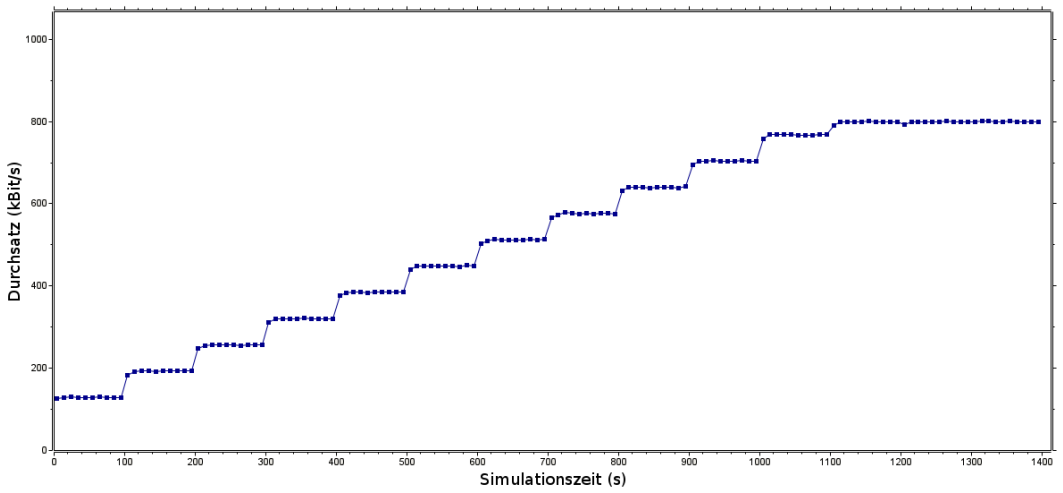


Abbildung 5.7: Gesamtdurchsatz des Uplink-Kanals bei 14 Mobilgeräten ohne Bandbreitengarantien

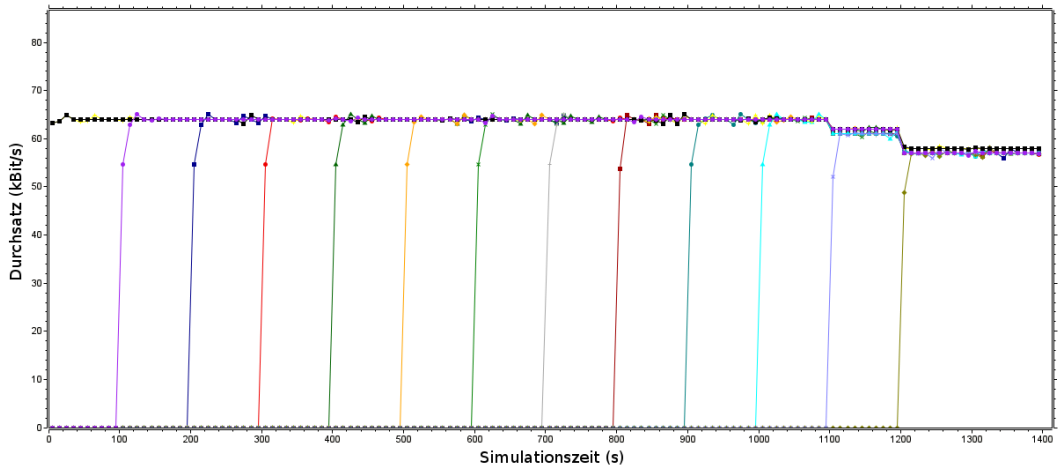


Abbildung 5.8: Durchsatz des Uplink-Kanals der Mobilgeräte bei 14 Mobilgeräten ohne Bandbreitengarantien

denen zwei zu Beginn mit dem Node B verbunden sind. Im Abstand von 100 Sekunden bewegt sich jeweils ein weiteres Gerät in die Reichweite des Node B und verbindet sich mit ihm. Eine Übersicht über den Durchsatz des Downlinks der einzelnen Geräte ist in Abbildung 5.5 zu sehen. Anfangs kann die Bandbreite noch gleichmäßig aufgeteilt werden, was die absteigende Stufenform der einzelnen Durchsatzkurven begründet. Wenn sich das achte Mobilgerät verbindet, reicht die Gesamtbandbreite des Node B von 900 kBit/s nicht mehr aus, um jedem Mobilgerät 125 kBit/s zuzuweisen, weshalb dem zuletzt verbundenem Gerät keine Downlinkbandbreite zugewiesen wird. Dies ist der Grund, warum sich nach 500 Sekunden der Downlink-Durchsatz der Mobilgeräte trotz neuer Verbindungen zum Node B nicht mehr ändert. Zum Vergleich ist in Abbildung 5.6 der gleiche Versuch zu sehen, wobei

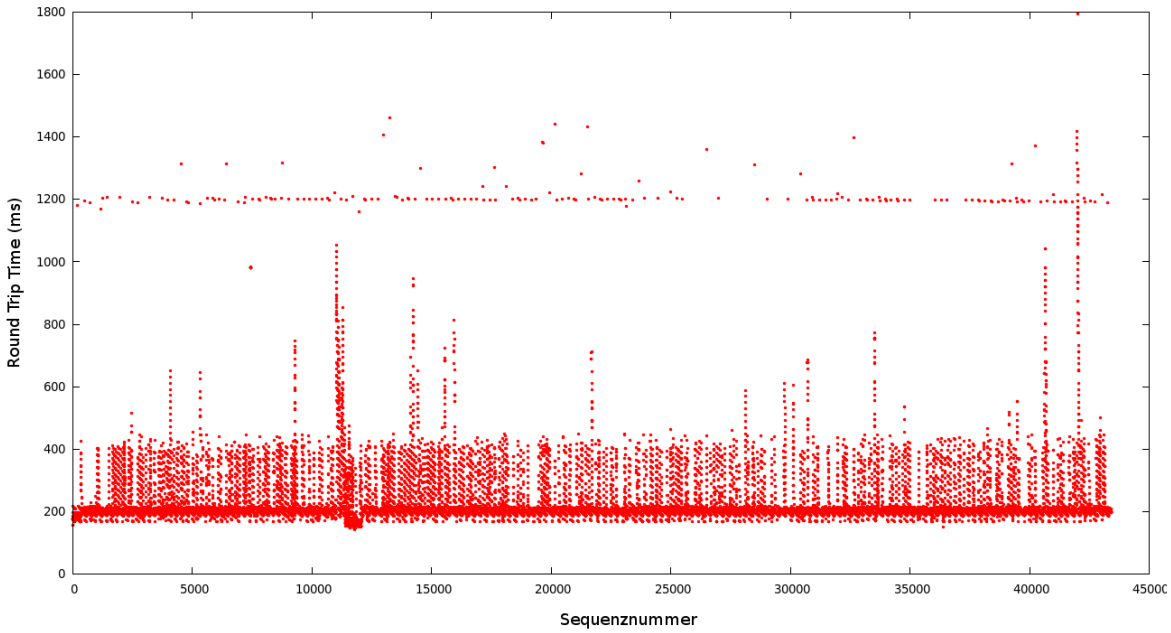


Abbildung 5.9: RTT der Ping-Anfragen an www.google.de

dort keine Bandbreitengarantien vergeben wurden. Darum kann auch nach Verbinden des achten Mobilgeräts weiter allen Geräten Downlinkbandbreite zugewiesen werden und die Stufenform des Durchsatzes wird fortgesetzt.

Bisher wurde nur der Downlink betrachtet. Jedoch gelten die für den Downlink getroffene Aussagen und Verhaltensweisen ebenso für den Uplink, wie in den Abbildungen 5.7 und 5.8 zu sehen ist.

5.3 Eigene Messreihe

Die empirischen Daten aus den Arbeiten [CGGPC06] und [TLL08] wurden in Spanien und in China aufgezeichnet. Um zu überprüfen, ob Mobilfunknetze in Deutschland die gleichen Eigenschaften aufweisen, wurden eigene Messungen durchgeführt. Dazu wurde ein UMTS-fähiges HTC Hero Smartphone [her] per USB-Tethering an einen Laptop angeschlossen. Anschließend wurden in einer ersten Messung Ping-Anfragen im Abstand von 20 ms an die Webseite www.google.de gesendet und die Antwortzeiten aufgezeichnet. Das Ergebnis der Messung ist in Abbildung 5.9 zu sehen. Die meisten Pakete haben eine Verzögerung von 180 ms bis 200 ms. Dies entspricht in etwa der Addition der von Cano-Garcia et al. [CGGPC06] gemessenen Werte für den Uplink und den Downlink. Zudem sind ebenfalls Verzögerungsspitzen zu beobachten, deren Dauer zumeist rund 400 ms beträgt. Außerdem haben vereinzelte Pakete eine hohe Verzögerung von rund 1200 ms. Die Homogenität der Dauer der Verzögerungsspitzen und die vereinzelten hohen Paketverzögerungen decken

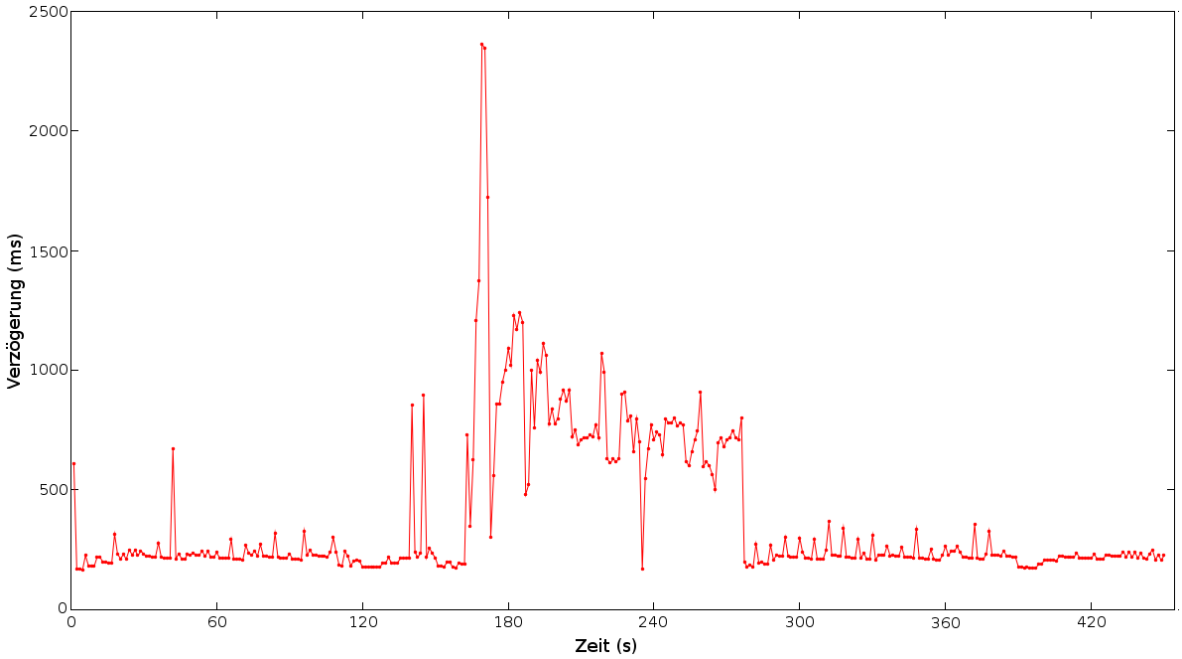


Abbildung 5.10: RTT bei gleichzeitigem Download

sich nicht mit den Beobachtungen von Cano-Garcia et al., jedoch ist der Versuchsaufbau in dieser Arbeit auch abgewandelt und kann somit nur begrenzt mit den Messwerten aus [CGGPC06] verglichen werden.

In einer zweiten Messung diente der Versuchsaufbau aus [TLL08] als Grundlage. Es wurde ein Dateidownload per HTTP gestartet, während gleichzeitig Ping-Anfragen mit einem Abstand von 20 ms an den Download-Server gesendet wurden. Die Resultate der Ping-Messung sind in Abbildung 5.10 zu sehen. Zu Beginn der Messung, als der Download noch nicht gestartet wurde, liegen die RTT wie im ersten Versuch bei rund 200 ms. Nach Beginn des Downloads steigen die RTT und schwanken zwischen 0,3 s und 1,3 s. Dies ist das gleiche Verhalten, das auch von Tan et al. [TLL08] beobachtet wurde. Nach Beenden des Downloads wurden die gleichen Werte für die RTT wie vor dem Beginn des Downloads aufgezeichnet.

5.4 Skalierbarkeit

Um die Skalierbarkeit der erstellten Simulation zu untersuchen, wurden verschiedene Testszenarien erstellt. Diese wurden dann auf einem Computer mit acht Intel Xeon CPUs mit jeweils 3 GHz und 32 GB Arbeitsspeicher simuliert. Die Simulationen wurden jeweils nur von einem dieser Prozessoren ausgeführt. Nachdem eine Simulation eine Stunde Prozessorzeit in Anspruch genommen hatte, wurde sie beendet und die erreichte Simulationszeit aufgezeichnet.

Parameter	Wert
Anzahl Mobilgeräte	100
Anzahl stationäre Server	2
Anzahl Node B	175
Größe der Simulationsgrundfläche	500 m * 500 m
Art der Node B-Platzierung	gleichmäßige Platzierung
maximale Reichweite eines Node B	25 m
Verfügbare Downlinkbandbreite pro Node B	1 MBit/s
Verfügbare Uplinkbandbreite pro Node B	900 kBit/s
Bandbreitengarantien	keine
Optimierungsbereich eines Node B	>80% der maximalen Reichweite
Suche nach besserem Node B nach X Positionsupdates	5
Art der Mobilität	Random-Waypoint-Mobilität
Geschwindigkeit der Mobilgeräte	5 m/s

Tabelle 5.3: Verwendete Parameterwerte in den Skalierbarkeitstestläufen

Die Parameter, die standardmäßig und sofern nicht anders erwähnt in den Testläufen verwendet werden, sind im Tabelle 5.2 zu sehen. Die Parameter zur Verzögerungsmodellierung sind aus der Tabelle 5.1 im Abschnitt über das Verzögerungsverhalten übernommen. In den Testläufen baut jedes Mobilgerät eine TCP-Verbindung zu einem der stationären Server auf und simuliert dabei mittels der *TCPBasicClientApp* aus dem INET/MANET-Framework Traffic, der beim Aufruf von Webseiten anfällt. Die Mobilgeräte bewegen sich auf der Grundfläche nach der Random-Waypoint-Mobilität, die nacheinander zufällige Wegpunkte auf der Simulationsfläche generiert.

Folgende Größen wurden untersucht:

Größe der Simulationsfläche Ein Testlauf mit einer Simulationsgrundfläche von 200 m * 200 m (Versuch V1) und ein Testlauf mit einer Simulationsgrundfläche von 2000 m * 2000 m (V2), worauf der gleichmäßigen Platzierung entsprechend 33 beziehungsweise 2538 Node B platziert werden

Anzahl der Mobilgeräte Auf der Grundfläche werden entweder 10 (V3), 100 (V4) oder 1000 Mobilgeräte (V5) platziert

Platzierungsstrategie Es werden die zwei Platzierungsstrategien untersucht. Die gleichmäßige Platzierung wird in zwei Testläufen untersucht, wobei die maximalen Reichweite der Node B 25 m (V6) und 50 m (V7) beträgt. Die zufällige Platzierung wird ebenfalls in zwei Testläufen untersucht, wobei im ersten Lauf 175 Node B mit einem Radius von 25 m (V8) und im zweiten Lauf 45 Node B mit einem Radius von 50 m (V9) platziert werden. Das entspricht jeweils der Anzahl Node B und der maximalen Reichweite der Testläufe der gleichmäßigen Platzierung.

Reconnect-Delay Die Zeit, die ein Mobilgerät nach einem fehlgeschlagenem Verbindungsversuch zu allen umliegenden Node B wartet, bis es einen neuen Versuch startet, wird mit 0,1 s (V10), 0,5 s (V11) und 5 s (V12) gewählt. Um das Reconnect-Verhalten testen

zu können, werden die Node B zufällig auf der Grundfläche platziert, da bei einem Wabenmuster keine Funklöcher auftreten. Es werden 45 Node B mit einem Radius von 50 m auf der Grundfläche platziert.

Größe des Optimierungsbereichs und der Anzahl an notwendigen Positionsupdates Der Optimierungsbereich wird mit 50% (V13 und V14), 70% (V15 und V16) und 90% (V17 und V18) der maximalen Reichweite eines Node B bestimmt; die Anzahl an Positionsupdates in diesem Bereich, damit ein neuer Optimierungsversuch gestartet wird, beträgt entweder ein oder fünf Updates. Es werden alle möglichen Kombinationen getestet, sodass für jeden Wert des Optimierungsbereiches ein Testlauf mit einem notwendigen Positionsupdate für einen Optimierungsversuch und ein Testlauf mit fünf notwendigen Positionsupdates für einen Optimierungsversuch durchgeführt wird.

Mobilität Es werden drei verschiedene Mobilitätsmuster untersucht. Die Simulationsgrundfläche hat jeweils eine Größe von 1000 m * 1000 m.

Das erste getestete Bewegungsmuster ist die NS2-Trace-File-Mobilität. In diesem Versuch (V19) wird ein Ausschnitt einer Straßenkarte von Stuttgart eingelesen. Die Node B werden gleichmäßig platziert, wobei deren maximale Reichweite 50 m beträgt. Alle Mobilgeräte bewegen sich mit einer Geschwindigkeit von 5 m/s.

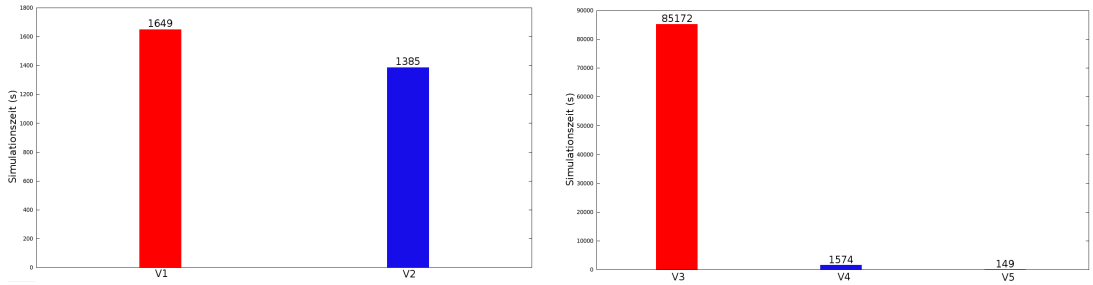
Das zweite untersuchte Bewegungsmuster ist die Random-Waypoint-Mobilität. Die Bewegungsgeschwindigkeit der Mobilgeräte wird mit 2 m/s (V20), 5 m/s (V21), 10 m/s (V22) und 20 m/s (V23) bestimmt.

Das dritte getestete Bewegungsmuster ist die Null-Mobilität (V24), bei der die Mobilgeräte auf ihrer zufällig gewählten Startposition verharren.

Keine Generierung von Traffic Es werden die gleichen Testläufe wie bei der Untersuchung der einzelnen Mobilitätsmuster durchgeführt, allerdings ohne TCP-Anwendung auf den Mobilgeräten, die Daten sendet und empfängt (NS2-Mobilität V25, Random-Waypoint-Mobilität V26-V29 und Null-Mobilität V30).

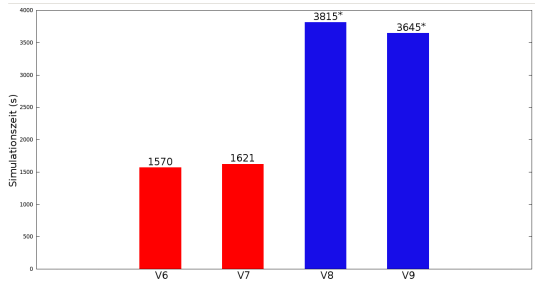
Jede Simulation wurde drei Mal durchgeführt, um Abweichungen der Ergebnisse erkennen zu können, die von unterschiedlich initialisierten Zufallsgeneratoren stammen. Wurden keine großen Abweichungen festgestellt, ist für die Ergebnisdarstellung der Durchschnitt der drei Testläufe berechnet worden. Andernfalls wurde das Szenario zehn Mal wiederholt, die Standardabweichung berechnet und der entsprechende Testlauf in den Ergebnissen mit einem Stern gekennzeichnet. Die nach einer Stunde beanspruchter Prozessorzeit erzielte Simulationszeiten sind in Abbildung 5.11 zu sehen.

Es gibt verschiedene Erkenntnisse, die aus den Ergebnissen der Skalierbarkeitstests gezogen werden können. Zum einen ist ersichtlich, dass bei höherer Anzahl an Node B auf der Grundfläche der Berechnungsaufwand steigt, was im Testlauf zu der unterschiedlichen Grundflächengrößen oder auch in dem zu den unterschiedlichen Radien bei der gleichmäßigen Platzierung der Node B zu sehen ist. Ein Grund hierfür ist, dass ein Mobilgerät für ein Handover zwischen zwei Node B eine Liste aller Node B benötigt, in dessen Reichweite es

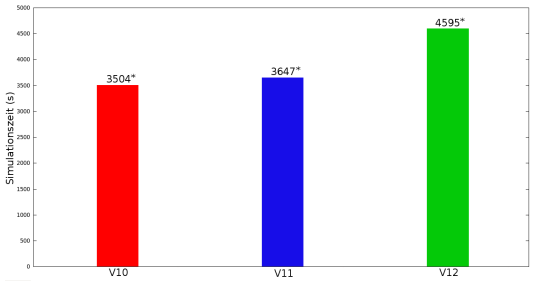


(a) Größe der Simulationsfläche

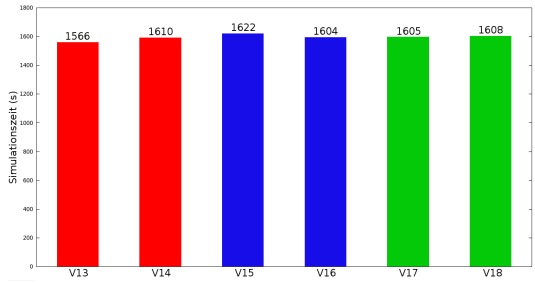
(b) Anzahl Mobilgeräte



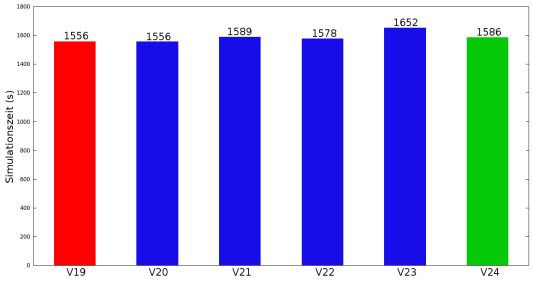
(c) Platzierungsstrategie



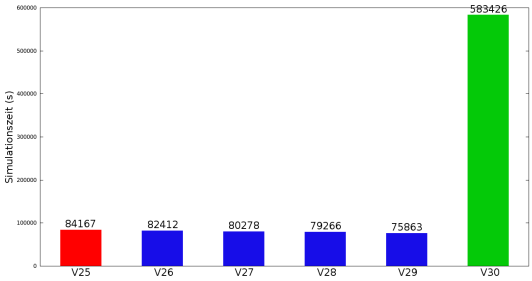
(d) Reconnect-Delay



(e) Optimierungsbereich



(f) Mobilität



(g) Kein Traffic

Abbildung 5.11: Ergebnisse der Skalierbarkeitstests

sich befindet. Die Berechnungszeit der Liste steigt linear in der Anzahl der Node B. Einen noch deutlicheren Einfluss auf den Berechnungsaufwand hat die Anzahl an Mobilgeräten der Simulation. Bei jeder Erhöhung der Anzahl an Mobilgeräten steigt der Berechnungsaufwand drastisch an. Keine Auswirkung dagegen hat die Wahl der Größe des Optimierungsbereichs eines Node B und die Anzahl der benötigten Positionsupdates in diesem Bereich für einen erneuten Optimierungsversuch.

Die Ergebnisse von Simulationen, bei denen die Node B zufällig platziert worden sind, schwanken je nach Initialisierung des Zufallsgenerators stark, wie bei der Untersuchung des Einflusses des Reconnect-Delays oder der zufälligen Platzierung selbst zu sehen ist. Hier beträgt die Standardabweichung zwischen 15% und 27% des Mittelwerts (zum Vergleich: bei allen anderen Versuchen beträgt die Standardabweichung weniger als 1% des Mittelwerts). Aus diesen Versuchen lässt sich ableiten, dass die Datenübertragung einen erheblichen Einfluss auf die Skalierbarkeit hat, da bei der zufälligen Platzierung Funklöcher auftreten, in denen keine Daten übertragen werden können. Die Größe der Funklöcher hängt von der zufälligen Platzierung der Node B ab und kann somit stark schwanken. Ein weiterer Hinweis für den immensen Einfluss der Datenübertragung ist bei den Tests der Mobilität zu sehen, die bei gleichmäßiger Platzierung der Node B einmal mit und einmal ohne Datenübertragung zum Server ausgeführt wurden. Dort war die erzielte Simulationsdauer je nach Mobilitätsart zwischen 46x und 368x länger, wenn kein HTTP-Traffic simuliert wurde.

Die Wahl des Reconnect-Delays beeinflusst ebenso die erreichte Simulationsdauer. Umso höher das Reconnect-Delay gewählt wird, desto kleiner wird der Berechnungsaufwand. Der Grund hierfür ist, dass in der Zeitspanne zwischen zwei Verbindungsversuchen kaum Berechnungen für das jeweilige Mobilgerät ausgeführt werden müssen. Umso länger also diese Zeitspanne ist, desto niedriger ist der Gesamtberechnungsaufwand.

Die einzelnen Mobilitätsarten unterscheiden sich ebenfalls in ihrer Skalierbarkeit. In den Testläufen ohne simulierten Datenverkehr hat trivialerweise die Null-Mobilität den geringsten Aufwand, gefolgt von der NS2-Mobilität und der Random-Waypoint-Mobilität. Die Geschwindigkeit, mit der sich die Mobilgeräte auf der Grundfläche bewegen, hat ebenfalls Auswirkungen auf die Simulationsdauer, wie anhand der Random-Waypoint-Mobilität zu beobachten ist. Umso höher die Geschwindigkeit der Mobilgeräte ist, desto höher ist der Berechnungsaufwand, da dann mehr Handover zwischen den Node B stattfinden, die berechnet werden müssen. Wenn Datenverkehr simuliert wird, unterscheiden sich die einzelnen Mobilitätsmuster kaum. Das lässt darauf schließen, dass der Einfluss des Datenverkehrs auf die Skalierbarkeit wesentlich höher ist als der der Mobilität.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Simulation für UMTS-Netze erstellt, die das reale Verhalten eines solchen Netzes in Bezug auf die Eigenschaften der Datenübertragung möglichst realistisch modelliert. Dazu wurde von der physikalischen Übertragung abstrahiert und eine Kommunikation basierend auf dem Link Layer aufgebaut.

Es wurde zuerst eine Einführung in die Grundlagen des UMTS-Standards und des Netzwerksimulators OMNeT++ gegeben. Danach wurden empirische Messungen zum Verzögerungsverhalten und zur Bandbreitenverteilung in UMTS-Systemen vorgestellt. Basierend auf diesen Erkenntnissen wurde dann die Simulation entwickelt und der zugehörige Entwurf vorgestellt.

Anschließend wurden die Details der Implementierung dargestellt. Dabei wurde auf die verwendeten Algorithmen eingegangen und Funktionsweisen beschrieben. Außerdem wurden Hinweise gegeben, auf was bei einer Änderung eines Moduls der Simulation geachtet werden muss.

Nach der Vorstellung des entwickelten Modells wurde es evaluiert und mit empirischen Messdaten verglichen. Dabei konnte gezeigt werden, dass das Verhalten der Simulation bezüglich Verzögerung und Bandbreitenzuteilung dem eines realen UMTS-Netzes entspricht. Danach sind die Ergebnisse der Skalierbarkeitstest dargestellt worden. Dabei konnten verschiedene Zusammenhänge aufgezeigt werden.

Das erstellte Modell simuliert das Verhalten eines UMTS-Netzes der ersten Generation. Inzwischen sind jedoch Erweiterungen wie HSDPA oder HSUPA verfügbar, die dem Nutzer zum einen eine deutlich höhere Bandbreite zur Verfügung stellen und die Paketverzögerungszeiten senken. Diese Erweiterungen könnten dem vorgestellten Modell noch hinzugefügt werden, damit die Simulation dem derzeitigen Stand der Realität entspricht. Um das Modell noch realistischer zu gestalten, kann die Struktur des UMTS-Netzes nach dem Funkbereich realitätsnaher modelliert werden. Dazu müssten zum Beispiel die RNC eingeführt werden oder auch das Core Network implementiert werden.

Die in der entwickelten Simulation verwendete Internetkomponente wurde sehr simpel gehalten, da dies nicht Hauptaugenmerk der Arbeit war. Anstatt jedes Paket um eine konstante Zeit zu verzögern, könnte hier eine deutlich realistischere Modellierung der Verzögerung entwickelt werden. Eine weitere mögliche Erweiterung der Simulation besteht bei den Mobilgeräten. Diesen könnten mehrere Interfaces hinzugefügt werden wie zum Beispiel ein WLAN-Interface, um die direkte Kommunikation der mobilen Geräte zu ermöglichen.

Eine Möglichkeit, um die Skalierbarkeit der Simulation zu erhöhen, ist die Berechnung der erreichbaren Node B eines Mobilgeräts zu beschleunigen. In einer Simulation mit sich

schnell bewegenden Mobilgeräten und geringer maximaler Reichweite der Node B wird diese Berechnung sehr oft durchgeführt. In der vorgestellten Implementierung wird die Liste aller Node B linear durchlaufen und überprüft, ob sich das Mobilgerät innerhalb der maximalen Reichweite befindet. Hier könnte ein verbesserter Algorithmus die Suche beschleunigen und somit die Simulation beschleunigen.

Literaturverzeichnis

[3GPa] 'Network architecture', 3GPP, Technical Specification 25401, 2000. (Zitiert auf Seite 11)

[3GPb] 'Radio interface protocol architecture', 3GPP, Technical Specification 25.301-v4.3.0 Release 4, June 2002. (Zitiert auf Seite 12)

[AS08] K. Anand, V. Sharma. Computing TCP Throughput in a UMTS Network. In *Proc. IEEE Wireless Communications and Networking Conf. WCNC 2008*, pp. 2507–2512. 2008. (Zitiert auf Seite 17)

[BCCF01] F. Borgonovo, A. Capone, M. Cesana, L. Fratta. Delay-throughput performance of packet service in UMTS. In *Proc. VTC 2001 Fall Vehicular Technology Conf. IEEE VTS 54th*, volume 4, pp. 2128–2132. 2001. (Zitiert auf Seite 17)

[CCE03] T. Chahed, A.-F. Canton, S.-E. Elayoubi. End-to-end TCP performance in W-CDMA / UMTS. In *Proc. IEEE Int. Conf. Communications ICC '03*, volume 1, pp. 71–75. 2003. (Zitiert auf Seite 17)

[CGGPC06] J. M. Cano-Garcia, E. Gonzalez-Parada, E. Casilari. Experimental Analysis and Characterization of Packet Delay in UMTS Networks. In *Proc. of 6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking, St. Petersburg*. 2006. (Zitiert auf den Seiten 7, 10, 13, 14, 19, 21, 24, 51, 52, 58 und 59)

[her] *HTC Hero Smartphone*. www.htc.com/de/product/hero/overview.html. (Zitiert auf Seite 58)

[LK05] J. Landman, P. Kritzinger. Delay analysis of downlink IP traffic on UMTS mobile networks. *Perform. Eval.*, 62:68–82, 2005. (Zitiert auf Seite 17)

[MBCCM10] J. M. Márquez-Barja, C. T. Calafate, J.-C. Cano, P. Manzoni. Evaluating the performance boundaries of WI-FI, WiMAX and UMTS using the network simulator (ns-2). In *Proceedings of the 5th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, PM2HW2N '10, pp. 25–30. 2010. (Zitiert auf Seite 17)

[omn] *OMNeT++ Network Simulation Framework*. www.omnetpp.org. (Zitiert auf Seite 10)

[TLL08] W. L. Tan, F. Lam, W. C. Lau. An Empirical Study on the Capacity and Performance of 3G Networks. *IEEE Transactions on Mobile Computing*, 7:737–750, 2008. (Zitiert auf den Seiten 10, 15, 19, 25, 53, 54, 58 und 59)

[umt] *Das Mobilfunkportal UMTSLink.at*. www.umtslink.at. (Zitiert auf den Seiten 7 und 12)

Alle URLs wurden zuletzt am 15.04.2011 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Thorsten Frosch)