Optimistische Replikation in mobilen ad-hoc Netzen

Dissertation zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
im Fachbereich Mathematik und Informatik der Freien Universität Berlin

vorgelegt von

Dipl.-Inform. Manuel Scholz

Berlin 2010

Erstgutachter: Prof. Dr. Heinz Schweppe, Freie Universität Berlin
Zweitgutachterin: Prof. Dr. Birgitta König-Ries, Friedrich-Schiller-Universität Jena

Datum der Disputation: 24. November 2010

Danksagung

Auf dem langen Weg von der ersten Idee bis zur Fertigstellung dieser Arbeit haben mir viele Menschen hilfreich zur Seite gestanden. All diesen Menschen möchte ich an dieser Stelle dafür danken, im Besonderen folgenden Personen:

Zuallererst meinem Doktorvater Prof. Heinz Schweppe für seine stetige Betreuung, die vielen Ratschläge und besonders die motivierenden und fruchtbaren Gespräche in der letzten Phase der Arbeit.

Meiner Zweitgutachterin Prof. Birgitta König-Ries, die mir ebenfalls mit guten Ratschlägen zur Seite stand und immer ein offenes Ohr für meine Fragen hatte.

Meiner Büropartnerin Katharina Hahn für ihren fachlichen und seelischen Beistand und die vielen tollen Gespräche bei einem heißen Kaffee.

Den fleißigen Diplomanden Danny Tschirner, Robert Bear, Holger Rösch, Yark Schröder, Frank Bregulla und Sebastian Martens für ihre Unterstützung.

Dem alten Kollegen Tobias Lenz und dem neuen Kollegen Andreas Noack für die fruchtbaren Diskussionen und fachlichen Ratschläge.

Meiner Freundin Heidrun für das leckere Essen und einen wundervollen ruhigen Ort, an dem man ungestört schreiben kann.

Meinen Eltern dafür, dass sie nicht so oft nachgefragt haben.

Ganz besonders meiner Frau Daniela, die mir gerade in schweren Zeiten den Rücken gestärkt und freigehalten hat.

Meinen beiden Töchtern Mila und Julie für ihre „Ablenkungsmanöver“. Sie haben mir oft gezeigt, dass es wichtigere Dinge im Leben gibt als Replikation, MANETs und Datenverteilung.

Kurzfassung

Die zunehmende Verkleinerung der Rechner und die Möglichkeiten der funkbasierten Datenübertragung ermöglichen heutzutage und in naher Zukunft die Bildung von mobilen Datennetzen, die sich in ihren Eigenschaften von den bisher verwendeten Festnetzen unterscheiden. Es stellt sich dabei die Frage, ob und wie sich die bisher in Festnetzen verwendeten Methoden in mobilen Netzen einsetzen lassen.

Die vorliegende Arbeit widmet sich dieser Fragestellung und untersucht die Verwendung von optimistischen Replikationsverfahren in mobilen ad-hoc Netzen (MANETs). MANETs sind dezentrale und infrastrukturlose Funknetze (WLAN), die von einzelnen mobilen Geräten (z. B. Smartphone, PDA, Laptop) spontan gebildet werden. Aufgrund der Mobilität und der Unzuverlässigkeit der Teilnehmer weisen MANETs eine hohe Dynamik auf, was zu langen Verbindungsunterbrechungen zwischen den Teilnehmern führen kann.

Die Verwendung von optimistischen Replikationsverfahren gewährleistet trotz der Verbindungsunterbrechungen eine hohe Verfügbarkeit der Daten. Dabei werden Änderungen von jedem Teilnehmer zunächst auf dem lokalen Datenbestand ausgeführt und anschließend an die anderen Netzteilnehmer verteilt. Es ergibt sich jedoch durch kausal parallel ausgeführte Änderungsoperationen auf den gleichen Daten das Problem von Schreibkonflikten.

Da es bisher nur wenig Erfahrung mit dem Einsatz von Replikationsverfahren in MANET-Szenarien gibt, wurde in dieser Arbeit ein optimistisches Replikationssystem entwickelt und untersucht, wie sich die Eigenschaften mobiler ad-hoc Netze auf dieses System auswirken. Die Methoden des Systems wurden für den Einsatz in MANETs entworfen und angepasst, da viele bisherige Replikationssysteme nicht für die Verwendung in MANETs geeignet sind. Der Einsatz eines zentralen Replikationsservers, der mit den mobilen Knoten via UMTS oder GPRS verbunden ist, wäre z. B. nicht sinnvoll, da MANET-Szenarien dadurch motiviert sind, dass keine Infrastruktur genutzt werden kann bzw. aus Kostengründen nicht genutzt wird.

Um die Mobilität der MANET-Szenarien zu berücksichtigen, wurden zunächst Bewegungsmodelle und Datenverteilungsprotokolle für MANETs betrachtet. Das entwickelte *Area Graph-based Bewegungsmodell* modelliert sowohl die makroskopische als auch die mikroskopische Bewegung der mobilen Knoten und ist dadurch realistischer als Bewegungsmodelle, die sich nur auf die Modellierung der Makro- *oder* der Mikrobewegung beschränken. Die entwickelten Datenverteilungsprotokolle, *adaptives probabilistisches Fluten* und *adaptive Synchronisation*, wurden mit Hilfe von Netzsimulatoren mit anderen Protokollen verglichen. Die Ergebnisse zeigen, dass die beiden Protokolle weniger Netzlast verursachen als vergleichbare Ansätze.

Das Konfliktlösungs- und -erkennungsverfahren unseres Replikationsansatzes basiert auf den kausalen Abhängigkeiten der Transaktionen. Die Abhängigkeiten werden im Replikationssystem von einer Datenstruktur, dem sogenannten *Vorgängergraph*, verwaltet. Dieser stellt eine Erweiterung der Vorgängerliste dar und erlaubt es, nicht nur Operationen, sondern auch Transaktionen zu verwalten. Zusätzlich wurde die Datenstruktur so erweitert, dass auch kommutative Transaktionen berücksichtigt werden können.

Um die verwendeten Replikationsverfahren möglichst realistisch zu testen, wurde das *prototypische Replikationssystem* SYMORE implementiert und in kleinen Szenarien auf Laptops und PDAs getestet. Ebenso wurden auf dem Replikationssystem aufbauend zwei Applikationen (verteiltes Puzzle und verteiltes Wiki) für mobile Geräte (Laptop, PDA) erfolgreich umgesetzt.

Abschließend wurden die verwendeten Replikationsmethoden sowohl experimentell, mit Hilfe von Emulationen, als auch analytisch bewertet. Als Zielgröße wurde dabei die relative Anzahl der in Konflikt stehenden Transaktionen betrachtet. Die Ergebnisse der Bewertung zeigen, dass sich in den meisten der untersuchten MANET-Szenarien nur eine geringe Anzahl von Konflikten und somit auch weniger Transaktionsabbrüche ergaben. Insgesamt eignet sich das entwickelte Replikationssystem somit für den Einsatz in kleinen bis mittelgroßen MANET-Szenarien von bis zu 40 Teilnehmern.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Durch die zunehmende Verkleinerung der Hardware und die Integration der Funktechnologie werden Rechner heutzutage immer flexibler und mobiler einsetzbar. Beispiele dafür sind Laptops, Netbooks wie der Eee-PC [epc10], PDAs (Personal Digital Assistant) oder Smartphones. Aufgrund der Funktechnologie dieser Geräte ist die Bildung von sogenannten mobilen ad-hoc Netzen (kurz MANETs) möglich. MANETs unterscheiden sich in ihren Eigenschaften erheblich von kabelgebundenen Netzen. Durch die Mobilität der Teilnehmer können Netzverbindungen spontan entstehen und ebenso schnell wieder unterbrochen werden. Daraus resultiert in mobilen Netzen eine sehr hohe Dynamik, die in herkömmlichen Netzen nicht vorhanden ist.

In dieser Arbeit wird die Replikation in MANETs betrachtet und der Frage nachgegangen, *wie sich die Eigenschaften mobiler ad-hoc Netze auf die Replikation auswirken*. Diese Frage soll durch die Entwicklung und Untersuchung eines Replikationssystems beantwortet werden. Die Methoden des Systems müssen speziell für den Einsatz in MANETs konzipiert sein, da viele bisherige Replikationssysteme nicht für die Verwendung in MANETs geeignet sind. So wäre z. B. der Ansatz, einen zentralen Replikationsserver zu verwenden, der mit den mobilen Knoten via UMTS oder GPRS verbunden ist, nicht umsetzbar. Das liegt daran, dass der Einsatz von MANETs durch Szenarien motiviert ist, in denen keine Infrastruktur genutzt werden kann bzw. aus Kostengründen nicht genutzt wird. Ein typisches MANET-Szenario ist ein Katastrophengebiet, in dem aufgrund einer Naturkatastrophe keine Infrastruktur mehr vorhanden ist und wo somit nur noch auf ad-hoc Verbindungen zurückgegriffen werden kann.

In dieser Arbeit werden Replikationsszenarien betrachtet, in denen jeder Teilnehmer Änderungen an den Daten vornehmen darf. Ein denkbares Szenario ist das bereits genannte Katastrophengebiet, in dem Rettungskräfte mit Hilfe einer Art Wiki Anwendung Daten über vermisste Personen, die Versorgungslage usw. aufnehmen. Um solche nebenläufigen Änderungen zu verwalten, wird in bisherigen Systemen häufig eine synchrone Kommunikation zwischen den Teilnehmern verwendet. Dieser Ansatz eignet sich ebenfalls nicht für den Einsatz in MANETs: Aufgrund der dynamischen Netztopologie ist die Kommunikation häufig gestört und dadurch verzögert. Eine synchrone

Kommunikation ist unter solchen Bedingungen praktisch unbrauchbar. Ebenso verhält es sich mit Replikationsansätzen, die Sperrverfahren einsetzen. Durch die verzögerte Kommunikation bleiben Sperren unverhältnismäßig lang bestehen und verhindern einen Zugriff auf die Daten durch andere Teilnehmer.

Aus den oben genannten Gründen wird ein asynchroner und optimistischer Ansatz zur Replikation der Daten vorgeschlagen, wie er bereits in ähnlicher Form bei der Datenbankreplikation im Festnetz (z. B. Oracle) verwendet wird. Bei optimistischen Replikationsverfahren werden die Daten beim Zugriff nicht gesperrt. Somit steht jedem Teilnehmer lokal der gesamte Datenbestand zur Verfügung, auch wenn dieser vom Netz getrennt ist. Dies ist besonders in MANET-Szenarien wichtig, da die Teilnehmer längere Zeit vom Netz getrennt sein können. Die Datenänderungen der Teilnehmer werden zunächst lokal ausgeführt und anschließend an die anderen Teilnehmer der Replikationsgruppe versendet. Aufgrund nebenläufiger Änderungen können somit Konflikte entstehen, die vom Replikationssystem aufgelöst werden müssen. Eine Vielzahl dieser Konflikte kann bei der Konfliktlösung zu kaskadierenden Abbrüchen führen, wodurch das System praktisch unbrauchbar wird.

Optimistische Ansätze eignen sich daher eher für Szenarien, in denen in erster Linie Lesezugriffe durchgeführt werden und Schreibzugriffe seltener vorkommen. Ein typisches Beispiel dafür ist ein Szenario, in der eine Wiki Anwendung als Informationssystem verwendet wird. Bei einem Vergleich von Lese- und Schreibzugriffen kann man erkennen, dass die Lesezugriffe deutlich überwiegen. Betrachtet man zum Beispiel die Zugriffsstatistiken von Wikipedia, so wird ersichtlich, dass die Anzahl der Änderungsoperationen [wik09a] nur einen geringen Bruchteil der Anzahl der lesenden Zugriffe [wik09b] darstellt.

Bisher gibt es fast keine Erfahrungen über den Einsatz von optimistischer Replikation in MANETs. Deshalb wird in dieser Arbeit untersucht, inwieweit sich optimistische Replikationsmethoden für den Einsatz in MANET-Szenarien eignen.

1.2 Herausforderungen

Wie bereits einleitend erwähnt, ergeben sich aufgrund der Eigenschaften eines MANETs Probleme für die Replikation, die in kabelgebundenen Netzen in dieser Form nicht auftreten. Dies gilt auch für andere Netze (UMTS, GSM/GPRS) mit mobilen Teilnehmern, da sie im Gegensatz zu MANETs eine feste Infrastruktur besitzen. Im Folgenden werden die wichtigsten Probleme und die sich daraus ergebenden Fragen im Hinblick auf optimistische Replikation betrachtet.

Mobilität Aufgrund einer fehlenden Infrastruktur wirkt sich die Mobilität der Teilnehmer in MANETs direkt auf die Netzverbindungen aus und ist deshalb die zentrale Ursache für eine hohe Dynamik. Die mobilen Knoten bewegen sich in den Funkbereich anderer Knoten hinein oder heraus. Dies führt zu Verbindungsänderungen zwischen den Teilnehmern und häufigen Änderungen der Netztopologie: Knoten verlassen das Netz und treten ihm wieder bei. Es ergeben sich viele einzelne

Partitionen, die aus einem oder mehreren Knoten bestehen. In solch dynamischen Netzen gestaltet sich die Verteilung der zu replizierenden Daten schwieriger als in Netzen mit fester Infrastruktur und es erhöht sich die Datenverteilungsdauer. Dadurch steigt ebenfalls die Gefahr von nebenläufigen Datenänderungen, wodurch bei optimistischer Replikation potentiell mehr Konflikte (nebenläufige Änderungen der gleichen Daten) verursacht werden.

Unzuverlässige Knoten Aufgrund der Mobilität sind die Geräte häufig vom Stromnetz getrennt und verfügen somit nur über eingeschränkte Energieressourcen. Dadurch werden die Geräte häufiger abgeschaltet oder fallen aufgrund von Energiemangel abrupt aus. Ebenso sind mobile Geräte vielen wechselnden Umwelteinflüssen ausgesetzt und weisen somit im Allgemeinen eine kürzere Lebensdauer als stationäre Desktop PCs auf. Würde sich ein Server auf einem mobilen Gerät befinden, so wäre die Leistungsfähigkeit eines Replikationssystems durch dessen Unzuverlässigkeit eingeschränkt. Serverbasierte Ansätze sind daher für den Einsatz in MANETs nicht geeignet.

Eingeschränkte Leistung Mobile Geräte sind aufgrund der Größe in ihrer Leistungsfähigkeit eingeschränkt. Sowohl die Rechenleistung als auch die Speicherkapazität sind deutlich niedriger als bei Desktop PCs. Um die mobilen Geräte nicht zu überlasten, dürfen die Replikationsmethoden nur eine geringe Komplexität aufweisen.

Diese aufgeführten Probleme werden im Rahmen der Arbeit berücksichtigt und führen zu den zentralen Forschungsfragen:

- Ist ein optimistischer Replikationsansatz für den Einsatz in MANETs trotz der erhöhten Datenverteilungsdauer praktikabel oder entstehen zu viele Konflikte?
- Wie lässt sich ein optimistischer Replikationsansatz in MANETs dezentral organisieren?
- Welche optimistischen Replikationsmethoden eignen sich aufgrund ihrer Komplexität für den Einsatz auf mobilen Geräten?

1.3 Ziele der Arbeit

Im Rahmen dieser Arbeit wird betrachtet, wie sich die Verwendung von optimistischen Replikationsmethoden in MANETs verhält. Insbesondere sollen dabei folgende Ziele verfolgt werden:

Evaluation von Replikationsmethoden in MANET-Szenarien Der Hauptteil der Arbeit behandelt die Bewertung von optimistischen Replikationsmethoden in MANET-Szenarien. Da für die Benutzbarkeit eines optimistischen Replikationsansatzes die Häufigkeit der nebenläufigen Schreibzugriffe und die daraus resultierende Gefahr von Konflikten eine entscheidende Rolle spielt,

stellt dieser Aspekt die zentrale Zielgröße der Bewertung unseres Ansatzes dar. Diese Zielgröße soll sowohl experimentell, mit Hilfe von Emulationen, als auch analytisch untersucht werden. Die experimentelle Untersuchung soll anhand eines Replikationssystems durchgeführt werden, wodurch sich das nachfolgende Ziel ergibt.

Entwicklung und Implementierung eines Replikationssystems für MANETs Die von uns verwendeten Replikationsmethoden sollen nicht nur isoliert betrachtet werden. Die Realisierung eines Replikationssystems dient als ein Proof-of-Concept (Machbarkeitsstudie) für die Gesamtheit der verwendeten Replikationsmethoden und als Basis für die experimentelle Bewertung der optimistischen Replikation in MANETs. Das System soll dezentral organisiert und auf realen mobilen Geräten einsetzbar sein. Ebenso sollen Beispielanwendungen entwickelt werden, die das System zur Replikation ihrer Daten nutzen.

Anpassung von Replikationsmethoden an die Eigenschaften von MANETs Um geeignete Replikationsmethoden für mobile ad-hoc Netze zu entwickeln, werden bisherige Replikationsmethoden betrachtet und für die Verwendung in MANET-Szenarien angepasst. Dazu gehören die Methoden für die Erkennung und die Lösung von Konflikten, die aufgrund der geringen Leistung der mobilen Geräte nur eine geringe Komplexität aufweisen dürfen. Ebenso arbeiten z. B. Verteilungsprotokolle in MANETs aufgrund der Netzdynamik grundsätzlich anders als Verteilungsprotokolle in Festnetzen. Zur Betrachtung der Mobilität müssen geeignete Bewegungsmodelle ausgewählt bzw. erweitert werden, um MANET-Szenarien möglichst realistisch zu modellieren.

1.4 Beiträge der Arbeit

Im Rahmen der gesamten Arbeit sind folgende Hauptbeiträge hervorzuheben:

Datenverteilung Für eine effiziente und sichere Verteilung der Daten werden zwei adaptive Protokolle speziell für die Datenverteilung in mobilen ad-hoc Netzen entworfen. Die beiden Protokolle (adaptives probabilistisches Fluten und adaptive Synchronisation) werden mit Hilfe von Netzsimulatoren mit anderen Protokollen verglichen.

Bewegungsmodell Da die herkömmlichen Bewegungsmodelle für mobile ad-hoc Netze einige Einschränkungen aufweisen, haben wir das Area Graph-based Bewegungsmodell entwickelt. Dieses kombiniert das Graph-based Bewegungsmodell [THB+02] mit dem Random Waypoint Modell [BHPC04], um sowohl die makroskopische als auch die mikroskopische Bewegung der Knoten zu modellieren. Das Area Graph-based Bewegungsmodell ist dadurch realistischer als Bewegungsmodelle, die sich nur auf die Modellierung der Makro- oder der Mikrobewegung beschränken.

Konflikterkennung und Konfliktlösung Für die dezentrale Konflikterkennung wird eine Datenstruktur (Vorgängergraph) verwendet, die die Änderungshistorie der Datenelemente enthält. Der Vorgängergraph ist eine Erweiterung der von manchen Systemen [Hup09, PSM03] verwendeten Vorgängerliste und erlaubt zusätzlich die Verwaltung von Transaktionen und die Berücksichtigung von kommutativen Transaktionen. Die Konfliktlösung und das Commit-Verfahren des Systems basieren auf einem von uns entwickelten Echtzeit-Zeitstempel-Verfahren.

Replikationssystem Als Proof-of-Concept wird das Replikationssystem SYMORE implementiert. Diese Java Implementierung basiert auf den von uns entwickelten Replikationsmethoden und ist in MANET-Szenarien einsetzbar. Ebenso werden auf dem Replikationssystem aufbauend zwei Applikationen (verteiltes Puzzle und verteiltes Wiki) für mobile Geräte wie Laptops oder PDAs entwickelt.

Evaluierung Um die Verwendbarkeit der Replikationsmethoden für größere Szenarien zu evaluieren, wird die vorhandene Implementierung mit Hilfe des MarNET-Emulators für verschiedene MANET-Szenarien getestet. Dabei werden kleine bis mittelgroße Replikationsgruppen von bis zu 40 Teilnehmern betrachtet. Weiterhin wird untersucht, inwieweit sich die Berücksichtigung von kommutativen Transaktionen auf die Abbruchrate der Transaktionen auswirkt.

Konfliktanalyse Für die Validierung der Emulation und die Abschätzung der Ergebnisse weiterer Szenarien werden die Konflikte analytisch betrachtet. Durch diese Konfliktanalyse kann mit Hilfe der Parameter eines Szenarios die resultierende Konfliktrate der Transaktionen berechnet werden.

1.5 Aufbau der Arbeit

Der Aufbau der Arbeit ist in Abbildung 1.1 dargestellt. Die Arbeit gliedert sich in vier Teile.

Einführung Im ersten Teil der Arbeit wird der Themenbereich *Replikation* eingeführt, dabei werden grundlegende Begriffe und Verfahren definiert und erläutert (Kapitel 2). Insbesondere wird auf den in der Arbeit thematisierten Bereich der *optimistischen Replikation* eingegangen. Ebenso werden die Eigenschaften und Besonderheiten von mobilen ad-hoc Netzen erläutert. Als Grundlage für alle weiteren Kapitel wird unser *Systemmodell* vorgestellt. In Kapitel 3 werden Replikationssysteme vorgestellt, die mit den in dieser Arbeit verwendeten Ansätzen verwandt sind.

Methoden und Verfahren Der zweite Teil der Arbeit beinhaltet die von uns verwendeten Methoden und Verfahren zur Thematik der Arbeit. Zu Beginn wird das *Area Graph-based Modell* vorgestellt, das verwendet wird, um MANET-Szenarien realistischer zu modellieren (Kapitel 4). Anschließend werden die von uns entwickelten Protokolle zur *Datenverteilung* (Kapitel 5), ein *Uhren-*

sychronisationsverfahren (Kapitel 6) und die Methoden zur *Konflikterkennung* und *-lösung* (Kapitel 7) vorgestellt. Diese einzelnen Methoden werden in dem *Replikationssystem* (Kapitel 8) SYMORE zusammengefasst und implementiert.

Bewertung Im dritten Teil der Arbeit wird die Gesamtheit der Replikationsmethoden bewertet. Dies geschieht einerseits mit Hilfe der vorgestellten Konfliktanalyse (Kapitel 9). Andererseits wird das implementierte Gesamtsystems mit Hilfe von experimentellen Untersuchungen evaluiert (Kapitel 10).

Abschluss Abschließend wird der Inhalt der Arbeit zusammengefasst und ein Ausblick auf weiterführende Problemstellungen gegeben (Kapitel 11).

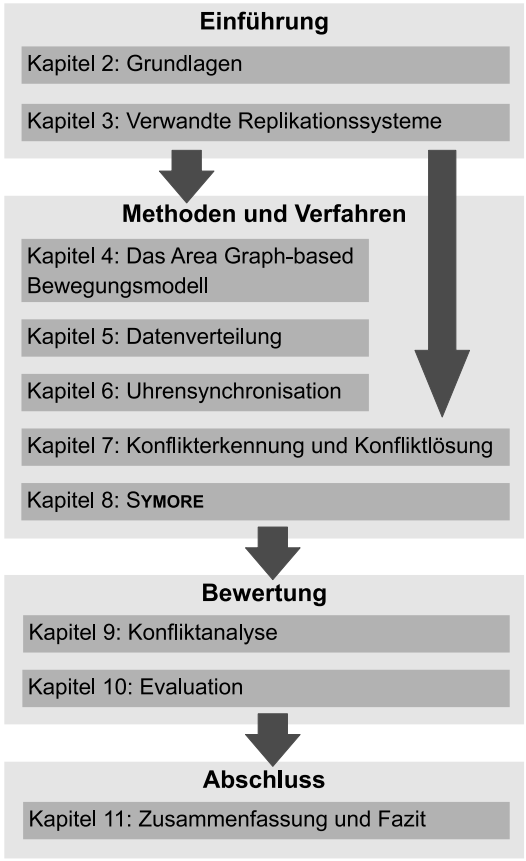


Abbildung 1.1: Aufbau der Arbeit. Kapitel 7 baut nicht zwingend auf den Kapiteln 4-6 auf und kann alternativ auch direkt nach Kapitel 3 gelesen werden.

Teil I

Einführung

Kapitel 2

Grundlagen

In diesem Kapitel werden die Grundlagen der in dieser Arbeit behandelten Themen erläutert. Einleitend werden mobile ad-hoc Netze und die Replikation im Allgemeinen erklärt (Abschnitt 2.1 und 2.2). Anschließend wird auf die speziellen Aspekte der optimistischen Replikation eingegangen (Abschnitt 2.3). Zuletzt wird das Systemmodell definiert, das als Grundlage für alle weiteren Kapitel dient (Abschnitt 2.4).

2.1 Mobile ad-hoc Netze

Ein **mobiles ad-hoc Netz** (kurz MANET) ist laut [TF06, Hek06] ein aus **mobilen Knoten** *bestehendes und auf Funktechnik basierendes Netz*. Weiterhin wird betont, dass MANETs *keine Infrastruktur* benötigen und *dezentral organisiert* sind. Dadurch heben sie sich inbesondere von infrastrukturbasierten Netzen mit mobilen Teilnehmern wie z. B. UMTS oder GSM/GPRS Netzen ab. Abhängig vom jeweiligen Szenario gibt es viele Varianten mobiler ad-hoc Netze. Beispiele reichen von kleinen, autarken Netzen mit weniger als fünf Knoten bis hin zu großflächigen MANETs mit mehreren hundert Teilnehmern, bei denen ein Teil der Knoten mit dem Internet verbunden ist. Häufig verwendete MANET-Szenarien sind Katastrophenszenarien, in denen aufgrund einer Katastrophe keine Infrastruktur mehr vorhanden ist und nur noch auf ad-hoc Verbindungen zurückgegriffen werden kann.

MANETs bestehen aus mobilen Geräten wie Laptops, Smartphones oder PDAs. Diese mobilen Geräte besitzen im Vergleich zu Desktop Rechnern eine eingeschränkte Leistungsfähigkeit bezüglich ihres Speichers und ihrer Prozessorleistung. Ebenso besitzen sie nur beschränkte Energieressourcen. Die teilnehmenden Geräte sind die Netzknoten eines MANETs und kommunizieren via Wireless LAN (IEEE 802.11) [Com99] miteinander. Der Standard IEEE 802.11 umfasst mehrere Substandards, die sich im verwendeten Frequenzband, der Funkreichweite und der Übertragungsrate unterscheiden [Hek06].

Mobilen Knoten, die sich in Funkreichweite anderer Knoten befinden, können mit diesen di-

rekt kommunizieren. Es besteht eine **direkte Verbindung**. Dies wird auch *single-hop*[1] Verbindung genannt. Alle Knoten, zwischen denen eine direkte Verbindung besteht, werden als **Nachbarn** bezeichnet. Werden Daten zwischen zwei mobilen Knoten über mehrere single-hop Verbindungen weitergeleitet, wird dies als **indirekte Verbindung** (engl. multi-hop) bezeichnet.

Eine zentrale Eigenschaft von MANETs ist ihre dynamische Topologie: Durch die Bewegung der Netzknoten können Funkverbindungen kurzfristig neu entstehen bzw. abbrechen. Ebenso sind die teilnehmenden Knoten im Gegensatz zu Festnetzen unzuverlässiger, da sie über beschränkte Energieressourcen verfügen und daher häufig abgeschaltet werden. Aus diesem Grund kommt es in MANETs häufig zur Partitionierung des Netzes. Eine **Partition** ist eine Menge von mobilen Knoten, die direkt oder indirekt miteinander verbunden sind. Zwischen den Knoten unterschiedlicher Partitionen besteht keine direkte oder indirekte Verbindung.

2.1.1 Routing in MANETs

Aufgrund der oben beschriebenen Eigenschaften ist die Entwicklung und Untersuchung von Routingprotokollen ein großer Bereich in der MANET Forschung. Routingprotokolle ermöglichen ein Versenden von Nachrichten an einen (**Unicast**) oder mehrere (**Multicast**) Teilnehmer derselben Partition. Es gibt bereits eine Vielzahl an Protokollen, die für unterschiedliche Anforderungen und Voraussetzungen entworfen wurden. Zwei häufig verwendete Protokolle sind *Dynamic Source Routing* (DSR) [JM96] und *Ad-hoc On Demand Distance Vector routing* (AODV) [PR99]. Beide Protokolle sind reaktiv, d. h., sie fordern eine Aktualisierung von Routinginformationen nur bei Bedarf an. Dadurch wird der Nachrichtenaufwand im Netz verringert.

2.1.2 Datenverteilung in MANETs

Im Gegensatz zum Routing sollen bei der Datenverteilung möglichst alle Teilnehmer eines Netzes erreicht werden. Dafür werden sogenannte **Broadcastprotokolle** eingesetzt. Broadcastprotokolle in MANETs können in zwei Kategorien eingeteilt werden: Flutungsprotokolle und auf Synchronisation basierende Protokolle.

Das einfache Grundprinzip von **Flutungsprotokollen**[2] ist das Senden und Weiterversenden einzelner Nachrichten. Ein einzelner Knoten startet einen Broadcast, indem er die Nachricht an seine Nachbarn verschickt. Diese schicken dann die Nachricht ebenfalls weiter, so dass schrittweise immer mehr Knoten erreicht werden. Wenn diese Art des Broadcasts gewählt wird, können nur Knoten erreicht werden, die sich in derselben Partition wie der Startknoten befinden. Knoten außerhalb dieser

[1] Aufgrund zum Teil fehlender oder nur schwer umsetzbarer Übersetzung englischer Begriffe werden bei Bedarf in dieser Arbeit englische Begriffe verwendet.

[2] Der Begriff „Flutungsprotokoll“ wird von uns als Übersetzung für den englischen Begriff „flooding protocol“ verwendet.

Partition erhalten die Nachricht nicht. Besonders in MANETs mit geringer Knotendichte (engl. sparse MANETs) stellt diese Einschränkung einen großen Nachteil dar. Ein Ansatz, um diesen Nachteil zu umgehen, ist das erneute Versenden der Nachrichten nach einer gewissen Zeit (Hyperflooding).

Synchronisationsbasierte Broadcastprotokolle verteilen die Nachrichten im Netz durch die paarweise durchgeführte Synchronisation der Netzteilnehmer. Schritt für Schritt werden so durch die einzelnen Synchronisationen die Nachrichten an alle anderen Teilnehmer verteilt. Das ist langsamer als eine auf Fluten basierende Verteilung [Khe07]. Im Gegensatz zu den Flutungsprotokollen werden dabei auch im Fall von temporären Netzpartitionen nach einer gewissen Zeit alle Netzteilnehmer erreicht. Vertreter aus beiden Broadcast-Kategorien werden in Kapitel 5 beschrieben, in dem ebenfalls die von uns entwickelten Protokolle vorgestellt und bewertet werden.

2.1.3 Simulation

Im Gegensatz zu LANs (Local Area Networks), WANs (Wide Area Networks) und dem Internet gibt es bisher keine größeren real existierenden MANETs oder diese sind nicht frei verfügbar (z. B. Militärnetze). Auch der künstliche Aufbau derartiger Netze mit Hilfe von echten Geräten und Personen ist sehr aufwändig und kostenintensiv. Aus diesem Grund werden MANET-Szenarien häufig simuliert, wodurch die Untersuchungen vereinfacht werden. Für die Simulation von MANETs werden Netzsimulatoren verwendet. Ein häufig verwendeter Simulator dieser Art ist ns2 [ns10].

Bei der Verwendung von MANET Simulationen werden sogenannte Bewegungsmodelle eingesetzt, um die Bewegung der einzelnen mobilen Knoten nachzubilden. Ein bekanntes Modell ist das *Random Waypoint Bewegungsmodell* [BHPC04], das die Bewegung der Netzknoten mit Hilfe von Wegpunkten modelliert. Eine detaillierte Betrachtung von Bewegungsmodellen für MANETs erfolgt in Kapitel 4.

Für die Simulation ebenfalls notwendig ist die Modellierung der Funkverbindungen zwischen den mobilen Knoten [KNG+04]. In den meisten Fällen wird dafür ein sehr einfaches Modell verwendet. Dieses geht von einer gleichmäßigen, kreisförmigen Ausbreitung der Radiowellen auf einer flachen Ebene aus. Innerhalb einer gegebenen Funkreichweite wird für die beteiligten Knoten eine ideale Verbindung ohne Verluste angenommen. Befinden sich zwei Knoten außerhalb dieser Funkreichweite, so gelten sie im Modell als nicht verbunden. Für eine realistischere Modellierung der Funkverbindungen wurden Modelle entwickelt, die die Verbindungsqualität genauer modellieren. Hierbei wird auf Basis des Funkabstands die Verlustrate der gesendeten Pakete festgelegt. Ein Beispiel hierfür ist das CMU 802.11 Modell, das im Simulator ns2 verwendet wird. Es gibt ebenfalls Modelle, die noch weitere Aspekte wie z. B. die Reflexion von Radiowellen berücksichtigen. Eine weiterführende Diskussion dieser Modelle findet sich im Artikel von Kotz et al. [KNG+04].

2.1.4 Netzmodell

In dieser Arbeit wird der Standard 802.11b verwendet, der Übertragungsraten von 11 Mbit/s ermöglicht. Die Funkreichweite kann dabei im offenen Gelände bis zu 100m betragen. In Gebäuden oder im Gelände mit dichter Bebauung bzw. dichtem Bewuchs liegt die Funkreichweite bei diesem Standard jedoch deutlich unter 100m [Hek06]. Für die Modellierung der Funkverbindung wird eine kreisförmige Ausbreitung der Radiowellen angenommen. Die Paketverlustrate wird auf Basis des Funkabstands mit Hilfe des Standardmodells des jeweiligen Simulators berechnet. Der Ausfall von Knoten wird in dieser Arbeit berücksichtigt, eine explizite Modellierung ist jedoch nicht notwendig (siehe Abschnitt 2.4.5).

Für die Datenverteilung werden Flutungsprotokolle und synchronisationsbasierte Broadcastprotokolle verwendet, da alle Teilnehmer des Netzes erreicht werden sollen. In dieser Arbeit werden MANETs mit geringer Knotendichte betrachtet (siehe Kapitel 10), in denen der Einsatz von Routingprotokollen fragwürdig ist (siehe [ZAZ04]). Ebenso wird ein Unicast und Multicast in den betrachteten Szenarien nicht benötigt. Aus diesen Gründen wird das Vorhandensein von Routingprotokollen im Rahmen dieser Arbeit nicht vorausgesetzt. Die für die Replikation notwendigen Daten werden in Form von Nachrichten und nicht als Streams versandt.

Datenverteilungsdauer Ein für die Replikation zentraler Aspekt ist die *Datenverteilungsdauer* in MANETs. Sie bezeichnet die Dauer, die für das Versenden, Empfangen und Verarbeiten einer Nachricht benötigt wird. Bei einem Broadcast ergibt sich ein Wert für jeden Knoten des Netzes und es kann eine **mittlere Dauer** und eine **Gesamtdauer** berechnet werden. In Netzen mit einer festen Infrastruktur beträgt die Datenverteilungsdauer meist weniger als eine Sekunde. Aufgrund der beschriebenen Eigenschaften von MANETs, wie z. B. die häufige Partitionierung des Netzes, kann die Datenverteilungsdauer hierbei zum Teil einige Stunden betragen. *Die Auswirkung der erhöhten Datenverteilungsdauer in MANETs auf die Replikation ist ein zentrales Untersuchungsziel dieser Arbeit.*

Szenarien

Im Rahmen dieser Arbeit werden verschieden MANET-Szenarien betrachtet. Dabei handelt es sich im Einzelnen um folgende Szenarien:

Katastrophenszenario/Forscherszenario In Katastrophengebieten stehen die Einsatzkräfte von Hilfsorganisationen oft vor dem Problem, eine große Menge von Zustandsinformationen (z. B. Nahrungsvorräte, Medikamente, Vermisste und Verwundete) zu verwalten. Ebenso ist in Katastrophengebieten nach einem Erdbeben, Orkan oder einer Flut die technische Infrastruktur häufig stark eingeschränkt oder steht überhaupt nicht zur Verfügung. Aus diesem Grund bietet sich hier die Replikation der Verwaltungsdaten mit Hilfe eines MANETs an, da dieses

nur die mobilen Geräte der Helfer und keine zusätzliche Infrastruktur benötigt. Denkbar wäre der Einsatz eines mobilen Verwaltungssystems, mit dessen Hilfe die aktuellen Daten von den Helfern vor Ort auf einem PDA oder Laptop eingetragen und verändert werden können. Aufgrund der Organisation solcher Hilfseinsätze ist anzunehmen, dass sich die Einsatzkräfte in regelmäßigen Abständen an einem Ort zusammenfinden, um weitere Vorgehensweisen abzustimmen.

Ein ähnliches Szenario ist für eine Gruppe von Forschern (z. B. Biologen, Archäologen) denkbar, die in abgelegenen Gebieten Feldforschung betreiben und Informationen sammeln. Aufgrund fehlender Infrastruktur wird die Replikation der Forschungsdaten ebenfalls mit Hilfe eines MANETs realisiert.

Sitzungsszenario/Spielrundenszenario Ein weiteres Szenario für Replikation in MANETs stellen Sitzungen oder Spielrunden dar, in denen Daten repliziert werden müssen. In beiden Szenarien bewegen sich die Teilnehmer nur wenig oder überhaupt nicht und befinden sich räumlich nah beieinander. Bei einer Sitzung ist zum Beispiel ein Terminkalender denkbar, der zwischen den Geräten der Teilnehmer direkt ohne eine Verbindung zu einem außenstehenden Netz abgeglichen wird. Eine Spielrunde, die sich spontan an einem bestimmten Ort (Zugabteil o. ä.) bildet und die ihre Spieldaten über ein MANET repliziert, ist ein Szenario mit ähnlichen Eigenschaften.

Touristenszenario Bereits heute ist es zum Beispiel in Berlin möglich, mit seinem PDA oder Smartphone ein Touristeninformationssystem zu nutzen, das Auskunft über die wichtigsten Sehenswürdigkeiten der Stadt gibt [mob10]. Auch in der Forschung werden die Möglichkeiten solcher mobilen Informationssysteme untersucht [HB06]. Eine mögliche Erweiterung besteht darin, dass die Touristen nicht nur lesend auf die Daten zugreifen, sondern das Informationssystem mit eigenen Bewertungen und aktuellen Neuigkeiten erweitern können. Diese Änderungen werden anschließend per ad-hoc Funkverbindung zu den Geräten anderer Touristen repliziert, so dass alle Touristen möglichst aktuelle Daten besitzen. Eine denkbare Anwendung dafür ist eine Art Wiki, in dem es zu jeder Sehenswürdigkeit einen Artikel gibt, der von Touristen gelesen, aber auch geändert werden kann. Im Touristenszenario ist eine besonders hohe Dynamik zu erwarten: Die Touristen nutzen das Informationssystem an einem Ort häufig nur für einen kurzen Zeitraum und außerhalb des Systems gibt es im Gegensatz zu den anderen Szenarien keine Gruppenzugehörigkeit.

Campusszenario Eine ähnliche Anwendung ist in einem Campusszenario denkbar, in dem Studenten mit Hilfe eines Wikis Informationen zu Übungsblättern, Vorlesungsinhalten und anderen Neuigkeiten austauschen können. Die Verteilung der geänderten Daten geschieht dabei wie beim Touristenszenario über das MANET, das sich zwischen den mobilen Geräten der Studenten aufspannt.

Im Rahmen dieser Arbeit werden nur die reinen MANET-Szenarien betrachtet, es ist jedoch in vielen Fällen anzunehmen, dass diese Szenarien jeweils nur temporär (wenige Stunden oder Tage) bestehen und früher oder später eine Verbindung zu einem Festnetz besteht. So kann es zum Beispiel im Katastrophenszenario einen Server in einer Zentrale geben, mit dem sich die Replikationsteilnehmer wöchentlich oder täglich abgleichen. Auch bei dem Touristen- und dem Campusszenario ist solch ein Abgleich mit einem Server denkbar. Trotzdem ist die Replikation in der Zeit, in der die Teilnehmer vom Festnetz getrennt sind, weiterhin notwendig, um die Daten möglichst aktuell und konsistent zu halten.

2.2 Replikation

In diesem Abschnitt werden zentrale Sachverhalte und Begriffe definiert und anhand von Beispielen verdeutlicht. Dabei wird speziell auf die Aspekte der Replikation in mobilen Szenarien eingegangen.

Ein **Replikationssystem** ist ein verteiltes System, das aus mehreren lokal gespeicherten Kopien einer Datenmenge besteht. Im Ursprungszustand sind alle Kopien der Datenmenge identisch. Wird die Replikation innerhalb eines Datenbanksystems durchgeführt, so spricht man von einem **Datenbankreplikationssystem** (auch *repliziertes Datenbanksystem*). Die lokalen Systeme werden jeweils als **Replikationsteilnehmer** bezeichnet. Die Gesamtheit aller Replikationsteilnehmer eines replizierten Systems wird als **Replikationsgruppe** bezeichnet. In herkömmlichen Replikationssystemen sind die Replikationsteilnehmer meist Server, die durch ein LAN oder WAN miteinander verbunden sind. In mobilen Szenarien sind die Replikationsteilnehmer mit lokalen Systemen/Datenbanken ausgestattete mobile Geräte wie PDAs, Smartphones oder Laptops, die per WLAN miteinander verbunden sind.

Jeder Replikationsteilnehmer besitzt einen **Replikationsmanager** (RM), der die Replikation koordiniert und eine **lokale Datenbank** (bzw. lokalen Speicher), die eine Menge von Datenelementen enthält. Ein **Datenelement** ist die kleinste Einheit der Datenbank. Eine verteilte Kopie eines Datenelements wird als **Replikat** bezeichnet. Die Gesamtheit der Zustände aller Datenelemente eines Replikationsteilnehmers wird **lokaler Datenbankzustand** (bzw. lokaler Systemzustand) genannt.

Die Replikationsteilnehmer führen Operationen auf den Daten aus. In diesem Kapitel verwenden wir den Begriff **Operation** als zusammenfassende Umschreibung für alle Arten von Datenänderungen. Dies vereinfacht den Vergleich der unterschiedlichen Replikationsansätze. Um Verwechslungen zu vermeiden, wird eine Operation, die genau einen Lese- oder Schreibzugriff auf einem einzelnen Datenelement durchführt, als **Datenbankoperation** bezeichnet. Eine geordnete Folge von Operationen wird als **Operationshistorie** bezeichnet.

Ein Replikationsteilnehmer, der Änderungen auf den Datenelementen durchführen darf, wird als **Master** bezeichnet. Teilnehmer, die kein Master sind dürfen nur lesend auf die Daten zugreifen. Ein Replikationssystem mit mehr als einem Master wird als **Multimaster** System bezeichnet. Ein Replikationssystem mit genau einem Master wird als **single-Master** System bezeichnet. Analog

spricht man von single-Master und Multimaster Replikation. Single-Master Systeme sind vom Aufbau einfach, da sie nur die Verteilung der Daten regeln müssen. Multimaster Systeme sind wesentlich komplexer. Durch die Möglichkeit von nebenläufigen Schreibzugriffen ist die Vermeidung von Dateninkonsistenzen für das Replikationssystem wesentlich aufwändiger. Im Rahmen dieser Arbeit betrachten wir Multimaster Replikation.

2.2.1 Zielsetzung

Das Hauptziel der Replikation ist, die Verfügbarkeit der Daten eines Datenbanksystems zu erhöhen. Die **Verfügbarkeit** bezeichnet die Wahrscheinlichkeit, mit der die Daten eines Systems zu einem bestimmten Zeitpunkt zur Verfügung stehen. Aufgrund von Netz- und Systemfehlern ist die Verfügbarkeit eines einzelnen Systems eingeschränkt. Durch das Replizieren der Daten kann diese Einschränkung reduziert werden. In mobilen Netzen ist der Aspekt der Verfügbarkeit besonders wichtig, da die Teilnehmer häufiger für eine gewisse Zeit vom Netz getrennt sind. Ebenso wird Replikation häufig verwendet, um die *Leistung* eines Datenbanksystems zu steigern. Typische Beispiele dafür sind verteilte Datenbanksysteme [OV99] oder das Domain Name System (DNS) [LA06], in denen die Zugriffslast verteilt wird. Aufgrund der Parallelisierung von Zugriffen können der Durchsatz und die Antwortzeiten verbessert werden. Zusammenfassend resultiert dies im *Zielkonflikt* der Replikation [MS06], der in Abbildung 2.1 dargestellt ist.

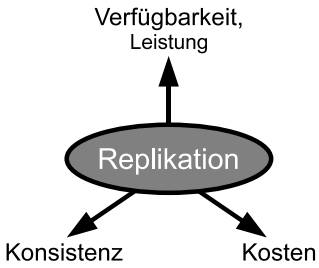


Abbildung 2.1: Zielkonflikt der Replikation: Replikationssysteme erhöhen die Verfügbarkeit der Daten und die Leistung eines Systems. Dafür müssen die Daten der einzelnen Replikationsteilnehmer in einem konsistenten Zustand gehalten werden. Dies verursacht zusätzliche Kosten.

Im Hinblick auf diese Problemstellung gibt es grundsätzlich zwei Lösungsansätze: **synchrone** oder **asynchrone** Aktualisierung der Daten. Bei der synchronen Aktualisierung ist die Aktualisierung der Daten der anderen Replikationsteilnehmer ein Teil der Operation selbst. Dadurch wird eine hohe Konsistenz der Daten gewährleistet, es müssen jedoch für die Ausführung der Operation alle Replikationsteilnehmer zur Verfügung stehen. Bei der asynchronen Aktualisierung wird die eigentliche Verteilung der Daten unabhängig von der Ausführung der Operationen durchgeführt. Somit

müssen bei der Ausführung der Operation nicht alle Replikationsteilnehmer zur Verfügung stehen, es können jedoch zeitweise Inkonsistenzen zwischen den Daten der einzelnen Teilnehmer auftreten. Nach Abschluss der Synchronisation sind die Daten aller Teilnehmer jedoch wieder konsistent.

Außer der Aktualisierung der Daten spielt auch die Art der Synchronisation eine entscheidende Rolle. Die zwei grundlegenden Ansätze für die Synchronisation von Daten werden im nächsten Abschnitt beschrieben.

2.2.2 Synchronisationsverfahren

Wenn ein repliziertes Datenbanksystem den Schreibzugriff für mehr als einen Teilnehmer zulässt (multimaster System), so muss es die nebenläufigen Schreibzugriffe der einzelnen Operationen der Teilnehmer koordinieren. Ohne diesen Kontrollmechanismus können parallele Schreibzugriffe zu einer dauerhaften Inkonsistenz der Daten führen. Es gibt zwei Ansätze für einen solchen Kontrollmechanismus: pessimistische und optimistische Synchronisation. *Pessimistische Synchronisation* wird in Festnetzen mit hoher Konnektivität verwendet und gewährleistet eine hohe Datenkonsistenz. *Optimistische Synchronisation* eignet sich eher für Netze mit einer geringen Konnektivität und gewährleistet eine höhere Verfügbarkeit der Daten. Heutige Datenbanksysteme (z. B. Oracle) stellen meistens beide Verfahren zur Verfügung. Entsprechend den Anforderungen an die Konsistenz und Verfügbarkeit der Daten wird dann eines der beiden Verfahren genutzt.

Klassische Replikationsszenarien zeichnen sich durch eine hohe Konnektivität der einzelnen Teilnehmer aus. Diese befinden sich meist in einem stabilen Festnetz mit geringen Latenzzeiten. Ziel des Synchronisationsverfahrens ist dabei die Gewährleistung einer single-Copy (Einzelexemplar) Konsistenz, die den Replikationsteilnehmern den Eindruck vermittelt, nur auf einer einzelnen Kopie der Daten zu arbeiten (siehe [BHG87]). Um dies zu gewährleisten, wird als Verfahren **pessimistische Synchronisation** benutzt.

Der Ansatz dieses Verfahrens besteht darin, *im Voraus* zu überprüfen, ob nebenläufige Datenzugriffe auf den Daten vorliegen. Dabei müssen bestehende Datenzugriffe z. B. durch Sperrverfahren geschützt werden. Das heißt, während ein Teilnehmer auf verteilte Daten schreibend zugreift, ist der Zugriff auf diese Daten für andere Teilnehmer nur eingeschränkt oder gar nicht möglich. Erst wenn der Zugriff beendet ist und enthaltene Änderungen an die anderen Replikationsteilnehmer verteilt wurden, wird die Sperre aufgehoben. Für die verschiedenen Anforderungen an die Konsistenz und Verfügbarkeit der Daten gibt es eine Vielzahl von Sperrverfahren [HTKR05] auf die in dieser Arbeit jedoch nicht weiter eingegangen wird.

Dadurch, dass bei pessimistischer Synchronisation der nebenläufige Datenzugriff im Voraus überprüft wird, können in Netzen mit geringer Konnektivität (Internet, MANETs) diverse Probleme auftreten. Das Hauptproblem sind die unvorhersehbaren Latenzzeiten großer und dynamischer Netze. Bei lang anhaltenden Netzunterbrechungen kann dies bei sperrbasierten Verfahren zu Blockierungen von unbestimmter Dauer führen. Bei anderen pessimistischen Verfahren kann die Vorabprüfung des

Datenzugriffs entsprechend verzögert werden. Ebenso stellt die Skalierbarkeit bei der Verwendung pessimistischer Synchronisation ein Problem dar. Die Leistung des Systems und die Datenverfügbarkeit werden bei zunehmender Größe der Replikationsgruppe stark reduziert, da bei jedem Datenzugriff eine verteilte Überprüfung auf nebenläufige Zugriffe durchgeführt werden muss. Aus diesem Grund verwenden viele Internetdienste (Usenet [SL98] oder DNS [LA06]) und mobile Datenbanksysteme optimistische Synchronisation [Bar03]. Auch das gemeinsame und verteilte Arbeiten (z. B. Programmieren) ist meist effizienter, wenn Sperren vermieden werden. Oft ist es besser, gelegentlich auftretende Konflikte zu lösen, als den Zugriff anderer Benutzer zu sperren [Ced10].

Im Gegensatz zum pessimistischen Verfahren wird bei **optimistischer Synchronisation** die Synchronisation nicht während des lesenden bzw. schreibenden Datenzugriffs einer Operation, sondern erst *danach* durchgeführt. Dies basiert auf der „optimistischen“ Annahme, dass nebenläufige Zugriffe auf dieselben Daten nur selten auftreten. Datenzugriffe werden asynchron an die anderen Replikationsteilnehmer verteilt. Die aufgrund nebenläufiger Datenzugriffe auftretenden Probleme werden erst behandelt, wenn sie nach der Verteilung der Daten erkannt werden.

Das optimistische Synchronisationsverfahren bietet im Gegensatz zur pessimistischen Synchronisation einige Vorteile. Da die Synchronisation hierbei erst nach der lokalen Ausführung einer Operation durchgeführt wird, kann der Replikationsteilnehmer auf seine lokalen Daten jederzeit zugreifen. Aufgrund dessen ist dieser Ansatz wesentlich unempfindlicher gegenüber höheren Latenzzeiten oder geringer Konnektivität als dies bei pessimistischer Synchronisation der Fall ist. Somit eignet sich optimistische Synchronisation besonders für dynamische Netze wie MANETs. Aufgrund des geringeren Synchronisationsaufwandes skaliert dieses Verfahren auch bei steigender Größe der Replikationsgruppe. Ebenso ermöglicht es mobilen Replikationsteilnehmern ein autonomes Arbeiten, was gerade in mobilen Netzen aufgrund der häufigen Netzunterbrechungen ein wichtiges Kriterium ist.

Die hohe Verfügbarkeit der Daten bei der Verwendung optimistischer Synchronisation führt jedoch auch zu Problemen. Operationen werden bei optimistischer Synchronisation erst nach der Ausführung synchronisiert. Das kann dazu führen, dass Operationen nebenläufig auf die gleichen Daten zugreifen und somit ein Konflikt entsteht. Dieser muss aufgelöst werden und die Daten der Replikationsteilnehmer müssen entsprechend angepasst werden. Die Replikationsteilnehmer sehen dann zum Teil einen temporär unterschiedlichen Zustand der Daten. Deshalb eignen sich optimistische Synchronisationsverfahren nur für Applikationen, die einen zeitweilig inkonsistenten Zustand der Daten erlauben. Ebenso ist optimistische Synchronisation für Szenarien mit einem hohen Anteil von Schreibzugriffen ungeeignet, da hierbei eine erhöhte Gefahr für die Bildung von Konflikten besteht.

Synchronisationsverfahren spielen bei der Replikation eine so zentrale Rolle, dass sie sehr häufig auch als Namensgeber für das gesamte Replikationsverfahren benutzt werden. Aus diesem Grund sprechen wir im Folgenden immer von **pessimistischer** und **optimistischer Replikation**.

2.2.3 Replikationsarchitekturen

Abhängig von den jeweiligen Randbedingungen eines Szenarios gibt es unterschiedliche Ansätze für die Architektur eines Replikationssystems. Die drei grundlegenden Architekturen für Replikation, wie sie in mobilen Szenarien häufig verwendet werden, sind in Abbildung 2.2 dargestellt.

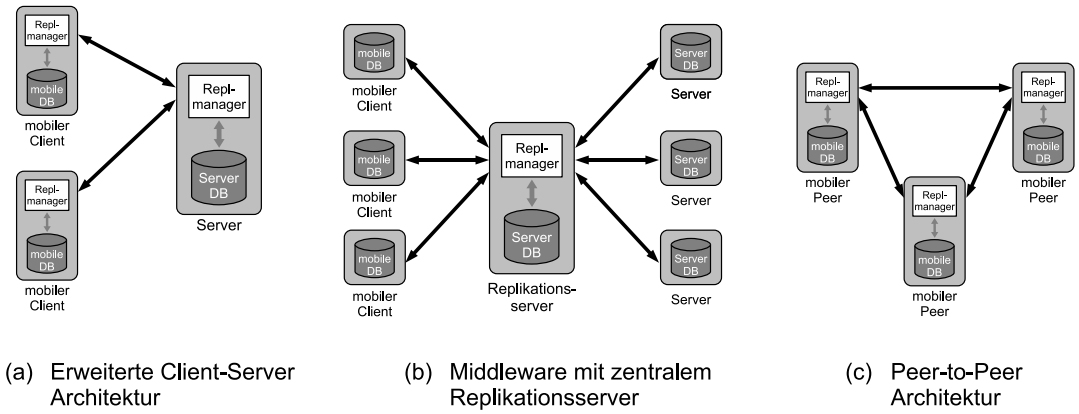


Abbildung 2.2: Replikationsarchitekturen

Die erweiterte **Client-Server Architektur** erlaubt eine direkte Kommunikation zwischen den Klienten und dem Server. Im Gegensatz zu den herkömmlichen Client-Server Architekturen ist es dem Klienten auch ohne eine Verbindung zum Server möglich, auf seinen lokalen Datenbestand zuzugreifen. Wenn die Verbindung wieder besteht, synchronisiert der Replikationsmanager die Daten von Klient und Server erneut, da in der Zwischenzeit auf beiden Seiten möglicherweise Datenänderungen vorgenommen wurden. Dieser Ansatz ist weit verbreitet und wird von vielen aktuellen kommerziellen Produkten wie Oracle Lite, Microsoft SQL Server CE oder Sybase SQL Anywhere verwendet.

Die **Middleware Architektur mit einem zentralen Replikationsserver** verwendet einen Replikationsserver, der die Replikation zwischen den Klienten und den Servern verwaltet. Durch die indirekten Verbindungen zwischen den Klienten und den Servern wird die Replikation durch den zentralen Replikationsserver für die Teilnehmer transparent gehalten. Außerdem können durch den Replikationsserver zusätzlich Sicherheits- und Lastverteilungsaspekte verwaltet werden.

Die **Peer-to-Peer Architektur** nutzt keinen Server. Dieser Ansatz bietet besonders in MANETs Vorteile gegenüber zentralisierten Architekturen. Aufgrund der hohen Dynamik von MANETs und der Unzuverlässigkeit mobiler Knoten, kann die für einen Server benötigte Zuverlässigkeit nicht gewährleistet werden. Würde sich ein Server auf einem mobilen Gerät befinden, so wäre die gesamte Replikation eines Systems dadurch eingeschränkt. Im Rahmen dieser Arbeit verwenden wir deshalb eine Peer-to-Peer Architektur.

2.3 Optimistische Replikation

In [SS05, Sai01] definieren Saito und Shapiro sechs Hauptelemente der optimistischen Replikation. Diese Elemente dienen zur Klassifikation der verschiedenen Ansätze und Systeme und werden in diesem Abschnitt beschrieben. Zu Beginn wird der allgemeine Ablauf der optimistischen Replikation erklärt.

2.3.1 Ablauf

Um ein Verständnis für die Arbeitsweise optimistischer Replikation zu bekommen, geben wir eine kurze beispielhafte Erklärung des Arbeitsablaufs, der in Abbildung 2.3 dargestellt ist.

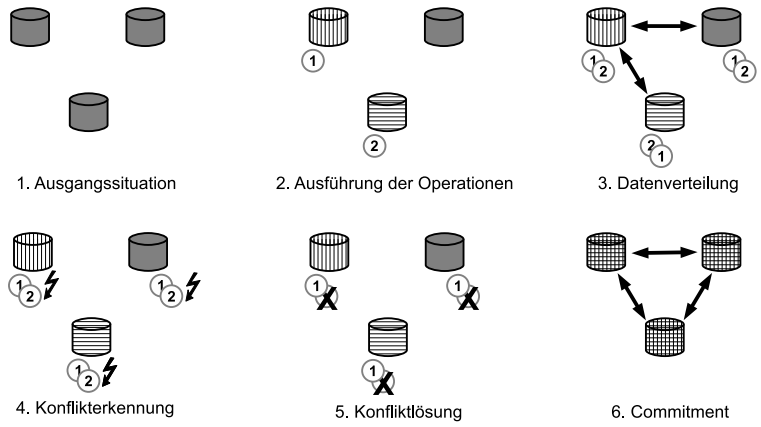


Abbildung 2.3: Ablauf der optimistischen Replikation [SS05]

1. Bei der Initialisierung des Systems wird gewährleistet, dass alle Replikationsteilnehmer den gleichen lokalen Datenbankzustand besitzen.

2. Die Replikation wird zunächst durch die lokale Ausführung von Operationen angestoßen. Im Beispiel wird jeweils eine Operation auf den lokalen Datenbanken zweier Teilnehmer ausgeführt. Die Ausführung ist jeweils nur lokal und somit noch vorläufig.

3. Im nächsten Schritt werden die Operationen möglichst an alle Teilnehmer der Replikationsgruppe verteilt, so dass alle Replikationsteilnehmer die gleichen Änderungsoperationen kennen. In WANs und mobilen Netzen wird dies meistens mit speziellen Broadcastprotokollen durchgeführt (siehe Abschnitt 2.1.2).

4. Nachdem die Operationen verteilt wurden, wird lokal überprüft, ob Konflikte zwischen den Operationen bestehen.

5. Wenn Konflikte erkannt werden, müssen diese gelöst werden, um wieder einen konsistenten Zustand herzustellen. Grundsätzlich gibt es dabei die Möglichkeit, die Ausführung der am Konflikt beteiligten Operationen zu berücksichtigen oder eine (ggf. mehrere) der in Konflikt stehenden Operationen abzubrechen. Diese vorläufige Konfliktlösung wird zunächst bei jedem Replikationsteilnehmer lokal durchgeführt.

6. Damit die Änderungen der Operationen dauerhaft werden, müssen sich die Replikationsteilnehmer gegenseitig abstimmen oder im Vorfeld auf ein vordefiniertes Commitverfahren geeinigt haben. Dadurch wird eine global konsistente Entscheidung getroffen und die vorläufigen Konfliktlösungen können endgültig festgeschrieben werden.

2.3.2 Art der verteilten Daten: Operation/Zustand

Ein wichtiges Unterscheidungsmerkmal für optimistische Replikationssysteme ist die Art der Verteilung von lokalen Änderungen an die anderen Teilnehmer der Replikationsgruppe. Einige Systeme übertragen nur die geänderten Datenelemente und werden deshalb **State-Transfer** (Zustandsübertragung) Systeme genannt. Diese Systeme stellen einen konsistenten Zustand her, indem sie einfach den aktuellen Stand der Elemente an alle Teilnehmer verteilen.

Andere Systeme übertragen nicht die geänderten Elemente, sondern die Operationen, die die Daten verändert haben und werden deshalb **Operation-Transfer** (Operationsübertragung) Systeme genannt. Dieser Ansatz ist komplexer und benötigt eine Operationshistorie, um einen konsistenten Zustand herzustellen. Bei diesem Verfahren ist eine differenziertere Lösung von auftretenden Konflikten möglich als beim State-Transfer [Sai01]. Die Verwendung von Transaktionen, die mehrere Update-Operationen beinhalten, kann in einem State-Transfer System zu Dateninkonsistenzen führen [HB99]. Ändert z. B. eine Transaktion mehrere Datenelemente einer Datenbank, so werden im Falle eines Konflikts nur die in Konflikt stehenden Datenelemente zurückgesetzt, jedoch nicht die gesamte Transaktion, wodurch die Atomarität der Transaktion verletzt wird [HJRW07]. Diese Inkonsistenzen können in State-Transfer Systemen häufig nur noch manuell behoben werden [HB99].

Die Bezeichnungen State-Transfer und Operation-Transfer beschreiben nur das logische Konzept der Datenübertragung. Die tatsächliche Implementierung der Datenübertragung wird dadurch nicht beschrieben. So verwendet Oracle 10g bei der Replikation der Operationen z. B. sogenannte Logical Change Records (LCRs). Diese LCRs beinhalten die veränderten Daten der einzeln Zeilen, die von einer Transaktion bearbeitet wurden. Die Zuordnung der LCRs zu einer bestimmten Transaktion wird ebenfalls übertragen, so dass die LCRs einer Transaktion atomar ausgeführt werden können. Somit ist dieses Verfahren konzeptionell ein Operation-Transfer Verfahren, obwohl rein technisch nur veränderten Zustände übertragen werden [Urb08].

In kommerziellen mobilen DBS Lösungen wie z. B. Oracle Lite oder Sybase SQL Anywhere wird als Verfahren häufig State-Transfer verwendet. Ein bekannter Vertreter aus der Gruppe der Operation-Transfer Systeme ist das Replikationssystem Bayou [DPS+94]. Analog zu den Systemen

spricht man auch von *State-Transfer* und *Operation-Transfer Replikation*.

2.3.3 Datenverteilung

Nachdem eine Operation lokal ausgeführt wurde, wird sie an die anderen Teilnehmer der Replikationsgruppe verteilt. Broadcastprotokolle regeln *wann* und *wie* die Daten der Operation zwischen den Teilnehmern ausgetauscht werden.

In statischen Netzen verwenden die Broadcastprotokolle oft eine feste Kommunikationstopologie wie z. B. einen aufspannenden Baum oder eine sternförmige Topologie. In Sensornetzen werden diese Ansätze ebenso verfolgt, da nach anfänglicher Installation der Sensoren die Netztopologie meist statisch bleibt. Während diese genannten Verfahren in statischen Netzen effizient arbeiten, sind sie größtenteils nicht für dynamische und mobile Netze geeignet.

Ein weiterer Ansatz ist die Verwendung von semistrukturierten Kommunikationstopologien, die sich für Netze mit einer leichten bis mittleren Dynamik eignen. Ein Beispiel dafür ist das Replikationssystem Roam [Rat98], welches das Netz in kleinere Unterbereiche, sogenannte *Wards*, einteilt.

In unstrukturierten Netzen wie MANETs bietet sich eine **epidemische Datenverteilung** an, bei der die Daten schrittweise von einem Teilnehmer zum nächsten verteilt werden (siehe Abschnitt 2.1.2). Diese Art der Datenverteilung stellt keine Anforderungen an die Netztopologie und wird von einigen Replikationssystemen wie Bayou [DPS+94] oder Gossip [LLSG92] verwendet. Solch eine epidemische Datenverteilung folgt den gleichen Grundprinzipien echter Epidemien, wie z. B. der Ansteckung bei Viruserkrankungen. Dadurch ist sie sehr robust und besonders für Netze mit dynamischer Topologie geeignet.

2.3.4 Konflikte

Bei der Verwendung von optimistischer Multimaster Replikation können Konflikte auftreten, denn diese können im Gegensatz zu pessimistischen Ansätzen nicht im Vorhinein verhindert werden. Konflikte können in zwei Kategorien eingeteilt werden: **syntaktische** und **semantische Konflikte**. Bei der Berücksichtigung von syntaktischen Konflikten wird die nebenläufige Ausführung von Operationen auf den gleichen Daten betrachtet. Bei der Berücksichtigung von semantischen Konflikten wird die Semantik der Operationen betrachtet. Nehmen wir zum Beispiel ein Bemerkungsfeld in einer verteilten Datenbank. Bei einer syntaktischen Konflikterkennung würde jede nebenläufige Änderung dieses Feldes als Konflikt interpretiert werden. Bei einer semantischen Konflikterkennung muss das nebenläufige Hinzufügen zweier unterschiedlicher Hinweise im Bemerkungsfeld nicht zwingend als Konflikt erkannt werden und es können beide Änderungen übernommen werden. Da im Rahmen dieser Arbeit die syntaktischen Konflikte im Vordergrund stehen, werden wir nur diese Art der Konflikte im folgenden Abschnitt thematisieren.

Zunächst definieren wir eine Ordnungsrelation und den Begriff „Nebenläufigkeit“. Eine intuitive

Definition einer partiellen Ordnungsrelation für verteilte Ereignisse[3] wird von Lamport in [Lam78] gegeben.

Kausale Ordnungsrelation Gegeben seien zwei Operationen a und b, von denen jeweils eine bei Replikationsteilnehmer s_1 (Operation a) und eine bei Replikationsteilnehmer s_2 (Operation b) lokal ausgeführt wird. Die lokale Ausführung einer Operation wird durch den Zeitpunkt definiert, an dem die Operation bei einem Teilnehmer vollständig ausgeführt und abgeschlossen wurde. Die Operation a hat *kausal vor* Operation b *stattgefunden* ($a < b$), wenn eine der folgenden Bedingungen gilt:

- $s_1 = s_2$ und a wurde vor b ausgeführt.
- $s_1 \neq s_2$ und b wurde ausgeführt, nachdem s_2 die Operation a bereits erhalten und ausgeführt hat.
- Es existiert eine andere Operation c mit: a hat kausal vor c stattgefunden und c hat kausal vor b stattgefunden.

Mit Hilfe der kausalen Ordnungsrelation ist es nun möglich, kausale Parallelität bzw. Nebenläufigkeit zu definieren.

Nebenläufigkeit Gegeben seien zwei Operationen a und b. Hat weder Operation a kausal vor Operation b, noch Operation b kausal vor Operation a stattgefunden, so wurden die Operationen a und b *zueinander nebenläufig* (oder auch *kausal parallel*) *ausgeführt*, in Zeichen $a \parallel b$.

Ein Beispiel für eine nebenläufige Ausführung zweier Operationen ist in Abbildung 2.4 dargestellt.

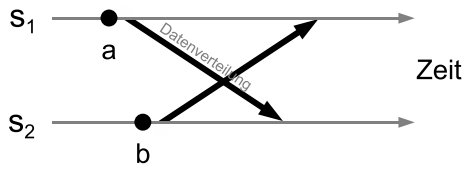


Abbildung 2.4: Nebenläufige Ausführung der Operationen a und b

Der Teilnehmer s_1 führt die Operation a lokal aus und sendet diese an den Teilnehmer s_2. Zeitlich versetzt führt s_2 die Operation b aus und sendet diese an s_1. Beide Operationen erreichen jeweils den anderen Teilnehmer, nachdem die lokale Operation bereits ausgeführt wurde. Somit wurden die

[3] In dieser Arbeit werden nur die Ausführungen von Operationen als Ereignisse betrachtet. Weitere Ereignisse, wie z. B. das Versenden und Empfangen einer Operation sind für die in dieser Arbeit verwendeten Verfahren nicht relevant und werden deshalb nicht weiter berücksichtigt.

Operationen a und b zueinander nebenläufig ausgeführt. Die Dauer dieses Sendevorgangs kann für die verschiedenen Arten von Netzen sehr unterschiedlich sein (siehe 2.1). Wie man an dem erläuterten Beispiel erkennen kann, hat dies eine starke Auswirkung auf die Entstehung nebenläufiger Operationen.

Kommen wir zu der allgemeinen Definition eines Konflikts für optimistische Replikation mit syntaktischer Konflikterkennung. Eine spezielle Definition eines Konflikts in unserem Systemmodell geben wir in Abschnitt 2.4.

Konflikt Gegeben seien zwei Operationen a und b. Die Operationen a und b *stehen miteinander in Konflikt*, wenn folgende Bedingungen gelten:

- Es gilt $a \parallel b$, d.h. a und b wurden zueinander *nebenläufig* ausgeführt und
- die Operationen a und b haben auf den *gleichen lokalen Daten gearbeitet* [4].

2.3.5 Konflikterkennung

Für die Erkennung von Konflikten gibt es zwei unterschiedliche Ansätze: Eine weit verbreitete Methode ist die Verwendung von Versionsvektoren, die zum Teil auch *Versionsuhren* oder *Zeitstempelvektoren* genannt werden. **Versionsvektoren** sind eine Weiterentwicklung von Lamport Uhren [Lam78] und wurden von Mattern [Mat89] und Fidge [Fid91] eingeführt. Mit Hilfe von Versionsvektoren lässt sich die nebenläufige Ausführung von Operationen feststellen. Zusammen mit den von der Operation zugegriffenen Daten können somit Konflikte erkannt werden. Versionsvektoren werden in ihrer ursprünglichen Form oder nur leicht verändert von einigen Replikationssystemen verwendet, so z. B. in [Rat98, DPS+94].

Ein weiterer Ansatz zur Konflikterkennung ist die explizite Darstellung der kausalen Abhängigkeiten von Operationen in **Vorgängerlisten**. Hierbei verweist jede Operation auf die kausal vorangehenden Operationen. Vorgängerlisten werden sowohl in der Replikation [Hup09, KRSD01] als auch im Bereich der Gruppenkommunikation [FLS01] eingesetzt.

2.3.6 Konfliktlösung

Nachdem ein Konflikt vom Replikationssystem erkannt wurde, muss dieser gelöst werden, damit keine dauerhaften Inkonsistenzen bestehen bleiben. Konflikte können vom System automatisch oder manuell durch den Benutzer gelöst werden. Bei letzterem Ansatz muss die Konfliktlösung danach an die anderen Teilnehmer gesendet werden, die ihrerseits zum Teil Vetorechte besitzen können. In einem mobilen und somit dezentralen System resultiert dies häufig in langen Verzögerungen. Somit

[4] Diese Bedingung wird bewusst sehr allgemein gehalten, um damit alle verschiedenen Arten von Replikationssystemen zu berücksichtigen. Eine Spezialisierung dieser Definition für unser Systemmodell wird in Abschnitt 2.4.1 gegeben

ist eine manuelle Konfliktlösung in MANET-Szenarien meist nicht praktikabel. Bei der Auflösung eines Konflikts gibt es zwei Möglichkeiten.

Stehen einem Replikationssystem Informationen über die Semantik der Operationen zur Verfügung, so ist es in vielen Fällen möglich, die *Ausführung beider in Konflikt stehenden Operationen zu berücksichtigen*. Wie dies vom Replikationssystem realisiert wird ist sehr unterschiedlich und hängt von der Semantik der Operationen ab. Diese reicht von einfachen Kommutativitätsregeln bis hin zur Beschreibung durch komplexe SQL Statements [TTP+95]. In einem System wie CVS [Ced10], in dem die Operationen textuellen Änderungen entsprechen, werden bei Konflikten die jeweiligen Änderungen z. B. zeilenweise in einem Text zusammengefügt. Diese Art der Konfliktlösung wird häufig als **Merge** bezeichnet.

Ohne genaueres Wissen über die Semantik der Operationen kann ein Replikationssystem Konflikte nur mit Hilfe von *Operationsabbrüchen* auflösen. Dabei ist es wichtig, bei allen Teilnehmern dieselben Operationen abzubrechen, da sich ansonsten Inkonsistenzen ergeben. Um dies zu gewährleisten, werden die in Konflikt stehenden Operationen geordnet. Eine global konsistente Konfliktlösung kann dabei nur gewährleistet werden, wenn die Operationen bei allen Replikationsteilnehmern in der gleichen Reihenfolge geordnet sind (totale Ordnung). Aufgrund der gleichen Ordnung kann sichergestellt werden, dass bei jedem Teilnehmer die gleichen Operationen abgebrochen werden, wie bei allen anderen Teilnehmern. Meistens werden Echtzeit-Zeitstempel oder Prioritätswerte verwendet, um die in Konflikt stehenden Operationen zu ordnen.

2.3.7 Konsistenz

Die Hauptaufgabe der Replikation ist es, die verteilten Daten in einem konsistenten Zustand zu halten. In diesem Abschnitt werden wir **Eventual Consistency** erläutern, da sie im Bereich optimistischer Replikation eine zentrale Rolle einnimmt. Weiterhin werden Commitverfahren optimistischer Replikation beschrieben, die benötigt werden, um einen global einheitlichen Zustand der Daten herzustellen.

Eventual Consistency

Häufig wird Eventual Consistency (zum Teil auch als Weak Consistency bezeichnet [Gol92]) nicht genau definiert, sondern nur kurz umschrieben. Da diese Umschreibung sehr zum Verständnis der Definition beiträgt, wird sie einleitend gegeben: *Die Gewährleistung von **Eventual Consistency** ist gegeben, wenn alle Replikationsteilnehmer nach einer bestimmten Zeit, in der keine Operationen mehr ausgeführt werden, den gleichen lokalen Zustand besitzen.*

Für die Definition der Eventual Consistency muss zunächst die Äquivalenz zweier Operationshistorien (siehe S. 14) definiert werden.

Äquivalenz von Operationshistorien Gegeben seien zwei Operationshistorien O_1 und O_2, die die

Ausführungsreihenfolge der einzelnen Operationen bestimmen. Die beiden Operationshistorien O_1 und O_2 sind *äquivalent*, wenn sie bei der Ausführung auf demselben Ausgangszustand denselben Endzustand erreichen. Aufgrund der Konfliktlösung ist es erlaubt, dass abgebrochene Operationen in den Historien enthalten sind. Eine Operation o, die bereits abgebrochen wurde, wird als $\bar{o}$ bezeichnet.

Darauf aufbauend wird Eventual Consistency wie folgt definiert (vgl. [SS05]).

Eventual Consistency Unter der Voraussetzung, dass alle Replikationsteilnehmer denselben Ausgangszustand besitzen, wird *Eventual Consistency* gewährleistet, wenn alle folgenden Bedingungen gelten:

- Zu jedem Zeitpunkt und für jeden Replikationsteilnehmer existiert ein Präfix einer Operationshistorie, das äquivalent zu einem Präfix einer Operationshistorie jedes anderen Replikationsteilnehmers ist. Dieses wird als *festgeschriebenes Präfix* (engl. committed prefix) bezeichnet.
- Das festgeschriebene Präfix jedes Replikationsteilnehmers wächst monoton über die Zeit.
- Für alle nicht abgebrochenen Operationen in dem festgeschriebenen Präfix sind ihre Vorbedingungen erfüllt. Diese Bedingungen werden von jedem System individuell festgelegt.
- Für jede initiierte Operation o gilt, dass *irgendwann* entweder o oder $\bar{o}$ im festgeschriebenen Präfix enthalten ist.

Diese sehr allgemeine Definition erlaubt eine Reihe von unterschiedlichen Umsetzungen, da die Äquivalenz von Operationshistorien und die Definition der zu erfüllenden Vorbedingungen applikationsspezifisch sind. So ist z. B. in Bayou die explizite Angabe der Vorbedingungen vorgesehen, während bei einem System wie Usenet [SL98] keine Vorbedingung vorhanden ist und diese somit immer als erfüllt angenommen wird. Dies hat eine beliebige Reihenfolge der Artikel zur Folge. Damit reduziert sich die Gewährleistung von Eventual Consistency in diesem Fall nur auf eine garantierte Übertragung der Operationen.

Für eine weitere Unterteilung der unterschiedlichen Ansätze, die Eventual Consistency garantieren, ist es notwendig, das Ordnungskriterium der Operationshistorien zu betrachten. Die gebräuchlichsten Ordnungskriterien (*FIFO*, *kausal* und *total*) sind wie folgt definiert:

FIFO (First In First Out) Ordnung Alle Operationen, die von einem bestimmten Replikationsteilnehmer initiiert wurden, werden von allen anderen Replikationsteilnehmern in der gleichen Ausführungsreihenfolge ausgeführt, wie beim ursprünglichen Teilnehmer. Operationen, die von unterschiedlichen Replikationsteilnehmern ausgeführt wurden, können in beliebiger Reihenfolge ausgeführt werden.

Kausale Ordnung Operationen, die zueinander kausal[5] in Beziehung stehen, werden von allen Replikationsteilnehmern in der gleichen Reihenfolge ausgeführt. Nebenläufige Operationen können in beliebiger Reihenfolge ausgeführt werden.

Totale Ordnung Alle ausgeführten Operationen werden bei allen Replikationsteilnehmern in der gleichen Reihenfolge ausgeführt. Dies impliziert nicht automatisch eine kausale Ordnung, da eine totale Ordnung keine bestimmte Ordnungsrelation voraussetzt.

Betrachtet man eine total geordnete Menge von Transaktionen und führt diese nicht verzahnt aus, so ergibt sich für die Transaktionen lokal und global eine serielle Historie und das Serialisierbarkeitskriterium ist erfüllt.[6]

Um stärkere Ordnungskriterien zu erhalten, können Ordnungskriterien auch kombiniert werden. So wird von manchen Replikationssystemen sowohl eine kausale als auch eine totale Ordnung gewährleistet.

Commitment

Bisher haben wir vorgestellt, wie ein Replikationssystem Operationen verteilt und Konflikte erkennt und löst, um einen konsistenten Gesamtzustand herzustellen. Damit ein Replikationsteilnehmer diese bisher vorläufig ausgeführten Operationen jedoch endgültig festschreiben kann, muss sichergestellt sein, dass sich die Ausführungsreihenfolge dieser und aller vorangegangenen Operationen nicht mehr ändert. Um dies zu gewährleisten gibt es Commitverfahren, die das Festschreiben der Operationen regeln.

Bei einem Commitverfahren einigen sich die Replikationsteilnehmer auf einen zusammenhängenden Teil der Operationshistorie, der sich nicht mehr ändern kann. Dies geschieht mit Hilfe eines global eindeutigen Zeitpunkts (Commitpunkt), der an alle Teilnehmer verteilt wird. Es wird eine endgültige Entscheidung (Festschreiben/Abbrechen) für alle Operationen bis zum Commitpunkt getroffen und die bisherige Operationshistorie kann bis zu diesem Punkt gekürzt werden. Es gibt unterschiedliche Ansätze, um eine globale Commit Entscheidung zu treffen. Die Entscheidung kann von einer bestimmten Seite vorgegeben, durch eine Wahl bestimmt oder implizit im Hintergrund durchgeführt werden.

Die Vorgabe einer Commit Entscheidung durch eine einzelnen Seite wird als **Primary-based Commitment** bezeichnet. Dieses Verfahren wird z. B. von Bayou verwendet, wo ein vorher festgelegter Replikationsteilnehmer allen Operationen Sequenznummern zuweist und sich bei der Bestimmung des Commitpunktes auf diese bezieht [DPS+94]. Da dieser Ansatz auf einer zentralen Verwaltung basiert, ist er für die Replikation in MANETs weniger geeignet.

[5] basierend auf der kausalen Ordnungsrelation (siehe Abschnitt 2.3.4)

[6] Serialisierbarkeit ist ein Korrektheitskriterium für die Ausführung von verzahnten Transaktionen. In den von uns betrachteten Szenarien gibt es jedoch weder lokal noch verteilt eine verzahnte Ausführung von Transaktionen (siehe Abschnitt 2.4).

Das wahlbasierte Verfahren (**Quorum based Commitment**) wird z. B. vom Replikationssystem Deno [eKBF03] verwendet. Es erlaubt das Festschreiben einer Operationen nach der Durchführung einer zweiphasigen Abstimmung. Wenn eine Operation die Mehrheit der Stimmen erhält, kann sie festgeschrieben werden. Simulationsergebnisse [Kel99] zeigen, dass die Effizienz dieses Ansatzes mit dem eines single-Master Systems vergleichbar ist. Es ist fraglich, ob solch ein Ansatz auch für optimistische Replikation in MANETs verwendet werden könnte, da durch die Abstimmung eine hohe Netzlast hervorgerufen wird. Ebenso ist durch die häufige Partitionierung von MANETs die Bildung einer Mehrheit in der Regel problematisch.

Der letzte Ansatz ist eine **implizite Einigung** der Replikationsteilnehmer im Hintergrund. Die Grundidee dieses Ansatzes basiert darauf, dass sich jeder Replikationsteilnehmer den ihm bekannten Informationsstand (Operationen) der anderen Teilnehmer merkt. Aufgrund der Schnittmenge der jeweils bekannten Operationen aller Teilnehmer kann jeder Teilnehmer feststellen, welche Operationen er festschreiben darf. Ein Beispiel für solch ein Verfahren ist in [Gol92] beschrieben.

2.4 Systemmodell

In diesem Abschnitt werden die für diese Arbeit relevanten Voraussetzungen und Annahmen erläutert und das Systemmodell unseres Replikationsansatzes erklärt. Die interen Abläufe und Algorithmen des Systems werden in den entsprechenden Kapiteln (6, 7, 8) erläutert.

2.4.1 Replikationsmodell

Die folgenden Definitionen basieren alle auf den Standarddefinitionen aus der Datenbankliteratur [KE06, EN02]. Einige der an dieser Stelle definierten Begriffe wurden bereits in Abschnitt 2.2 für die allgemeine Diskussion der Grundlagen kurz eingeführt. In diesem Abschnitt werden diese Begriffe speziell für unser Modell definiert.

Replikationsgruppe Die *Replikationsgruppe* $R = \{s_1 ... s_n\}$ ist die Menge aller Replikationsteilnehmer. Die Replikationsgruppe ist eine *dezentral organisierte Gruppe* (engl. peer group), deren Mitglieder gleichberechtigt sind.

Replikationsmanager Jeder Replikationsteilnehmer s_k besitzt einen *Replikationsmanager* (RM), der die Replikation verwaltet. Jeder RM besitzt eine global eindeutige Kennung k, die er auf Basis der MAC-Adresse des mobilen Gerätes bestimmt, auf dem er sich befindet. Die lokale Datenbank eines Teilnehmers wird ebenfalls vom RM verwaltet.

Echtzeit-Zeitstempel Für die zeitliche Zuordnung von Ereignissen werden vom Replikationssystem die Zeitstempel der lokalen Echtzeituhr des jeweiligen Teilnehmers verwendet. Diese Echtzeit-Zeitstempel werden als global eindeutig angenommen.

Lokale Datenbank Eine *lokale Datenbank* D_k ist die Menge aller Datenelemente eines Replikationsteilnehmers s_k. Es gibt zwei Sichten auf die lokale Datenbank. Eine Sicht zeigt den aktuellen und vorläufigen Zustand und die andere Sicht den endgültig festgeschriebenen Zustand der Daten an (siehe unten).

Datenelement Ein *Datenelement* ist die kleinste Dateneinheit einer Datenbank, die gelesen, geschrieben, erstellt und gelöscht werden kann. Ein Datenelement e_i^k einer lokalen Datenbank D_k (des Teilnehmers s_k) besitzt zwei verschiedene Zustände. Erstens den aktuellen Zustand $val(e_i^k)$, der sich aufgrund der Ausführung aller aktuellen Änderungen ergibt. Darin sind auch nur vorläufig ausgeführte Änderungen erhalten. Zweitens den festgeschriebenen Zustand $cval(e_i^k)$, der sich aufgrund der Ausführung aller bereits festgeschriebenen Änderungen ergibt.

In Fällen, in denen die Unterscheidung mehrerer Elemente nicht notwendig ist, werden die Indizes i und k weggelassen.

Datenbankoperation Eine Datenbankoperation wird immer auf der lokalen Datenbank eines Teilnehmers ausgeführt. Es gibt vier unterschiedliche Arten von Operationen:

- Eine *Leseoperation* $r\,[e]$, die den aktuellen Wert $val(e)$ des Datenelements e aus der lokalen Datenbank liest. Die Operation $r_c\,[e]$ liest den festgeschriebenen Wert $cval(e)$. Dazu wird jeweils die entsprechende Sicht der lokalen Datenbank verwendet (siehe oben).
- Eine *Schreiboperation* $w\,[e]$, die den Wert $val(e)$ des Datenelements e in der lokalen Datenbank überschreibt.
- Eine *pre-Commit Operation* c, die eine lokale Ausführung einer Gruppe von Operationen (Transaktion) veranlasst. Durch die Ausführung der pre-Commit Operation wird die Sicht auf den vorläufigen Zustand der lokalen Datenbank verändert. Aufgrund entstehender Konflikte muss dieses pre-Commit unter Umständen zurückgenommen werden (Abbruch).
- Eine *Abbruchoperation*, die den Abbruch einer Transaktion veranlasst. Eine lokal abgebrochene Transaktion wird vom Replikationssystem nicht berücksichtigt.

Transaktion Eine Transaktion TA_j mit einer eindeutigen Kennung j ist eine *geordnete Menge von Datenbankoperationen, die atomar verteilt* wird. Das Write-Set einer Transaktion $\mathcal{W}(TA_j) \subseteq D_k$ enthält alle Datenelemente, die durch Schreiboperationen der Transaktion verändert wurden. Das Read-Set einer Transaktion $\mathcal{R}(TA_j) \subseteq D_k$ enthält alle Datenelemente, die durch Leseoperationen der Transaktion gelesen wurden. Transaktionen arbeiten nur auf lokalen Daten und werden erst verteilt, *nachdem* sie lokal ausgeführt (pre-Commit) wurden. Eine lokale Ausführung ist nur vorläufig gültig und wird durch eine globale Commit Entscheidung entweder endgültig festgeschrieben oder zurückgenommen. Jede Transaktion enthält ebenfalls einen

Echtzeit-Zeitstempel t_j, der den lokalen Ausführungszeitpunkt (Zeitpunkt des pre-Commit) der Transaktion angibt und eine lokal eindeutige Sequenznummer $n_j \in \mathbb{N}$. Die Echtzeit-Zeitstempel werden verwendet, um eine totale Ordnung aller Transaktionen der einzelnen Teilnehmer herzustellen. Durch die Ungenauigkeit von Echtzeit-Zeitstempeln muss dies global nicht notwendigerweise die Echtzeitreihenfolge widerspiegeln.

Aus der Sicht eines Teilnehmers werden die Transaktionen, die er selber initiiert hat als **lokale Transaktionen** bezeichnet. Transaktionen von anderen Teilnehmern werden als **entfernte Transaktionen** bezeichnet.

Konflikt[7] Gegeben seien zwei Transaktionen TA_i und TA_j. Die Transaktionen stehen miteinander in einem Konflikt, falls ein Schreib-Schreib Konflikt oder ein Lese-Schreib Konflikt zwischen ihnen besteht.

Es besteht ein *Schreib-Schreib Konflikt* zwischen den Transaktionen, falls sie kausal parallel ausgeführt werden und sich ihre Write-Sets schneiden, d. h. es gilt:

- $TA_i \parallel TA_j$ und
- $\mathcal{W}(TA_i) \cap \mathcal{W}(TA_j) \neq \emptyset$

Die Transaktionen TA_i und TA_j stehen miteinander in einem *Lese-Schreib Konflikt*, falls sie kausal parallel ausgeführt werden und sich das Write-Set der einen Transaktion mit dem Read-Set der anderen Transaktion schneidet, d. h. es gilt:

- $TA_i \parallel TA_j$ und
- $\mathcal{R}(TA_i) \cap \mathcal{W}(TA_j) \neq \emptyset$ oder $\mathcal{W}(TA_i) \cap \mathcal{R}(TA_j) \neq \emptyset$

Commitpunkt Ein *Commitpunkt* besitzt einen Echtzeit-Zeitstempel cp und markiert damit einen Zeitpunkt. Dieser Zeitpunkt wird verwendet, um festzulegen für welche Transaktionen eine Commit Entscheidung getroffen werden soll.

2.4.2 Verwendung von Transaktionen

Die Verwendung von Transaktionen im Gegensatz zu Operationen bietet mehr Möglichkeiten: Aufgrund der Atomarität von Transaktionen können Operationen gruppiert werden, so dass diese gemeinsam ausgeführt werden. Dadurch ist ebenfalls eine differenziertere Konflikterkennung möglich. Bei der Replikation von Operationen können nur Konflikte zwischen einzelnen Operationen erkannt werden. Probleme, die sich aufgrund der gemeinsamen Ausführung von Operationen ergeben, werden nicht erkannt. So kann bei der alleinigen Betrachtung von Operationen ein *Repeatable Read* oder die Vermeidung eines *Lost Update* (siehe [KE06]) nicht gewährleistet werden.

[7] Diese Definition ist eine Spezialisierung der in Abschnitt 2.3.4 gegebenen Definition.

Bei der Verwendung von Transaktionen und der Berücksichtigung von Lese-Schreib Konflikten (siehe oben) ist dies nicht problematisch: Transaktionen können nur kausal parallel ausgeführt werden, wenn sie entweder auf unterschiedliche Datenelemente zugreifen oder nur lesend auf die gleichen Datenelemente zugreifen.

2.4.3 Annahmen

Für das Systemmodell werden folgende Annahmen zugrunde gelegt:

Multimaster-Replikation Alle Replikationsteilnehmer können Änderungen an den Daten vornehmen.

Vollständige und partielle Replikation In den in der Einleitung (siehe Abschnitt 2.1.4) beschriebenen Szenarien wird von den einzelnen Replikationsteilnehmern nur ein relativ kleiner Datenbestand verwaltet, der sich auf den Kontext des Szenarios beschränkt. Jeder Nutzer benötigt den vollständigen Datenbestand. So ist es z. B. nicht sinnvoll ein mobiles Wiki nur mit einem Teil der Informationen eines Szenarios zu versorgen. Deshalb betrachten wir in dieser Arbeit in erster Linie vollständige Replikation.

Würde man den Datenbestand partitionieren, so dass die Replikationsteilnehmer zum Teil Daten von anderen Teilnehmer anfordern müssten, wäre die Verfügbarkeit der Daten stark eingeschränkt. Ebenso wäre aus Sicht der Konfliktbildung diese Partitionierung nicht relevant, da sich am eigentlichen Vorgang eines kausal parallelen Zugriffs nichts ändern würde.

Es ist ebenfalls denkbar, dass die Daten so partitioniert sind, dass die Teilnehmer jeweils nur Daten aus ihrer Partition verwenden und sich dadurch Untergruppen bilden. Sind die einzelnen Partitionen jeweils disjunkt zueinander, verhält sich die Bildung von Konflikten ebenfalls analog zur vollständigen Replikation (siehe Kapitel 9 Abschnitt 9.3).

Keine lokale Nebenläufigkeit Wie bereits weiter oben erläutert, besitzt jeder Replikationsteilnehmer eine lokale Datenbank. Die vom Teilnehmer initiierten Transaktionen werden zunächst lokal ausgeführt und anschließend an die anderen Teilnehmer verteilt. Jede Datenbank wird lokal nur von einem Benutzer und einer Applikation verwendet. Wenn entfernte Transaktionen von anderen Benutzern eintreffen, können sich jedoch Überschneidungen ergeben. In den aufgeführten Szenarien (siehe Abschnitt 2.1.4) werden nur kurze und aus wenigen Operationen bestehende Transaktionen ausgeführt. Ebenso ist die Häufigkeit der Transaktionen geringer als in anderen Replikationsszenarien. Die Probleme der Mehrbenutzersynchronisation, wie sie in herkömmlichen Datenbanksystemen bestehen, treten hierbei nicht in diesem Maße auf. Eine verzahnte Ausführung der Transaktionen ist deshalb nicht notwendig. Die Transaktionen werden bei den lokalen Datenbanken der Teilnehmer seriell ausgeführt.

Umgebungsinformationen Die Teilnehmer kennen die Anzahl ihrer Nachbarn. Weitere Umgebungsinformationen, wie z. B. GPS Daten (Ort, Zeit) oder der Abstand zu anderen Teilnehmern stehen den Teilnehmern nicht zur Verfügung.

2.4.4 Gruppenverwaltung

Die Gruppenverwaltung für ein Replikationssystem umfasst hauptsächlich drei Aufgaben: erstens die Verwaltung von Mitgliedschaftsänderungen (Leave, Join); zweitens die Bereitstellung eines Fehlerdetektors, der ausgefallene oder unerreichbare Teilnehmer erkennt (Fail); drittens die Verteilung von Mitgliedschaftsänderungen an die Teilnehmer der Gruppe. Die Aufgaben der Gruppenverwaltung sind in Abbildung 2.5 schematisch dargestellt.

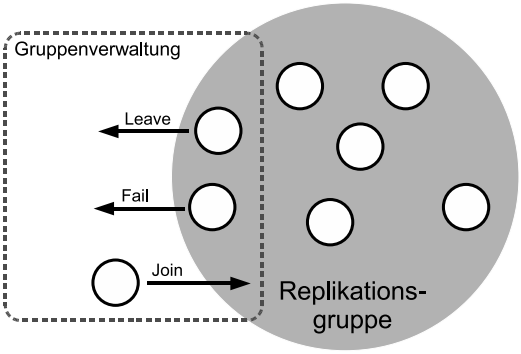


Abbildung 2.5: Gruppenverwaltung

Die im Folgenden beschriebene Gruppenverwaltung entspricht dem in [CDK02] vorgestellten Verfahren, wie es vom ISIS-System verwendet wird [Bir93]. Das Verfahren gewährleistet eine korrekte Ausführung einer Gruppenverwaltung für asynchrone Systeme (siehe ebd.). Die möglichen Probleme, die beim Einsatz der Gruppenverwaltung in MANETs entstehen können, werden im Abschnitt 2.4.5 diskutiert.

Die Gruppenverwaltung basiert auf der Verwendung von **Gruppenansichten**. Dies sind Listen der aktuellen Teilnehmer der Replikationsgruppe, die in unserem System in Form von **Gruppentransaktionen** verteilt werden, die eine gesonderte Tabelle aktualisieren. Aufgrund der Zeitstempel der Transaktionen gibt es für die Gruppenansichten eine totale Ordnung und somit weiß jeder Teilnehmer, welchen Zeitraum eine Gruppenansicht betrifft. Eine Gruppenansicht ist erst gültig, wenn die Mehrheit der Replikationsgruppe diese Ansicht bestätigt hat. Im normalen fehlerlosen Fall liefert dieses Verfahren konsistente Gruppenansichten und ist im Detail in [CDK02] (Abschnitt 14.2.2) und [Bir93] beschrieben. Die Abstimmungen zwischen den Teilnehmern hat einen hohen Nachrichtenaufwand zur Folge. Um diesen Aufwand zu reduzieren, können die Nachrichten der Gruppenverwaltung in die Datenverteilung der Replikation integriert werden.

Die Gruppenverwaltung eines Replikationssystems muss auch den Ausfall oder die Nichterreichbarkeit von Replikationsteilnehmern berücksichtigen. Im Folgenden werden die in der Literatur beschriebenen Lösungen für dieses Problem skizziert.

Die Einigung der Replikationsteilnehmer auf eine Gruppenansicht in einem asynchronen System, in dem Teilnehmer ausfallen können, ist zum Teil problematisch. Dies ist ein Spezialfall des Konsensusproblems (siehe [CDK02] Abschnitt 14.2.2). Fischer et al. haben in [FLP85] bewiesen, dass es keinen Algorithmus geben kann, der Konsens in einem asynchronen System garantiert, falls Teilnehmer ausfallen. Dieses Problem lässt sich jedoch mit Hilfe von Fehler-Detektoren umgehen (siehe [CDK02], Abschnitt 11.5.4). Das ISIS-System verwendet z. B. auf Nachrichten basierende Fehler-Detektoren, um mit Hilfe eines Timeouts zu bestimmen, ob ein Teilnehmer ausgefallen ist [Bir93]. Ob dieser tatsächlich ausgefallen ist oder nicht, spielt keine Rolle. Als „ausgefallen“ erkannte Teilnehmer werden von den anderen Teilnehmern nicht mehr berücksichtigt. Aufgrund dieser Technik wird ein asynchrones System praktisch zu einem synchronen System gemacht und somit ist das Erzielen eines Konsens’ nicht mehr unmöglich.

Ein weiterer Ansatz wird von Chandra und Toueg beschrieben [CT96]. Hierbei werden ebenfalls auf Nachrichten basierende Fehler-Detektoren verwendet, die Teilnehmer aufgrund eines Timeouts als fehlerhaft (ausgefallen) einstufen. Im Gegensatz zum Verfahren des ISIS-Systems ist es hierbei erlaubt, die Nachrichten der Teilnehmer entgegenzunehmen, die vom Fehlerdetektor als ausgefallen eingestuft wurden. Somit können z. B. Transaktionen eines fehlerhaften Teilnehmers berücksichtigt werden, wenn diese vor dem nächsten globalen Commit mindestens einen nicht fehlerhaften Teilnehmer erreichen.

Hierbei muss die Gruppenverwaltung dafür sorgen, dass sich aufgrund von Partitionierung nicht mehrere Teilgruppen bilden können, die jeweils unabhängig ein globales Commit ausführen. Deshalb darf ein globales Commit in nur maximal einer Partition durchgeführt werden. Diese sogenannte **primäre Partition** wird dadurch festgelegt, dass sie die Mehrheit der zuvor existierenden Replikationsgruppe enthält. Durch die Verwendung einer primären Partition wird eine globale Commit Entscheidung nur von einer Replikationsgruppe getroffen und somit ist die Konsistenz der Daten gewährleistet.

2.4.5 Fehlermodell

Aus Sicht der Replikation können Fehlerfälle in drei relevante Klassen unterteilt werden:

- **Verzögerung der Kommunikation** Wie bereits in Abschnitt 2.1 erwähnt, treten Fehler dieser Klasse in MANETs sehr häufig auf. Ob die verzögerte Kommunikation dabei von dem Teilnehmer selbst (z. B. temporärer Ausfall) oder vom Netz verursacht wird, spielt keine Rolle.

- **Ausfall der Kommunikation** Diese Fehlerklasse bezeichnet einen dauerhaften Ausfall der

Kommunikation. Die Ursachen dafür sind die dieselben wie im ersten Fehlerfall (siehe Abschnitt 2.1).

- **Partitionierung** Diese Fehlerklasse bezeichnet eine dauerhafte Partitionierung der Replikationsgruppe, so dass *keine primäre Partition* vorhanden ist. Eine temporäre Partitionierung des Netzes ist für das Replikationssystem bereits durch die erste Fehlerklasse (Verzögerung der Kommunikation) abgedeckt. Eine dauerhafte Partitionierung des Netzes, bei der eine primäre Partition vorhanden ist, wird durch den zweiten Fehlerfall (Ausfall der Kommunikation) abgedeckt.

Die Klasse der **beliebigen Fehler**, häufig auch byzantinische Fehler genannt, umfasst alle Arten von weiteren Fehlern und wird im Rahmen dieser Arbeit nicht thematisiert. Es wird angenommen, dass diese Fehler nicht auftreten.

Die Klasse der **fehlerhaften Uhrensynchronisation** kann an dieser Stelle nicht hinreichend genau diskutiert werden. Aus diesem Grund verweisen wir auf die detaillierte Darstellung dieses Problems in Kapitel 6.

Auswirkung auf die Konflikterkennung und -lösung

Eine *Verzögerung der Kommunikation* stellt für die von uns verwendeten Replikationsmethoden an sich kein Problem dar, da für die Replikation eine asynchrone Verteilung verwendet wird. Trotzdem können lange Verzögerungszeiten die Gefahr von Konflikten zwischen den Transaktionen erhöhen. Die daraus resultierenden Transaktionsabbrüche können praktisch zu einem nicht verwendbaren System führen, auch wenn die Replikation korrekt arbeitet. Dieser Sachverhalt wird in Kapitel 9 genauer analysiert. Er stellt einen zentralen Aspekt der Arbeit dar.

Ein *Ausfall der Kommunikation* wirkt sich auf die Konflikterkennung und -lösung nicht aus. Da der Teilnehmer von der Replikationsgruppe ausgeschlossen wird, wirken sich von ihm ausgeführte Transaktionen nicht mehr auf die Replikationsgruppe aus. Im Gegensatz zur verzögerten Kommunikation wird dadurch die Gefahr von Konflikten nicht erhöht, sondern sogar verringert, weil die Transaktionen des ausgeschlossenen Teilnehmers keine Konflikte mehr verursachen können.

Bei einer *Partitionierung* des Netzes ist für die Bildung von Konflikten nicht relevant, ob eine primäre Partition existiert. Ob eine verzögerte oder ausgefallene Kommunikation durch eine Partitionierung hervorgerufen wird, ist im Hinblick auf die Konfliktbildung ebenfalls nicht relevant.

Auswirkung auf das Commitverfahren und die Gruppenverwaltung

Wie bereits in Abschnitt 2.4.4 gezeigt, ist die Verwendung einer Gruppenverwaltung in asynchronen Systemen grundsätzlich möglich. Es stellt sich jedoch die Frage, ob und wie zeitnah eine Replikationsgruppe in einem MANET Konsens bezüglich der Gruppenansichten erzielen kann. Zunächst einmal ist die Realisierung von möglichst korrekten Fehler-Detektoren mit Hilfe von Timeouts in

MANETs problematisch. Die Unterscheidung zwischen einer stark *verzögerten Kommunikation* und dem *Ausfall der Kommunikation* ist in einem MANET nur schwer möglich. Daraus resultiert bei zu kleinen Timeout-Werten, dass Teilnehmer fälschlicherweise als ausgefallen erkannt werden. Diese müssen dadurch wieder ein Join durchführen, was den Nachrichtenaufwand unnötig erhöht.

Bei der Wahl zu großer Timeout-Werte wird die Aktualisierung der Gruppenzugehörigkeit unnötig verzögert. Da sich der Zustand von mobilen Knoten in MANETs häufig ändern kann, ist es in diesem Fall wahrscheinlich, dass die Aktualisierung der Gruppenzugehörigkeit nicht mehr zeitnah erfolgen kann. Dadurch weichen die von der Gruppenverwaltung angenommenen Teilnehmer von den tatsächlichen Teilnehmern ab und eine globale Commit Entscheidung kann nicht getroffen werden.

Eine *Partitionierung* des Netzes, bei der keine primäre Partition vorhanden ist, blockiert die Gruppenverwaltung und damit auch das Commitverfahren. Ohne eine primäre Partition kann die Gruppenverwaltung keinen Konsens mehr erreichen und ist so lange blockiert, bis wieder eine primäre Partition vorhanden ist.

Abschließend lässt sich festhalten, dass die Verwendung einer Gruppenverwaltung in MANETs von der Gruppendynamik des jeweiligen Szenarios abhängt. Ebenso wie bei der zuvor beschriebenen Konfliktproblematik gibt es Szenarien, in denen das System zwar Eventual Consistency gewährleisten kann, jedoch praktisch unbrauchbar wird. So hat zum Beispiel das in der Einleitung beschriebene Touristenszenario eine hohe Teilnehmerfluktuation und daher ist der Einsatz einer Gruppenverwaltung fraglich. Im Katastrophenszenario bleibt die Gruppenzugehörigkeit hingegen über lange Zeiträume unverändert. Durch die Organisation der Einsatzkräfte ist die Gruppenzugehörigkeit initial bekannt. Ebenso ist die Verbindungsunterbrechung zu einem Teilnehmer über mehrere Stunden als Normalfall anzusehen und führt deshalb nicht zum Ausschluss des Teilnehmers. Die Einsatzgruppe ist fest organisiert und deswegen ist der Ausfall (Fail) eines Teilnehmers nur selten zu erwarten. In einem solchen Fall muss sich jedoch die Mehrheit der Replikationsgruppe abstimmen (siehe Abschnitt 2.4.4). Da sich jedoch die Gruppenteilnehmer in regelmäßigen Abständen gemeinsam treffen, kann eine Abstimmung spätestens zu diesem Zeitpunkt problemlos durchgeführt werden.

Es zeigt sich somit, dass es MANET-Szenarien gibt, in denen der Einsatz einer Gruppenverwaltung möglich ist.

Kapitel 3

Verwandte Replikationssysteme

In diesem Kapitel stellen wir optimistische Replikationsansätze vor, deren Methoden sich für mobile Szenarien eignen bzw. dafür konzipiert wurden. Dies sind meist Replikationssysteme oder Dienste, die mit den im vorigen Kapitel eingeführten Replikationsmethoden arbeiten.

3.1 State-Transfer Replikationssysteme

Wie bereits in Kapitel 2 beschrieben, übertragen State-Transfer Systeme immer nur den aktuellen Zustand der Datenelemente und führen keine Operationshistorie. Sie sind in ihrer Fähigkeit der Konflikterkennung und -lösung beschränkt. In State-Transfer Systemen können z. B. keine kommutativen Operationen zusammengeführt werden. Ebenso ist eine transaktionale Verarbeitung von Datenbankoperationen in einem State-Transfer System nicht ohne Weiteres durchführbar (siehe Abschnitt 2.4.2). Trotzdem verwenden diese Replikationssysteme teilweise ähnliche Replikationsmethoden wie die in dieser Arbeit fokussierten Operation-Transfer Systeme, weshalb sie an dieser Stelle ebenfalls beschrieben werden.

3.1.1 Wingman

Wingman [Ham95] ist ein Multimaster State-Transfer Replikationsdienst für Microsoft Access und Visual Basic. Er stellt die Konsistenz der lokalen Datenbanken der Teilnehmer auf Zeilenebene her und unterstützt keine transaktionale Verwaltung seiner Operationen.

Konflikte werden mit Hilfe von Versionsvektoren (sogenannten lineages) erkannt. Die Auflösung der Konflikte basiert auf der Anzahl der durchgeführten Änderungen auf einer Zeile. Die Zeile mit der geringeren Anzahl an Änderungen wird mit dem Inhalt der anderen Zeile überschrieben. Somit wird vom System die Einhaltung von Eventual Consistency gewährleistet. Durch diese Vorgehensweise kann die Konfliktlösung jedoch leicht beeinflusst werden. Ein Teilnehmer kann z. B. durch zusätzliche Änderungen einer Zeile forcieren, dass im Konfliktfall seine Änderungen übernommen werden. Im Gegensatz zu den meisten anderen Replikationssystemen wird die Konfliktlösung bei Wingman dezentral durchgeführt. Für die Verteilung der Daten verwendet das System ein Synchronisationsprotokoll, das für einen Datenaustausch konzipiert ist, der paarweise durchgeführt wird.

Ebenso werden Änderungen am Schema vom System unterstützt, diese dürfen jedoch nur von einem einzigen Master durchgeführt werden.

Wingman stellt durch seine dezentrale Konfliktlösung einen interessanten Ansatz für dezentral organisierte Netze dar. Der Dienst ist jedoch nicht für mobile Funknetze konzipiert und verwendet keine Broadcastprotokolle, die die Eigenschaften von MANETs berücksichtigen.

3.1.2 Bengal Datenbank Replikationssystem

Bengal [EMRP01] ist ein State-Transfer System, das entworfen wurde, um die Replikation zwischen heterogenen Datenbanksystemen wie z. B. Sybase SQL Anywhere, Oracle und Microsoft SQL Server durchzuführen. Es ist für die Verwendung in mobilen Szenarien vorgesehen und ist wie Wingman dezentral organisiert. Für die Konflikterkennung werden Versionsvektoren verwendet. Die Replikation wird auf Zeilenebene ohne Berücksichtigung transaktionaler Aspekte durchgeführt. Die Datenverteilung wird mit Hilfe eines epidemischen Gossip-Protokolls paarweise durchgeführt, welches manuell gestartet wird.

Die Konfliktlösung kann entweder manuell oder automatisch mit Hilfe eines sogenannten Conflict Resolvers (Konfliktlöser) vorgenommen werden. Dieser ist ein Algorithmus, der mit Hilfe eines Regelsystems definiert werden kann. Leider wird in [EMRP01] nicht dokumentiert, wie diese Conflict Resolvers arbeiten, bzw. wie sie definiert werden. Ebenso ist nicht beschrieben, wie Eventual Consistency durch das Replikationssystem gewährleistet wird. Im Hinblick auf Replikation in MANET-Szenarien stellt die manuelle Datenverteilung von Bengal ein Problem dar.

3.1.3 Kommerzielle Client-Server Replikationssysteme für mobile Geräte

Im Gegensatz zu den Systemen aus dem Forschungsbereich konzentrieren sich die kommerziellen Replikationssysteme in erster Linie auf Szenarien mit einem festen Server und mehreren mobilen Klienten. MANETs werden von kommerziellen Systemen bisher nicht berücksichtigt. Im Bereich der Datenbanken sind Oracle Lite, Microsoft SQL Server CE und IBM DB2 Everyplace bekannte Produkte, die Replikationslösungen für mobile Geräte bieten.

Lotus Notes

Eines der ersten kommerziellen Produkte, das Multimaster Replikation verwendet hat, ist Lotus Notes [KBH+88], eine datenbankgestützte und dokumentenorientierte Gruppen-Software. Notes wurde für Szenarien entwickelt, in denen die Verbindungen zwischen den Teilnehmern zum Teil sehr langsam sind oder zeitweise ausfallen können. Aus diesem Grund verwendet Notes optimistische Replikation.

Das System benutzt für die verwalteten Objekte Versionsnummern, die bei jeder Änderung erhöht werden. Zusätzlich wird ein Zeitstempel gespeichert. Ähnlich wie bei Wingman wird bei un-

terschiedlichen Versionsnummern zweier Replikate das Objekt mit der höheren Versionsnummer verwendet und das andere Objekt verworfen. Im Unterschied zu Wingman verwendet Notes jedoch nur die Versionsnummern und keine Versionsvektoren, weshalb kausal parallele Änderungen vom System nicht erkannt werden.

Microsoft SQL Server 2005 CE

Microsoft SQL Server 2005 CE ist ein State-Transfer Datenbank Replikationssystem, das für mobile Geräte wie PDAs entworfen wurde. Für die mobile Datenbank auf den kleinen Geräten ist häufig nur ein Ausschnitt einer großen Datenbank vorgesehen. Die Serverdatenbank befindet sich wiederum auf einem vollwertigen Microsoft SQL Server. Das System unterstützt Multimaster Replikation, d. h. die Daten können nicht nur auf dem Server, sondern auch auf den Klienten verändert werden. Wenn sich ein Klient mit dem Server synchronisiert, so schickt er zunächst die veränderten Daten. Danach werden mögliche Konflikte auf dem Server erkannt und gelöst. Die endgültigen Werte werden dann zum Klienten zurückgeschickt.

Im Gegensatz zu Wingman und Bengal ist die Granularität der Replikation und somit auch die Konflikterkennung variabel einstellbar. Ebenso kann der Applikationsentwickler zwischen einer Vielzahl von vordefinierten Konfliktlösungsstrategien wählen. Zum Beispiel, dass die jüngste/älteste Änderung oder die Änderung mit höchster Priorität gewinnt. Das System stellt ebenfalls ein Framework zur Implementierung eigener Lösungsstrategien zur Verfügung.

Obwohl das Microsoft System bereits viele Aspekte der mobilen Datenbankreplikation (Funkübertragung, mobile Geräte, einfache Konfliktlösung) berücksichtigt, ist es durch die Verwendung eines zentralen Servers nicht für MANETs geeignet.

Oracle Lite

Oracle hat ein mobiles Datenbankreplikationssystem namens Oracle Lite[1] entwickelt. Dabei wird ähnlich wie bei der Microsoft Lösung ein Teil einer Server Datenbank einem PDA oder Laptop lokal als materialisierte Sicht zur Verfügung gestellt. Der Austausch der Daten wird über ein publish-subscribe Modell umgesetzt, d. h., die mobilen Klienten schreiben sich (engl. subscribe) für eine bestimmte Publikation des Servers ein. Publikationen werden über SQL Statements definiert.

Die Granularität der Replikation ist auf Tabellen und Zeilen beschränkt, was bei der Änderung von einzelnen Elementen häufig zu unnötigen Konflikten führt. Die Konfliktlösung kann manuell oder durch drei vordefinierte Strategien (Server gewinnt, Klient gewinnt, ältester/jüngster Zeitstempel gewinnt) realisiert werden. Es wird ebenfalls eine Schnittstelle zur Verfügung gestellt, um eigene Konfliktlösungsstrategien zu implementieren.

[1] Alle Angaben dieses Abschnittes beziehen sich auf die Version 10g.

Im Hinblick auf Datenbankreplikation für MANETs ist die Verwendung eines Servers sehr problematisch. Zusätzlich werden durch die grobe Replikationsgranularität Konflikte erzeugt, die eigentlich vermeidbar wären.

3.1.4 Replication Proxy Server (RPS)

In [Gol05] wird ein Replikationsdienst beschrieben, der ein bestehendes mobiles Datenbanksystem als Basis verwendet und erweitert. Der Replikationsdienst ermöglicht es den mobilen Applikationen zur Laufzeit auszuwählen, welche Daten vom Server repliziert werden sollen und welche nicht. Diese Möglichkeit ist flexibler und komfortabler als bei den bisherigen kommerziellen Datenbanklösungen. Zusätzlich wird eine einheitliche Schnittstelle für die wichtigsten Konfliktbehandlungsmethoden zur Verfügung gestellt.

Der Replikationsdienst wird mit Hilfe eines sogenannten Replication Proxy Server (RPS) realisiert, der sich als zusätzliche Schicht zwischen den mobilen Klienten und dem Datenbankserver befindet. Die Klienten greifen dadurch nicht mehr direkt auf den Server zu, sondern benutzen eine konsolidierte Datenbank auf dem RPS, die eine Kopie der Datenbank des Servers ist. Dadurch ist die mobile Applikation einerseits unabhängig vom zugrunde liegenden Datenbanksystem. Anderseits ist ein zusätzlicher Replikationsschritt zwischen RPS und dem Datenbankserver erforderlich.

Der RPS stellt insgesamt eine sinnvolle Erweiterung der bisherigen kommerziellen mobilen Datenbanklösungen dar. Wie bei diesen Lösungen ist jedoch der Einsatz eines Servers in einem MANET problematisch. Auch die Einbindung des RPS in eine Peer-to-Peer Architektur ist fraglich. Hierbei müsste jeder Replikationsteilnehmer diesen Replikationsdienst zusätzlich lokal einbinden und dafür ist die Leistung der mobilen Geräte wahrscheinlich zu gering.

3.2 Operation-Transfer Replikationssysteme

Der in dieser Arbeit vorgestellte Replikationsansatz gehört zu den Operation-Transfer Replikationssystemen. Diese übertragen nicht die veränderten Daten, sondern die Operationen, die diese Änderungen durchführen. Dies erlaubt eine differenziertere Konfliktlösung als es bei State-Transfer Systemen der Fall ist. In diesem Abschnitt beschreiben wir Operation-Transfer Replikationssysteme für mobile Szenarien. Dabei betrachten wir die Systeme besonders im Hinblick auf ihre Eignung für MANET-Szenarien.

3.2.1 Gossip

Der Gossip Framework [LLSG92] ist ein Replikationssystem, das eine hohe Datenverfügbarkeit für Applikationen mit schwachen Konsistenzkriterien realisiert. Gossip verwendet Versionsvektoren, um die Änderungsoperationen der einzelnen Teilnehmer kausal zu ordnen. Es unterstützt drei

Arten von Ordnungskriterien (kausal, forciert, direkt), welche unterschiedliche Konsistenzkriterien zu unterschiedlichen Kosten realisieren. „Forciert“ und „direkt“ sind striktere Ordnungskriterien und benötigen einen primären Replikationsmanager, der als zentrale Instanz dafür sorgt, dass diese Ordnungskriterien eingehalten werden. Dieser Ansatz ist für MANET-Szenarien nicht verwendbar. Bei einem Ausfall des Replikationsmanagers besteht zwar die Möglichkeit, einen neuen primären Replikationsmanager zu wählen, dies ist jedoch wiederum mit einem hohen Aufwand verbunden.

Bei der kausalen Ordnung der Änderungsoperationen wird ein primärer Replikationsmanager nicht benötigt, jedoch können sich bei den einzelnen Teilnehmern unterschiedliche Zustände ergeben. Dies entsteht dadurch, dass kausal parallele Operationen nur gemäß ihrer lokalen Einfügereihenfolge geordnet werden (FIFO Ordnung – siehe Abschnitt 2.3.7). Berücksichtigt man die häufigen Unterbrechungen zwischen MANET Knoten und die sich daraus ergebende erhöhte Wahrscheinlichkeit für nebenläufige Operationen, so ist dieser Ansatz ebenfalls nicht für MANETs geeignet.

Die Operationen werden durch sogenannte Gossip Messages verbreitet, die paarweise zwischen den einzelnen Teilnehmern ausgetauscht werden. Gossip Messages sollen möglichst nur die für den Empfänger unbekannten Änderungsoperationen enthalten. Um dies zu realisieren, wird eine Zeitstempeltabelle verwendet, die die Zeitstempel der bisher global festgeschriebenen Werte für jeden einzelnen Replikationsteilnehmer verwaltet. Diese Tabelle wird ebenfalls benutzt, um die Operationshistorie zu kürzen. Gossip verwendet dafür eine konservative Strategie, die die Historie erst kürzt, wenn bekannt ist, dass alle Teilnehmer die entsprechenden Operationen bereits erhalten haben. Die Häufigkeit und das genaue Protokoll der Gossip Messages wird nicht von Gossip vorgegeben, sondern außerhalb des Systems definiert. Zusammenfassend kann man sagen, dass die Konsistenzkriterien von Gossip entweder zentralisiert (forciert, direkt) und damit nicht für MANETs geeignet sind oder keine ausreichenden Konsistenz gewährleisten können (kausal).

3.2.2 Bayou

Bayou [DPS+94, TTP+95] wurde am Xerox Palo Alto Research Center entwickelt und verwendet zum Teil ähnliche Methoden wie Gossip. Es ist ein optimistisches Datenbankreplikationssystem, das eine hohe Datenverfügbarkeit für mobile Teilnehmer gewährleistet, auch wenn diese nicht mit den anderen Teilnehmern verbunden sind. Im Gegensatz zu Gossip verwendet Bayou einen semantischen Ansatz, um einen konsistenten Zustand der Replikationsteilnehmer zu gewährleisten. Dieser Mechanismus erlaubt den Applikationen, die Bayou verwenden, eine genaue Spezifikation der Konflikterkennung und -lösung.

Konkret geschieht dies durch Implementierung einer Schreiboperation von Bayou. Die Schreiboperation besteht aus der eigentlichen Änderungsoperation, einer Konsistenzprüfung und einer Merge Prozedur für die Konfliktlösung. Alle drei Teile werden in einer SQL ähnlichen Sprache definiert. Die Implementierung muss von der Applikation vorgegeben werden, und somit ist die Konflikterkennung und -lösung im Gegensatz zu anderen Systemen nicht transparent. Dieser applikations-

spezifische Ansatz hat jedoch den Vorteil, dass die Konflikterkennung und -lösung für jede einzelne Operation differenziert durchgeführt werden kann.

Der Nachteil ist die Komplexität der Schreiboperationen in Bayou, die von der Applikation vorgegeben werden müssen. Für jede Schreiboperation muss zunächst eine Konsistenzprüfung in Form einer Abfrage durchgeführt werden. Dies soll an einem kurzen Beispiel (siehe [TTP+95]) verdeutlicht werden: Im Falle einer verteilten Kalenderapplikation wird beim Einfügen eines Termins anhand einer Abfrage überprüft, ob sich andere Termine mit dem einzufügenden Termin überschneiden. Ist dies der Fall, so wird die Merge Prozedur ausgeführt, die für alle angegebenen alternativen Termine erneut überprüfen muss, ob diese überschneidungsfrei sind. Im Gegensatz zu einer syntaktischen Konflikterkennung ist diese Vorgehensweise aufwändiger. Das System wird dadurch verstärkt beansprucht und somit sein Durchsatz verringert. Insbesondere für Applikationen mit einer komplexen Datensemantik kann dies einen erheblichen Aufwand darstellen. Überdies erlaubt Bayou keinen manuellen Einfluss auf die Konfliktlösung z. B. durch Abstimmung, wie er von einigen Applikationen benötigt wird.

Um die Operationen zu ordnen, verwendet Bayou eine dynamische Variante der Versionsvektoren. Wenn Operationen kausal parallel ausgeführt wurden, so werden sie gemäß der Teilnehmerkennungen geordnet. Diese Information wird jedoch nicht für die Konfliktlösung verwendet, sie wird in Bayou ausschließlich semantisch durchgeführt.

Bayou setzt keine bestimmte Netztopologie voraus. Die Datenverteilung ist dezentral organisiert und wird von einem sogenannten Anti-Entropy Protokoll durchgeführt, das die Daten epidemisch verteilt. Wie bei den bisherigen Gossip Protokollen der anderen Systeme, werden auch hier die Daten paarweise zwischen den Replikationsteilnehmern synchronisiert. Durch den initialen Austausch der jeweiligen Versionvektoren, wird die Differenz der beiden Teilnehmer bestimmt. Anschließend werden die entsprechenden Operationen von einem Teilnehmer zum anderen übertragen. Das Anti-Entropy Protokoll führt nur eine einseitige Synchronisation durch. Für eine beidseitige Synchronisation muss das Protokoll in beide Richtungen ausgeführt werden.

Im Gegensatz zur Datenverteilung wird die Verwaltung der globalen Commits von einem zentralen Replikationsmanager durchgeführt. Mit Hilfe der Commit Entscheidungen werden Schreiboperationen endgültig festgeschrieben und können ihre Position im Write Log (Operationshistorie) nicht mehr ändern. Für festgeschriebene Operationen wird eine totale Ordnung gewährleistet. Somit kann das Write Log gekürzt werden, um Speicherplatz zu sparen. Die festgeschriebenen Operationen werden vom zentralen Replikationsmanager markiert. Diese Markierungen werden dann an die anderen Teilnehmer verteilt.

Im Hinblick auf MANET-Szenarien weist Bayou hauptsächlich zwei Mängel auf. Erstens ist Bayou nicht vollständig dezentral realisiert, sondern verwendet ein zentral verwaltetes Commitverfahren. Zweitens ist der semantische Ansatz zu komplex für mobile Anwendungen, die aufgrund der beschränkten Leistung mobiler Geräte eher leichtgewichtig sein sollten.

3.2.3 Epidemic algorithms for replicated databases (EARD)

In [HSAA03] wird eine Gruppe von epidemischen Replikationsalgorithmen für replizierte Datenbanken vorgestellt. Obwohl diese Algorithmen kein vollständiges Replikationssystem darstellen, behandeln sie einen Großteil der Aspekte mobiler Datenbankreplikation. Ebenfalls berücksichtigen sie im Gegensatz zu vielen anderen Systemen nicht nur Operationen, die auf einem einzelnen Element ausgeführt werden, sondern auch Transaktionen, die aus mehreren Datenbankoperationen bestehen. Die Transaktionen werden atomar ausgeführt und anschließend ebenfalls atomar verteilt. Durch die Berücksichtigung der Write und Read-Sets der Transaktionen wird für festgeschriebene Transaktionen Serialisierbarkeit gewährleistet.

In [HSAA03] werden verschiedene pessimistische und ein optimistischer Replikationsalgorithmus erläutert. Die pessimistischen Algorithmen garantieren Serialisierbarkeit, während der optimistische Ansatz nur nichtserialisierbare Ausführungen der Transaktionen *erkennt*. Im Hinblick auf mobile Netze stellt der optimistische Algorithmus den vielversprechendsten Ansatz dar. Dabei werden die Transaktionen zunächst lokal ausgeführt und danach epidemisch an die anderen Teilnehmer verteilt. Konflikte werden erkannt indem zunächst die Versionsvektoren der Transaktionen verglichen werden. Sind zwei Transaktionen kausal parallel ausgeführt worden, so werden ihre Read und Write-Sets verglichen. Es wird jeweils die Schnittmenge aus dem Read-Set der einen Transaktion mit dem Write-Set der anderen Transaktion gebildet und es werden die beiden Write-Sets geschnitten. Ergibt sich in mindestens einem der Fälle eine nicht leere Menge, wird dies entsprechend als Schreib-Schreib bzw. Lese-Schreib Konflikt behandelt.

Im Falle eines Konflikts gibt es zwei mögliche Konfliktlösungsstrategien. Entweder werden alle beteiligten Transaktionen abgebrochen, was eine unnötig strikte Lösung darstellt, oder die Konfliktlösung wird der Applikation überlassen. In diesem Falle ist es jedoch nicht mehr möglich, eine Transaktion abzubrechen, da sie bereits festgeschrieben wurde. Dadurch kann keine Serialisierbarkeit mehr gewährleistet werden. Eine Beschreibung der applikationsgesteuerten Konfliktlösung wird in [HSAA03] nicht gegeben.

Eine experimentelle Evaluation vergleicht den optimistischen Ansatz mit den pessimistischen Algorithmen mit Hilfe von Simulationen. Die Simulationen werden im Festnetz mit schnellen Verbindungen durchgeführt, in dem es jedoch teilweise zu kurzen Verbindungsunterbrechungen kommt. Die Ergebnisse zeigen, dass sich der optimistische Ansatz besser eignet als die pessimistischen Algorithmen: Er verursacht keine globalen Verklemmungen und Blockierungsverzögerungen. Weiterhin ist er durch seine asynchrone Kommunikation flexibler und robuster als die pessimistischen Ansätze, die eine synchrone Kommunikation benötigen.

Zusammenfassend kann man sagen, dass in [HSAA03] ein vielversprechender Ansatz zur optimistischen Datenbankreplikation für mobile Netze vorgestellt wird. Trotzdem fehlen folgende Aspekte bzw. werden sie nur am Rande behandelt: Eine Konfliktlösung ist praktisch nicht vorhanden. Außerdem wird die Anbindung an eine Applikation nicht weiter ausgeführt. In der Evaluation

werden nur Festnetze simuliert. Daher haben die Simulationsergebnisse nur eine begrenzte Aussagekraft für MANET-Szenarien.

3.2.4 IceCube

In [PSM03, KRSD01] wird ein Replikationssystem vorgestellt, das den Prozess der Konflikterkennung und -lösung als Optimierungsproblem auffasst. IceCube erkennt, verwaltet und löst bestehende Konflikte, um einen konsistenten Zustand bei allen Replikationsteilnehmern basierend auf der Semantik der einzelnen Operationen herzustellen. Wie bei den Schreiboperationen von Bayou enthalten die Operationen bei IceCube eine Vorbedingung und den Code, der die eigentliche Änderung der Daten durchführt. Im Gegensatz zu Bayou und anderen Systemen werden die in Konflikt stehenden Operationen von IceCube nicht zusammengeführt (merge), sondern neu geordnet, um die Anzahl der Konflikte so gering wie möglich zu halten. Diese Umordnung basiert auf Objekt- und Applikationssemantik, die mit Hilfe von statischen oder dynamischen Constraints (Nebenbedingung) definiert wird. Die Neuordnung der Operationen ist ein NP-schweres Problem. Deshalb versucht IceCube mit Hilfe einer Heuristik eine Näherungslösung in angemessener Laufzeit zu bestimmen.

Im Gegensatz zu anderen Systemen verwendet IceCube keine Versionsvektoren, sondern nutzt die explizite Darstellung der kausalen Abhängigkeiten der Operationen, um Nebenläufigkeit zu erkennen (siehe Abschnitt 2.3.5). Die Verteilung der Operationen wird manuell durchgeführt und ist nicht auf eine bestimmte Netzwerktopologie ausgerichtet. Als Proof-of-Concept wurden zwei Beispielapplikationen, ein verteiltes Puzzle und ein verteilter Terminkalender mit Hilfe von IceCube realisiert. Die dadurch gewonnenen Erfahrungen zeigen, dass starke statische Constraints (z. B. Ordnungskriterien) nötig sind, um die Komplexität der Neuordnung zu begrenzen.

IceCube wurde für mobile Szenarien wie z. B. die offline Verwendung von Laptops, entworfen. Es ist jedoch fragwürdig, inwiefern sich dieses System für die Verwendung in MANETs eignet. Für kleine mobile Geräte wie PDAs oder Smartphones scheint der Optimierungsansatz der Operationshistorie zu komplex. Weiterhin wird die Datenverteilung nur manuell durchgeführt, was in MANET-Szenarien eine unnötige Verzögerung zur Folge hätte.

3.2.5 Joyce

Joyce [OS05] ist ein Framework, das operationsbasierte Replikation unterstützt. Es basiert auf dem zuvor besprochenen IceCube und versucht somit ebenso eine optimale Anordnung der Operationen zu finden. Dabei berücksichtigt Joyce sowohl syntaktische als auch semantische Constraints (Log und Object Constraints). Die Konflikterkennung und -lösung wird in Gruppen unterteilt, die von einem sogenannten Multi-Log verwaltet werden. Die Datenverteilung ähnelt Systemen wie Gossip oder Bayou und wird epidemisch und paarweise durchgeführt. Nachdem zwei Teilnehmer ihre Versionsvektoren verglichen haben, tauschen sie alle noch nicht bekannten Operationen miteinander aus.

Um eine globale Konsistenz zu gewährleisten, verwendet Joyce einen vorher festgelegten Teilnehmer als zentrale Instanz (Primary). Es gibt pro Gruppe nur einen Primary, was wie bereits erwähnt für MANET-Szenarien problematisch ist. Es wird außerdem angenommen, dass nur ein Teil der Teilnehmer die Daten lokal speichert. Auch diese Einschränkung ist für Replikation in MANETs eher problematisch, da die Teilnehmer oft nicht mit dem Netz verbunden sind und deshalb die Daten lokal benötigen.

Um Joyce zu bewerten, wurde der Texteditor Babble für verteilte Texte implementiert. Die gewonnenen Erkenntnisse haben gezeigt, dass Joyce insgesamt die Entwicklung der auf verteilten Daten arbeitenden Applikationen vereinfacht. Die Anpassung der Applikationslogik an das Constraint Modell von Joyce ist jedoch kompliziert und für Applikationsentwickler ungewohnt.

Wie bei IceCube ist bei Joyce fraglich, ob sich dieser Ansatz aufgrund seiner Komplexität für den Einsatz in MANETs eignet. Ebenso ist die Verwendung eines Teilnehmers als zentrale Instanz für MANETs ungeeignet.

3.2.6 LogDB

In [Hup09] wird ein Datenbankreplikationssystem vorgestellt, das ebenfalls die kausalen Abhängigkeiten der Operationen verwendet, um kausal parallele Zugriffe festzustellen. Die kausal parallelen Versionen der Daten werden ähnlich wie bei Versionsverwaltungssystemen wie CVS oder SVN als Verzweigungen (engl. branches) behandelt. Das System bietet sowohl die Möglichkeit der Konflikterkennung als auch der Konfliktlösung, indem eine der kausal parallelen Versionen global konsistent ausgewählt wird. Die anderen Versionen werden verworfen. Dieser Vorgang wird als *Distributed Consistent Cutting* bezeichnet.

Für die Verteilung der Operationen wird ein Multicast Protokoll und ein Synchronisationsprotokoll vorgestellt. Beide Protokolle regeln den Datenaustausch auf Applikationsebene und berücksichtigen nicht die Eigenschaften von Funknetzen. Die Evaluation des Systems wird mit Hilfe von simulierten Szenarien im Festnetz durchgeführt. In den Szenarien werden in erster Linie kleine Replikationsgruppen von bis zu sieben Teilnehmern in Festnetzen betrachtet.

Die mögliche Verwendung des Systems für mobile Szenarien wird in [Hup09] zwar angesprochen, jedoch nicht weiter betrachtet. Die Evaluation des Systems berücksichtigt keine Mobilität der Teilnehmer. Ebenso werden die Eigenschaften von Funknetzen in den Untersuchungen nicht berücksichtigt. Im Gegensatz zur vorliegenden Arbeit werden weder Transaktionen noch Ansätze zur Reduzierung der Konfliktanzahl betrachtet. Die Auswirkung der Netztopologie auf die Konfliktanzahl wird in der Evaluation nicht untersucht. Dieser Aspekt ist gerade in mobilen Szenarien sehr entscheidend, weil sich die Netztopologie in MANETs verhältnismäßig stark auf die Datenverteilungsdauer und damit auf die Konfliktanzahl auswirkt.

3.2.7 Tabellarische Übersicht

Im Folgenden wird in den Tabellen 3.1 und 3.2 eine Übersicht der relevanten Replikationssysteme gegeben. Die Informationen dazu wurden den bereits genannten Quellen entnommen.

	Bayou	**IceCube/Joyce**	**LogDB**
Szenario			
Anzahl der Teilnehmer	Arbeitsgruppe (~3-50 Teiln.)	Kleine Gruppen (< 30 Teiln.)	3-7 Teiln. (simuliert)
Geräte	Beliebig (Implementiert für Desktop PCs)	Implementiert für Desktop PCs und Laptops	Beliebig (Implementiert für Desktop PCs)
State-/Operation-Transfer	Operation-Transfer	Operation-Transfer	Operation-Transfer
Kommunikationsmodell	Client-Server (Commitverfahren), Peer-to-Peer (Datenverteilung)	Client-Server (Commitverfahren), Peer-to-Peer (Datenverteilung)	Peer-to-Peer
Schreibende Teilnehmer	Single-master	Single-master (Primärknoten)	Multimaster
Datenverteilung	Epidemisch (Anti-Entropy Protokoll)	Manuell	Direktes Multicastprotokoll und Synchronisationsprotokoll
Netz	Unabhängig von der Netztopologie	Keine Angaben	Mobiles Funknetz, Festnetz (simuliert)
Anwendungen	Zugrunde liegendes proprietäres DBS, verteilter Terminkalender	Verteilter Terminkalender und Editor	Keine Angaben
Replikationssystem			
Commitverfahren	Zentrales Commitverfahren	Keine Angaben	Dezentrales Commitverfahren
Operationen	Proprietäre Operationen auf SQL Anweisungen basierend	Update Operationen mit Constraints, atomare Ausführung gruppierter Operationen möglich	Einfache Datenbankoperationen (read/write)
Konflikterkennung	Vorbedingungen werden überprüft, um Konflikte zu erkennen, nebenläufige Ausführung wird nicht berücksichtigt	Semantisch und syntaktisch (Vorgängerliste) basierend auf den gegebenen Constraints	Syntaktisch (Vorgängerliste)
Konfliktlösung	Merge, das durch eine in der Operation definierte Funktion ausgeführt wird, wenn die Vorbedingung nicht erfüllt ist	Optimierung der Reihenfolge, um die Konfliktanzahl zu verringern; basierend auf einer semantischen Graphstruktur	Festlegung einer gültigen Version; Abbruch aller Operationen, die nicht zu dieser Version gehören
Konsistenz	Eventual Consistency	Eventual Consistency, keine Ordnungsgarantie	Eventual Consistency mit totaler und kausaler Ordnung

Tabelle 3.1: Überblick über die Eigenschaften der Replikationssysteme Bayou, IceCube/Joyce und LogDB

	Epidemic Algorithms for Replicated Databases	**Wingman**	**Gossip**
Szenario			
Anzahl der Teilnehmer	5 - 25 Teiln. (simuliert)	Kleine Gruppen (< 30 Teiln.)	Arbeitsgruppe (~3-50 Teiln.)
Geräte	Desktop PCs, mobile Geräte sind angedacht	Desktop PCs and Laptops	Keine Angaben
State-/Operation-Transfer	Operation-Transfer	State-Transfer	Operation-Transfer
Kommunikationsmodell	Peer-to-Peer	Peer-to-Peer	Client-server (Commitverfahren), Peer-to-Peer (Datenverteilung)
Schreibende Teilnehmer	Multimaster	Multimaster	Single-master (primärer Replikationsmanager)
Datenverteilung	Epidemisch	Manuell	Epidemisch (Paarweise)
Netz	Internet, LAN (simuliert), mobile Szenarien werden nur angedacht	Internet, LAN	Keine Angaben
Anwendungen	Datenbank + E-Commerce Anwendungen	MS Access, Visual Basic	Datenbank, Forum, verteilter Terminkalender
Replikationssystem			
Commitverfahren	Dezentrales atomares Commitverfahren	Keine Angaben	Forcierte und sofortige Update Operationen (primärer Replikationsmanager)
Operationen	Datenbank Transaktionen	Zeilen Updates	Anfrage und Update Operationen (nur blind Writes)
Konflikterkennung	Syntaktisch (Versionsvektoren), w/w und r/w Konflikte werden erkannt, Protokoll ist zu strikt: ein Teil der erkannten r/w Konflikte stellt keinen tatsächlichen Konflikt dar	Syntaktisch (Versionsvektoren)	Keine Konflikterkennung
Konfliktlösung	Abbruch aller Transaktionen oder Quorumsentscheidung	Paarweise, Teiln. mit den meisten Update Operationen gewinnt	Operationen werden entsprechend ihrer lokalen Ankunftszeit sortiert
Konsistenz	Eventual Consistency, Serialisierbarkeit	Eventual Consistency	Mindestens schwache Konsistenz, Ordnungsmodi: Kausal - Kausalordnung (Lamport Clocks) Forciert, sofort - Totale kausale Ordnung mit Hilfe von Versionsvektoren

Tabelle 3.2: Überblick über die Eigenschaften der Replikationssysteme EARD, Gossip und Wingman

3.3 Schlussfolgerung

Viele der beschriebenen Systeme sind für mobile Szenarien konzipiert worden. Damit ist in den meisten Fällen die Offline-Benutzung eines Laptops oder PDAs gemeint, der sich nach gewisser Zeit wieder mit dem Netz verbindet und dann synchronisiert. Während der Klient nicht mit dem Netz verbunden ist, soll es ihm weiterhin möglich sein, Daten zu lesen und zu verändern. Obwohl dieses Szenario einige Ähnlichkeiten mit MANET-Szenarien aufweist, sind die zuvor beschriebenen Ansätze nur in Teilen für MANETs verwendbar.

So wird z. B. die Verteilung von vielen Systemen manuell angestoßen, was bei den spontanen Verbindungsänderungen in MANETs nicht sinnvoll ist. Ebenso wird die Konflikterkennung von den meisten Systemen bis auf Wingman, Bengal und EARD zentral durchgeführt. Auch dieser Ansatz ist für MANETs wenig praktikabel, da ein mobiler Server häufig ausfallen kann oder nicht mit den anderen Teilnehmern verbunden ist.

Betrachtet man die Replikationsmethoden der Systeme im Hinblick auf die Verwendung für kleine mobile Geräte wie PDAs oder Smartphones, so stellt man fest, dass die Ansätze oft zu aufwändig bzw. zu komplex sind. Insbesondere semantische Replikationsansätze sind dabei eher ungeeignet. Bisher kann man nur bei den kommerziellen Lösungen eine konkrete Ausrichtung auf PDAs und Smartphones feststellen. Leider mangelt es diesen State-Transfer basierten Systemen an einer differenzierten Konflikterkennung und -lösung, wie sie bei Operation-Transfer Systemen gegeben ist.

Für die Konzeption eines mobilen Datenbankreplikationssystems für MANETs lassen sich daher folgende Forderungen ableiten:

- Die Datenverteilung der Replikation sollte speziell an die Gegebenheiten von MANETs angepasst sein.
- Die Verwaltung der Replikation sollte dezentral realisiert werden.
- Die Konflikterkennungs- und Konfliktlösungsmethoden sollten die eingeschränkte Leistung mobiler Geräte berücksichtigen.

In der vorliegenden Arbeit stellen wir neue Replikationsmethoden für MANETs vor, welche die oben genannten Forderungen erfüllen. In den nachfolgenden Kapiteln werden sowohl die Methoden im Einzelnen, als auch das Replikationssystem SYMORE vorgestellt, das diese Methoden integriert. Anschließend wird dieses System experimentell und analytisch evaluiert.

Teil II

Methoden und Verfahren

Kapitel 4

Das Area Graph-based Bewegungsmodell

Im Rahmen dieser Arbeit wird die Replikation in MANETs betrachtet. Wie bereits zuvor erwähnt, gibt es bisher keine größeren real existierenden MANETs bzw. sind diese nicht frei verfügbar (z. B. Militärnetze). Ein künstlicher Aufbau derartiger Netze wäre ebenfalls zu aufwändig. Aus diesem Grund werden häufig Simulationen und Emulationen verwendet, um die Eigenschaften von MANETs zu untersuchen. Bei diesen Ansätzen werden Bewegungsmodelle verwendet, die die Bewegung der mobilen Knoten modellieren.

Die Topologie und Dynamik eines Netzes wird bei Simulationen hauptsächlich vom verwendeten Bewegungsmodell bestimmt. Dieses wirkt sich auf die Datenverteilung und somit auch auf die Replikation innerhalb des Netzes aus. Aus diesem Grund ist es im Rahmen dieser Arbeit notwendig, Bewegungsmodelle zu betrachten. Dies ist ein Unterschied zu Netzen mit fester Infrastruktur, in denen nicht die Mobilität *aller* Netzteilnehmer berücksichtigt werden muss.

Es gibt heutzutage bereits eine Vielzahl von Bewegungsmodellen, welche für die verschiedenen Anforderungen einzelner Untersuchungen entwickelt wurden. Viele dieser Modelle abstrahieren stark von den tatsächlichen Gegebenheiten und sind somit nicht sehr realistisch.

Um wichtige Eigenschaften der von uns betrachteten Szenarien (siehe Abschnitt 2.1.4) zu modellieren, haben wir ein neues Modell, das *Area Graph-based Bewegungsmodell*, entwickelt [BRS05]. Das Modell modelliert sowohl die makroskopische als auch die mikroskopische Bewegung der Knoten. Es ist dadurch realistischer als Bewegungsmodelle, die sich nur auf die ausschließliche Modellierung der Makro- oder der Mikrobewegung beschränken.

4.1 Verwandte Bewegungsmodelle

Die erste Verwendung von Bewegungsmodellen stammt aus der Physik, wo die Bewegung von Partikeln modelliert wurde [Ein05]. Bewegungsmodelle werden auch häufig in der MANET Forschung verwendet, wo sie die Bewegung der mobilen Knoten (einzelne Fußgänger oder Gruppen, Autos, etc. mit mobilen Geräten) in unterschiedlichen Umgebungen (Städte, Messehallen, Katastrophengebiet, etc.) modellieren. Allen Bewegungsmodellen ist gemeinsam, dass sie von den realen Bewegungen der Knoten abstrahieren und die Bewegungen mit Hilfe eines analytischen Modells nachbilden. Eine

Übersicht über verschiedene Bewegungsmodelle ist in [AGPM08] gegeben.

4.1.1 Einfache Bewegungsmodelle

Einfache Bewegungsmodelle reduzieren die untersuchten Szenarien auf ihre grundlegenden Eigenschaften, wie zum Beispiel die Anzahl und die Geschwindigkeit der mobilen Knoten oder die Grundfläche des Szenarios. Dies führt dazu, dass unterschiedliche Szenarien mit denselben Grundeigenschaften durch dasselbe Bewegungsmodell repräsentiert werden. Durch diese Verallgemeinerung können Untersuchungsergebnisse einerseits einfacher verglichen werden. Andererseits ist die Aussagekraft der Untersuchungsergebnisse beschränkt, da viele Eigenschaften der Szenarien vernachlässigt werden.

Bekannte Modelle dieser Art sind das *Random Walk Bewegungsmodell* [Spi01], das *Random Waypoint Bewegungsmodell* [BHPC04] und das *Random Direction Bewegungsmodell* [CBD02].

Den drei Modellen liegt jeweils eine rechteckige Grundfläche zugrunde, auf der sich die Knoten schrittweise von einem Wegpunkt zum nächsten bewegen. Alle drei Modelle sind gedächtnislos, d. h. die jeweils nächste Bewegung eines mobilen Knotens hängt nicht von der vorherigen ab. Exemplarisch wird das Random Waypoint Modell im Folgenden erklärt.

Random Waypoint Bewegungsmodell Ein heutzutage häufig genutztes Bewegungsmodell ist das *Random Waypoint Bewegungsmodell*. Ähnlich zum Random Walk Modell bewegen sich die Knoten von einem Standort zu einem neuen Wegpunkt (engl. waypoint). Der neue Wegpunkt wird jeweils zufällig und gleichverteilt auf der rechteckigen Ebene ausgewählt. Die Geschwindigkeit des Knotens wird zufällig und gleichverteilt aus einem vorgegebenen Intervall ermittelt. Bei der Ankunft des Knotens wird eine Aufenthaltsdauer zufällig und gleichverteilt aus einem gegebenen Intervall bestimmt. Der Knoten wartet an dem Standort für die gewählte Zeit und bewegt sich danach entsprechend dem beschriebenen Verfahren weiter.

Dieses Modell soll das Verhalten eines bewusst steuernden mobilen Knotens, z. B. eines Fußgängers oder eines Autos, repräsentieren. Ein Beispiel dafür ist die Bewegung eines Messebesuchers: Der Besucher bewegt sich von Messestand zu Messestand und verbringt an jedem Standort eine gewisse Zeit. In Abbildung 4.1 wird ein Beispiel für ein Bewegungsmuster des Random Waypoint Modells gezeigt.

Die Verteilung der Knoten auf der Fläche entspricht nicht einer Gleichverteilung. Bei der Initialisierung des Modells werden zwar alle Knoten zufällig und gleichverteilt auf der Grundfläche positioniert, diese initiale Verteilung entspricht jedoch nicht der Verteilung, die sich durch die Bewegung der Knoten im weiteren Verlauf ergibt. Der Grund dafür ist, dass die Knoten sich durch eine gleichverteilte Wahl eines neuen Wegpunktes auf der Fläche häufiger über die mittlere Region bewegen als über die Randgebiete. Dadurch steigt die Knotendichte zum Mittelpunkt der Grundfläche des Modells [BW02].

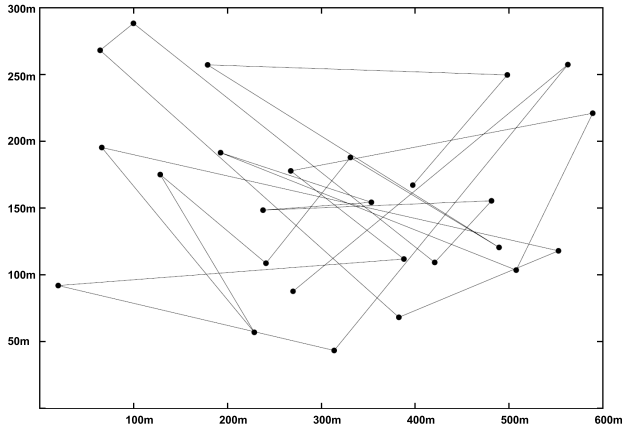


Abbildung 4.1: Bewegungsmuster eines Random Waypoint Bewegungsmodells [CBD02]

4.1.2 Komplexere Bewegungsmodelle

Komplexere Bewegungsmodelle berücksichtigen im Gegensatz zu den einfachen Modellen mehr Aspekte der realen Welt (Wege, Strassen, Hindernisse) und geben deshalb die Bewegung der mobilen Knoten innerhalb der Grundfläche genauer vor.

Bekannte Modelle dieser Art sind das *City-Section Bewegungsmodell* [Dav00], das *Obstacle Bewegungsmodell* [JBRAS03] und das *Graph-based Bewegungsmodell* [THB+02]. Während das City-Section Modell die Bewegung der Knoten auf ein rechtwinkliges Gitternetz abbildet, werden beim Obstacle Modell die Bewegungen der Knoten durch die explizite Modellierung von Hindernissen beeinflusst. Das Graph-based Modell ermöglicht eine sehr flexible Modellierung der Szenarien mit Hilfe eines Graphen. Da das Area Graph-based Modell eine Erweiterung des Graph-based Modells darstellt, wird letzteres im Folgenden ausführlicher erklärt.

Das Graph-based Bewegungsmodell Das *Graph-based Bewegungsmodell* wird von Tian et al. in [THB+02] vorgestellt. Es basiert auf einem bidirektionalen und zusammenhängenden Graphen, der die möglichen Wege der mobilen Knoten[1] vorgibt.

Die Knoten des Graphen stellen Orte dar, die von den mobilen Knoten besucht werden können. Das können z. B. Einkaufszentren, Sehenswürdigkeiten oder andere öffentliche Orte sein. Die Kanten repräsentieren Straßen, Zugverbindungen oder einfache Gehwege zwischen den Orten. Ein typisches Szenario, wie es auch im Artikel verwendet wird, ist ein Graph, der ein Stadtzentrum repräsentiert.

Bei der Initialisierung des Modells werden alle mobilen Knoten zufällig auf die Knoten des Gra-

[1] Um Verwechslungen zu vermeiden, wird in diesem Abschnitt jeweils von *mobilen Knoten* oder von den *Knoten des Graphen* gesprochen.

phen verteilt. Die mobilen Knoten wählen sich nun einen neuen Graphknoten als Ziel und verwenden dann den kürzesten Weg, um zu diesem Ziel zu gelangen. Es wird nicht erwähnt, ob die Kanten auf irgendeine Art gewichtet sind und inwiefern dies bei der Berechnung des kürzesten Weges berücksichtigt wird. Die Geschwindigkeit der mobilen Knoten wird zufällig aus einem gegebenen Intervall bestimmt. Obwohl in dem Beitrag erwähnt wird, dass die Kanten unterschiedliche Verbindungen (Straße, Zugverbindung, etc.) darstellen können, wird nicht gezeigt, wie dies im Modell umgesetzt wird. Wenn ein Knoten sein Ziel erreicht hat, so wartet er eine gewisse Zeit und wählt sich danach ein neues Ziel.

MANET Simulationen in [THB+02] zeigen, dass das Graph-based Bewegungsmodell die Ergebnisse der verwendeten Routingprotokolle stark beeinflusst. Als Vergleichsmodell wurde das Random Walk Modell verwendet. Im Vergleich zum City Section und Obstacle Bewegungsmodell ist der Ansatz vom Graph-based Modell allgemeiner und somit kann eine größere Bandbreite an Szenarien dargestellt werden. Die Annahme, dass alle Orte eines Szenarios keine räumliche Ausdehnung haben, hat zur Folge, dass sich alle mobilen Knoten an diesem Ort in direkter Funkreichweite befinden. Wie sich diese Einschränkung auf die Datenverteilung auswirkt, wird am Ende dieses Kapitels diskutiert (Abschnitt 4.4).

4.2 Das Area Graph-based Bewegungsmodell

Wie im vorangegangenen Abschnitt beschrieben, ist das Graph-based zwar ein sehr flexibles Modell, abstrahiert jedoch von der Topologie der Graphknoten und stellt sie nur als Punkte dar. Aufgrund dieses Mangels haben wir das *Area Graph-based Bewegungsmodell* (AGM) entwickelt. Das AGM besteht aus einem Makromodell und mehreren Mikromodellen. Das **Makromodell** stellt die übergeordnete Topologie eines Szenarios mit Hilfe eines Graphen dar, wie es auch beim Graph-based Modell der Fall ist. Zusätzlich werden jedoch die Knoten des Graphen als Flächen modelliert, die eine eigene Topologie enthalten, welche durch ein eigenständiges Bewegungsmodell nachgebildet wird. Diese Modelle werden als **Mikromodelle** bezeichnet. Dies erlaubt eine detailliertere und somit auch realistischere Modellierung als die zuvor genannten Bewegungsmodelle. Darüberhinaus ist ebenfalls eine rekursive Definition des Area Graph-based Bewegungsmodells möglich, so dass verschiedene Detailstufen modelliert werden können. Im folgenden Abschnitt werden wir das Area Graph-based Bewegungsmodell definieren und seine Hauptmerkmale diskutieren.

Definition

Das *Area Graph-based Bewegungsmodell* besteht aus den folgenden Teilkomponenten:

Area Graph Das Makromodell des AGM basiert auf einem *Area Graph*. Der Area Graph ist ein in die Ebene eingebetteter gerichteter Graph $G = (V, E)$. Die Knoten (engl. vertices) des Graphen $v \in V$ werden als *Cluster* bezeichnet. Die Kanten (engl. edges) des Area Graphen

$e \in E$ sind *gerichtete Kanten*. Die Kanten sind mit den Clustern über *Verbindungspunkte* $c \in C$ verbunden, die sich auf dem Rand der Cluster befinden. Ein einfaches Beispiel für einen Area Graphen ist in Abbildung 4.2 gegeben.

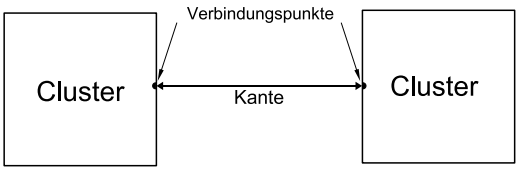


Abbildung 4.2: Beispiel eines Area Graphen mit zwei Clustern

Cluster Ein *Cluster* $v \in V$ wird durch eine rechteckige Fläche C_v begrenzt. Auf dem Rand dieser Fläche befinden sich die Verbindungspunkte des Clusters $c_i^v \in C_v$, an denen die Kanten die Clusterfläche berühren. Die Verbindungspunkte stellen die Ein- und Ausgänge eines Clusters dar. Jedes Cluster v besitzt ein *Zeitintervall* WT_v, aus dem zufällig und gleichverteilt die Aufenthaltsdauer (engl. waiting time) eines mobilen Knotens in diesem Cluster bestimmt wird. Jedes Cluster enthält ebenfalls ein *Mikromodell*, das die Bewegung der mobilen Knoten innerhalb des Clusters definiert. Das Mikromodell kann sowohl ein einfaches Modell (Random Walk, Random Waypoint), als auch ein Area Graph-based Modell sein, das die Bewegung der mobilen Knoten innerhalb des Clusters detaillierter modelliert.

Kante Eine gerichtete *Kante* e ist wie folgt definiert: $e = (c_i^v, c_j^{v'}, w, P, S)$. Die Verbindungspunkte c_i^v und $c_j^{v'}$ definieren den Start- und Zielpunkt der Kante. Das Gewicht w bestimmt die Wahrscheinlichkeit, mit der die Kante von den mobilen Knoten des Startclusters als nächstes Ziel gewählt wird. P ist eine geordnete Menge von Punkten in der Ebene und definiert die Teilstrecken einer Kante. Dadurch wird die geometrische Form der Kante bestimmt. Ist die Menge der Zwischenpunkte leer, so ist die Kante die Strecke zwischen dem Start- und Zielpunkt. Sind für eine Kante ein oder mehrere Zwischenpunkte definiert, so ergibt sich ein Pfad, der aus mehreren Teilstrecken besteht. Zusätzlich besitzt jede gerichtete Kante ein Geschwindigkeitsintervall S, aus dem zufällig und gleichverteilt die Geschwindigkeit gewählt wird, mit der sich die mobilen Knoten auf der Kante fortbewegen.

Sei E_v die Menge aller ausgehenden Kanten eines Clusters v, so wird die Wahrscheinlichkeit, dass ein Knoten eine bestimmte Kante wählt, wie folgt berechnet:

$$p(e = (c_i^v, c_j^{v'}, w, P, S)) = \frac{w}{\sum\limits_{e^* \in E_v} w^*} \tag{4.1}$$

Bei einem Cluster mit drei ausgehenden Kanten mit je einem Gewicht von $w = 1$ würde die Wahrscheinlichkeit für die Wahl einer bestimmten Kante $p(e) = \frac{1}{3}$ betragen.

Ablauf

Zunächst wird das Modell *initialisiert*. Dazu werden alle mobilen Knoten $m \in M$ des Modells auf die einzelnen Cluster verteilt. Die Aufteilung der mobilen Knoten auf die Cluster richtet sich nach deren durchschnittlicher Aufenthaltsdauer in einem Cluster. Nach der Initialisierung bewegen sich die Knoten entsprechend einem definierten Ablauf. In Abbildung 4.3 ist ein Ablaufdiagramm für einen Knoten dargestellt.

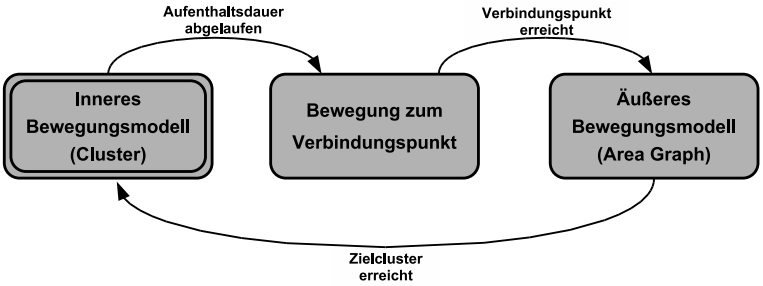


Abbildung 4.3: Ablaufdiagramm eines mobilen Knotens im Area Graph-based Bewegungsmodell

Die Bewegung eines mobilen Knotens wird durch sein aktuelles Bewegungsmodell festgelegt. Befindet sich der Knoten innerhalb eines Clusters, so ist das innere Bewegungsmodell für die Bewegung des Knotens zuständig. Befindet sich der Knoten außerhalb eines Clusters, so wird seine Bewegung durch das übergeordnete Area Graph-based Modell festgelegt. Wenn die Aufenthaltsdauer innerhalb eines Clusters abgelaufen ist, unterbricht der mobile Knoten den Ablauf des inneren Bewegungsmodells und wählt entsprechend der Gewichtung der ausgehenden Kanten einen Verbindungspunkt (siehe Gleichung 4.1). Danach begibt sich der mobile Knoten auf dem kürzesten Weg zum gewählten Verbindungspunkt. In einfachen inneren Modellen (die mobilen Knoten können sich frei auf der Grundfläche bewegen) ist dies der direkte Weg. Bei komplexeren inneren Modellen muss ein Algorithmus angegeben werden, der den kürzesten Weg für dieses Modell bestimmt. Wenn ein Area Graph-based Modell in einem Cluster verwendet wird, so wird der kürzeste Weg durch Dijkstras Algorithmus bestimmt.

4.3 Modellbeispiele

In Abbildung 4.4 ist ein einfaches Beispiel für ein AGM dargestellt. Das Modell besteht aus zwei Clustern mit einer jeweiligen Aufenthaltsdauer von 0s - 60s. Die Geschwindigkeit der mobilen Knoten auf den beiden Kanten (von A nach B und umgekehrt) ist fest und beträgt $1\frac{\mathrm{m}}{\mathrm{s}}$, was der Geschwindigkeit eines Fußgängers entspricht. Das Gewicht der Kanten ist zu vernachlässigen, da die mobilen Knoten in jedem Cluster nur einen Ausgang zur Wahl haben. In den Clustern wird die Bewegung der

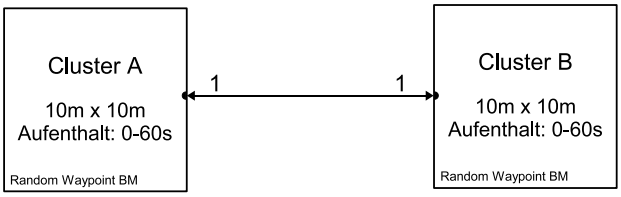


Abbildung 4.4: Beispiel eines Area Graph-based Bewegungsmodells mit zwei Clustern

Knoten durch das Random Waypoint Bewegungsmodell bestimmt, das ebenfalls eine Geschwindigkeit von $1\frac{\mathrm{m}}{\mathrm{s}}$ verwendet und eine Wartezeit von 0s - 10s für die Wegpunkte vorsieht.

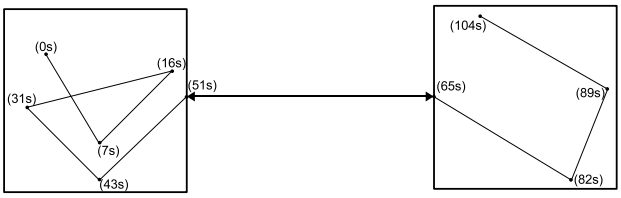


Abbildung 4.5: Bewegungsmuster eines mobilen Knotens entsprechend dem Bewegungsmodell aus Abbildung 4.4

In Abbildung 4.5 wird das zufällige Bewegungsmuster eines Knotens in diesem Modell gezeigt. Zunächst befindet sich der Knoten im linken Cluster, für das eine Aufenthaltsdauer von 45s bestimmt wurde. Zu Beginn (0s) bewegt sich der Knoten zu vier anderen Wegpunkten innerhalb von Cluster A, wobei er den letzten Punkt nach 43s erreicht. Während er an diesem Punkt wartet, ist seine Aufenthaltsdauer abgelaufen und somit bewegt er sich auf den Verbindungspunkt von Cluster A zu und erreicht diesen nach 51s. Nach insgesamt 65s erreicht der Knoten Cluster B und bewegt sich dort wiederum gemäß dem Random Waypoint Bewegungsmodell.

Rekursive Modelle

Mit Hilfe des Area Graph-based Bewegungsmodells ist es durch Verschachtelung möglich, verschiedene Detailstufen zu definieren. Wie zuvor bereits erwähnt, besteht das AGM aus einem Makromodell und mehreren Mikromodellen. Die Mikromodelle können entweder einfache Modelle oder wiederum ein AGM sein. Diese Möglichkeit erlaubt die Definition von komplexen Modellen mit verschiedenen Detailstufen.

Ein Beispiel für die rekursive Definition eines Campusgeländes, wie es für das in Abschnitt 2.1.4 geschilderte Campusszenario denkbar ist, wird in Abbildung 4.6 gezeigt.

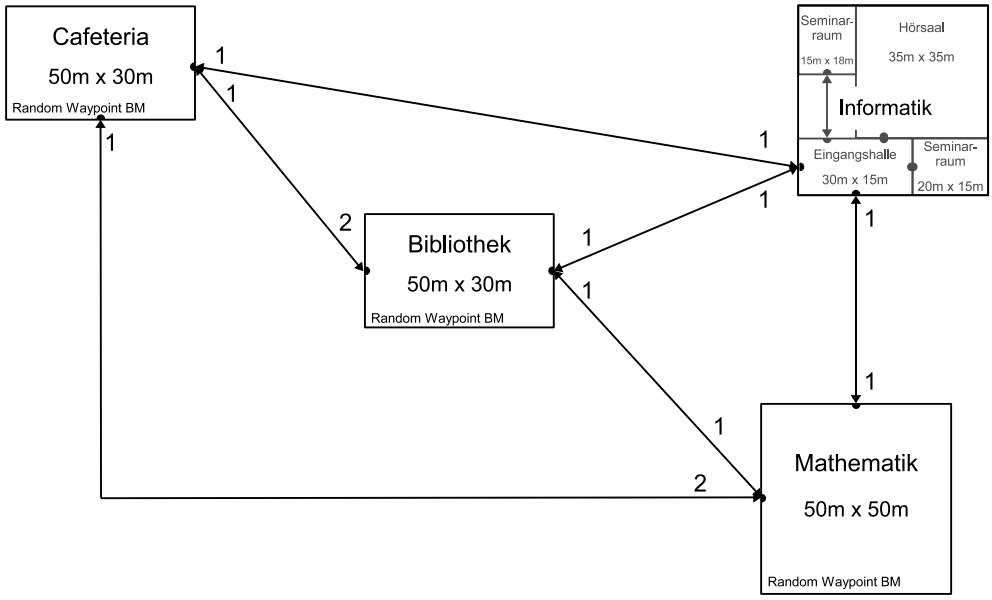


Abbildung 4.6: Area Graph-based Modell eines Campusgeländes. Das Cluster des Informatikgebäudes ist ebenfalls mit dem AGM modelliert worden.

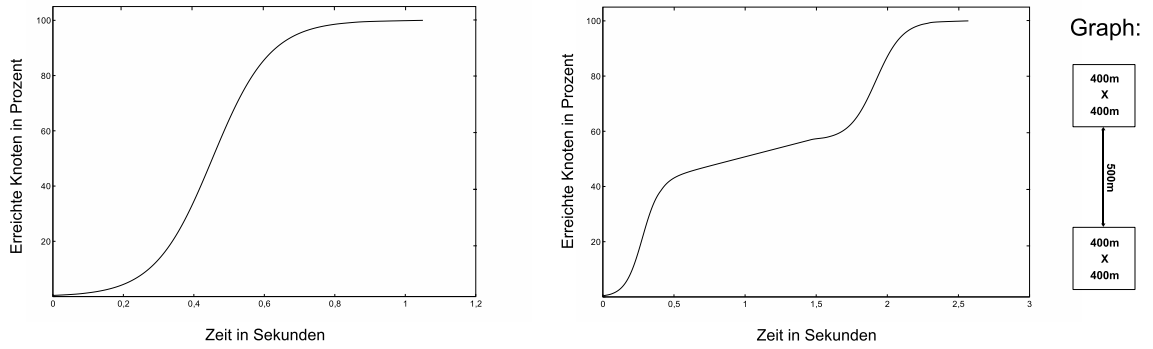
4.4 Datenverteilung im Area Graph-based Modell

Wir haben das Area Graph-based Bewegungsmodell entwickelt, um realistischere Simulations- und Emulationsergebnisse für die Evaluationen unserer Replikationsmethoden und des Gesamtsystems SYMORE zu erhalten. Für optimistische Multimaster Replikation ist die Betrachtung der Datenverteilung im AGM notwendig, da sich diese auf die Bildung von Konflikten auswirkt (siehe Abschnitt 2.1.2 und 2.3.4).

Im Gegensatz zu Festnetzen können mit Funktechnologie ausgerüstete mobile Knoten zunächst nur mit anderen Knoten in ihrer Funkreichweite kommunizieren. Aus diesem Grund verhält sich die Datenverteilung in MANETs ähnlich wie die Ausbreitung von Epidemien und lässt sich ebenfalls analytisch ermitteln, siehe [KBTR02, DH00]. Ein Beispiel für solch eine berechnete Verteilungskurve für das Random Waypoint Bewegungsmodell wird in Abbildung 4.7(a) gezeigt.

Im Gegensatz zu den einfachen und weitestgehend homogenen Bewegungsmodellen findet die Datenverteilung im Area Graph-based Modell sowohl in den Clustern als auch auf den Kanten dazwischen statt. Diese heterogene Struktur wirkt sich auf die Datenverteilung aus [Tsc06]. In Abbildung 4.7(b) ist die Verteilungskurve für ein Area Graph-based Modell mit zwei Clustern gezeigt, die jeweils ein Random Waypoint Modell beinhalten. Die berechnete Verteilungskurve basiert auf der in [Tsc06] vorgestellten Analyse der Datenverteilung im AGM.

Wie man erkennen kann, verläuft die Verteilung im AGM stufenweise. In den Clustern geschieht die Verteilung gemäß dem Mikromodell (Random Waypoint Modell) und zeigt die bereits in Abbil-



(a) Datenverteilung in einem Random Waypoint Modell (600m x 600m)

(b) Datenverteilung in einem Area Graph-based Modell mit zwei Clustern (Random Waypoint Modell)

Abbildung 4.7: Berechnete Datenverteilung (Fluten) in einem Netz mit 250 mobilen Knoten

dung 4.7(a) dargestellte epidemische Kurve. Auf den Kanten zwischen den Clustern ist die Verteilungskurve linear, was die Topologie der Kante widergespiegelt. In dem gezeigten Beispiel beginnt die Verteilung in einem der beiden Cluster. Dadurch ergeben sich die zwei Stufen der Kurve, da die Daten zuerst in einem und dann in dem anderen Cluster verteilt werden.

Eine weiterführende und detaillierte Besprechung der Datenverteilung im Area Graph-based Bewegungsmodell wird in [Tsc06] gegeben. Die Auswirkung dieser Eigenschaften des AGMs auf verschiedene Broadcastprotokolle wird in Kapitel 5 gezeigt. In Kapitel 10 werden die Auswirkungen des AGMs auf die Replikation betrachtet.

Synchronisationsbasierte Broadcastprotokolle

Bisher haben wir in diesem Abschnitt über die Datenverteilung mit Hilfe von Flutungsprotokollen gesprochen. Synchronisationsbasierte Broadcastprotokolle werden ebenfalls durch das Area Graph-based Modell beeinflusst, jedoch in geringerem Maße [Tsc06]. Synchronisationsbasierte Broadcastprotokolle zeigen in einfachen Bewegungsmodellen ebenfalls eine epidemische Verteilungskurve. Bei der Verteilung innerhalb eines AGMs zeigen die Verteilungskurven synchronisationsbasierter Broadcastprotokolle nur eine schwache Ausprägung von Stufen [Mar07]. Dies liegt an der verhältnismäßig langsamen Verteilung synchronisationsbasierter Broadcastprotokolle. Im Vergleich zur Datenverteilung in einfachen Bewegungsmodellen ist die Verteilung im AGM langsamer (vgl. ebd.).

4.5 Vergleich mit dem Graph-based Bewegungsmodell

Das AGM stellt eine Erweiterung des Graph-based Bewegungsmodells (GBM) dar und verwendet zusätzlich Mikromodelle für die Modellierung der Bewegung der Knoten in den Clustern. Dadurch

ergeben sich Unterschiede in der Datenverteilung. In Abbildung 4.8 wird die Datenverteilung der beiden Modelle für ein Szenario mit vier Clustern gezeigt.

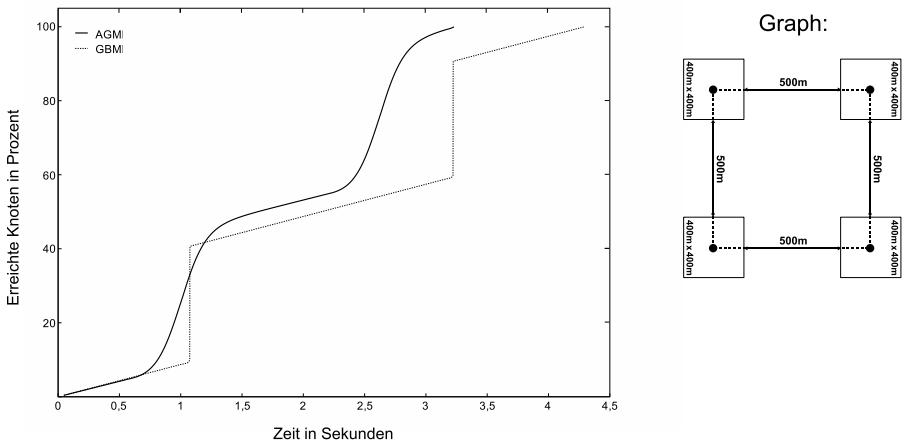


Abbildung 4.8: Berechnete Datenverteilung (Fluten) in einem AGM und einem GBM in einem ringförmigen Graph mit vier Clustern. Das modellierte Netz besteht aus 250 mobilen Knoten. Die Verteilung startet in der Mitte einer Kante.

Die Cluster werden im GBM als Punkte modelliert, die sich im Zentrum der Cluster des AGM befinden. Die Kanten des GBM werden dazu entsprechend verlängert (siehe Abbildung 4.8). Somit wird die Datenverteilung im GBM nur durch die Verteilung auf den Kanten des Graphen bestimmt. Wird ein Cluster erreicht, so erhalten alle mobilen Knoten innerhalb des Clusters die verteilten Daten zum gleichen Zeitpunkt, da sie sich aufgrund der Modellierung alle am selben Punkt befinden. Die Verteilung auf den Kanten des Graphen wird dadurch nicht verzögert.

Vergleicht man die beiden Verteilungskurven aus Abbildung 4.8, kann man sehen, dass sich die Verteilung innerhalb der Cluster unterscheidet und die Datenverteilung in diesem Fall im AGM schneller ist als im GBM. Im AGM werden bereits nach 3,2s alle mobilen Knoten im Netz erreicht. Im GBM werden alle Knoten nach 4,3s erreicht. Betrachtet man die durchschnittliche Datenverteilungsdauer (siehe Abschnitt 2.1.4), so ergibt sich ein ähnliches Bild: Im AGM wird ein mobiler Knoten im Mittel nach 1,8s erreicht. Im GBM liegt dieser Wert bei 2,2s.

Es gibt ebenfalls Szenarien, in denen die Verteilungskurven der beiden Bewegungsmodelle weniger voneinander abweichen. In Abbildung 4.9 ist ein Beispiel gezeigt, in der die Verteilungskurve des AGMs und des GBMs ähnlich verlaufen. Dadurch ergeben sich auch geringere Abweichungen in der Gesamtdauer und der mittleren Datenverteilungsdauer. Die Gesamtdauer (4,5s) und die mittlere Dauer (2,1s) sind in diesem Fall beim AGM größer als beim GBM (4,0s und 2,0s).

Im Allgemeinen werden die Unterschiede in der Datenverteilung der beiden Modelle durch die folgenden Eigenschaften eines zu modellierenden Szenarios bestimmt:

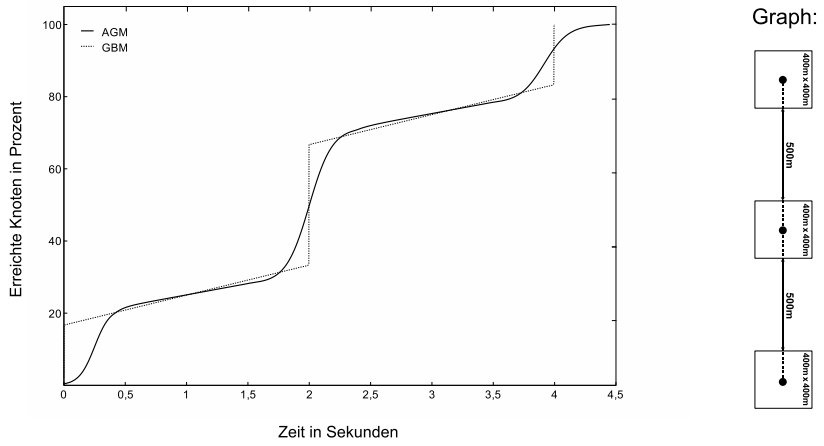


Abbildung 4.9: Berechnete Datenverteilung (Fluten) in einem AGM und einem GBM in einem Graphen mit drei Clustern. Das modellierte Netz besteht aus 250 mobilen Knoten. Die Verteilung startet in der Mitte eines der äußeren Cluster.

- **Cluster** Die Größe und Form der modellierten AGM Cluster beeinflusst die Unterschiede in der Datenverteilung der beiden Modelle: Je größer die Cluster eines modellierten Szenarios sind, desto größer ist auch der Bereich, in dem sich die Datenverteilung unterscheidet. Ebenso wirkt sich die Form der Cluster auf die Datenverteilung aus. Je nach Ausrichtung des Clusters kann dadurch die Datenverteilung im Gegensatz zum GBM schneller oder langsamer werden. Die Position der Verbindungspunkte eines Clusters beeinflusst ebenfalls die Art und die Geschwindigkeit der Datenverteilung des AGM im Vergleich zum GBM.

- **Anteil der Cluster** Die Unterschiede in der Datenverteilung der beiden Modelle ergeben sich nur innerhalb der Cluster. Aus diesem Grund ist der Anteil der Datenverteilung innerhalb der Cluster ein Maß, das die Unterschiede in der Datenverteilung des AGM und des GBM widerspiegelt. In Szenarien mit vielen, großen Clustern und wenigen, kurzen Kanten ergeben sich somit größere Unterschiede in der Datenverteilung.

- **Startpunkt der Verteilung** Startet die Datenverteilung innerhalb eines Clusters (siehe Abbildung 4.9), so ergibt sich am Anfang für das GBM eine schnellere Datenverteilung, da alle mobilen Knoten innerhalb des Clusters sofort erreicht werden. Startet die Verteilung hingegen auf einer Kante, so werden die Cluster im AGM früher erreicht (siehe Abbildung 4.8).

Die unterschiedliche Modellierung der Cluster im AGM und GBM beeinflusst ebenfalls die Bildung von Unterbrechungen und Partitionen des Funknetzes. Da die Cluster im GBM als Punkte modelliert werden, befinden sich die mobilen Knoten in diesem Punkt alle in gegenseitiger Funkreichweite und es ensteht für die Funkverbindungen ein vollständiger Graph. Gerade in größeren Clustern, die

Einkaufszentren, Museen oder Parkanlagen repräsentieren, kann es jedoch zu Unterbrechungen der Funkverbindungen der mobilen Knoten kommen. Ebenso ist in MANETs mit geringer Knotendichte häufiger mit Partitionen innerhalb eines Clusters zu rechnen. Dieser Aspekt wird vom GBM im Gegensatz zum AGM nicht berücksichtigt.

Abschließend ist festzuhalten, dass sich die Unterschiede zwischen dem Area Graph-based Modell und dem Graph-based Modell auf die Datenverteilung auswirken. Dadurch können sich für die Replikation relevante Unterschiede ergeben. Entgegen unseren Erwartungen sind die Unterschiede in bestimmten Szenarien nur gering.

4.6 Zusammenfassung

In diesem Kapitel wurden Bewegungsmodelle betrachtet, wie sie bei der Simulation von MANETs eingesetzt werden. Die Thematisierung von Bewegungsmodellen ist notwendig, da die Mobilität der Teilnehmer eines MANETs die Datenverteilung und somit auch die Replikation stark beeinflusst. Dies ist ein grundlegender Unterschied zur Replikation in Netzen mit fester Infrastruktur, wo der Aspekt der Mobilität nicht in diesem Maße berücksichtigt werden muss.

Insbesondere haben wir in diesem Kapitel das Area Graph-based Bewegungsmodell vorgestellt. Dieses Modell kombiniert das Graph-based Modell [THB+02] mit einfachen Bewegungsmodellen (Random Walk, Random Waypoint Modell, etc.), um eine realistischere Modellierung von Szenarien zu ermöglichen. Das Area Graph-based Bewegungsmodell besteht aus einem Makromodell, das durch einen gerichteten Graphen definiert wird. Die Knoten des Graphen (Cluster) haben im Gegensatz zum Graph-based Modell eine räumliche Ausdehnung und besitzen jeweils ein eigenes Mikromodell, um die Bewegung innerhalb ihrer Fläche zu modellieren. Diese Mikromodelle können entweder einfache Modelle oder wiederum Area Graph-based Bewegungsmodelle sein. Dadurch ist auch eine rekursive Definition eines Modells möglich, die die Betrachtung von unterschiedlichen Detailstufen ermöglicht.

Die Datenverteilung im Area Graph-based Bewegungsmodell unterscheidet sich von der Datenverteilung im Graph based Bewegungsmodell aufgrund der unterschiedlichen Modellierung der Graphknoten des jeweiligen Makromodells. Es ergeben sich für die Modellierung der gleichen Szenarien Unterschiede in der Datenverteilung. Wie stark sich die Datenverteilung unterscheidet, hängt von den Eigenschaften der Cluster, dem Anteil der Datenverteilung innerhalb der Cluster und dem Startpunkt der Datenverteilung ab.

Kapitel 5

Datenverteilung

Bei der Replikation in MANETs spielt die Datenverteilung eine wichtige Rolle. Durch die hohe Dynamik und die beschränkten Ressourcen von mobilen Geräten stellt die Datenverteilung in MANETs eine besondere Herausforderung dar. Dies ist ein Unterschied zu kabelgebundenen Festnetzen, in denen die Datenverteilung aufgrund der hohen Konnektivität des Netzes und der größeren Ressourcen der beteiligten Rechner anderen Rahmenbedingungen unterliegt.

Die Datenverteilung hat einen direkten Einfluss auf den Replikationsablauf: Je mehr Replikationsteilnehmer durch die Broadcastprotokolle schneller erreicht werden, desto geringer ist die Wahrscheinlichkeit für die kausal parallele Ausführung von Transaktionen. Somit wirkt sich die Datenverteilung auch auf die Bildung von Konflikten aus (siehe Abschnitt 2.3.4).

Aus den genannten Gründen ist es notwendig, die Datenverteilung in MANETs im Rahmen der Replikation zu betrachten. Wir untersuchen dazu typische Broadcastprotokolle, wie sie für die Verwendung in MANETs vorgeschlagen werden. Bei der Betrachtung bisheriger Protokolle zeigt sich, dass eine Adaption der Protokolle an die lokale Umgebung der mobilen Knoten einen positiven Einfluss auf die Verteilung ausübt. Aus diesem Grund haben wir auf der Basis bisheriger Protokolle zwei adaptive Broadcastprotokolle für die Datenverteilung in MANETs entwickelt und untersucht: ein adaptives Flutungsprotokoll [BRS05] und ein adaptives Synchronisationsprotokoll [BBHS05].

Für ein besseres Verständnis der Thematik geben wir am Anfang (Abschnitt 5.1) eine kurze Einführung zum Thema Datenverteilung in MANETs. Danach erläutern wir gebräuchliche Broadcastprotokolle und stellen unsere adaptiven Protokolle vor (Abschnitt 5.2 und 5.3). In Abschnitt 5.4 evaluieren wir die Protokolle.

5.1 Datenverteilung in MANETs

In den von uns betrachteten Replikationsszenarien benötigen alle Teilnehmer der Replikationsgruppe die zu replizierenden Daten (siehe Abschnitt 2.4). Das MANET eines Szenarios wird jeweils durch die Teilnehmer der Replikationsgruppe gebildet. In diesen Szenarien ist das Ziel der Datenverteilung, die lokalen Transaktionen (Datenänderungen) eines Teilnehmers an möglichst viele Netzteilnehmer (mobile Knoten) möglichst schnell zu verteilen.

Ebenso ist bei der Verteilung der Transaktionen wichtig, die Netzlast (Anzahl und Größe der verschickten Nachrichten) so gering wie möglich zu halten. In MANETs ist die Netzlast ein wichtiger Faktor, da die mobilen Geräte nur über begrenzte Ressourcen verfügen (siehe Abschnitt 2.1). Dies ist ein Unterschied zu Festnetzen, in denen die Netzlast eines Protokolls weniger relevant ist. Somit ergibt sich als Gesamtziel für die Datenverteilung in MANETs eine *effiziente* Verteilung, d. h. möglichst viele Netzteilnehmer in kürzester Zeit mit minimalen Kosten (Netzlast) zu erreichen.

Wie in Abschnitt 2.1 beschrieben, gibt es für das Verteilung von Daten in MANETs mit Hilfe von Broadcastprotokollen zwei unterschiedliche Ansätze. Beide Ansätze und unsere jeweiligen Protokolle werden in den nächsten Abschnitten (5.2 und 5.3) erläutert. Die hier beschriebene Datenverteilung basiert auf dem Versenden von Nachrichten, es werden keine Streams verwendet. In den von uns betrachteten Szenarien enthalten die Nachrichten die zu verteilenden Transaktionen oder Listen mit Transaktionskennungen (**Kennungslisten**) .

5.2 Flutungsprotokolle

Flutungsprotokolle basieren auf dem Senden und Weiterversenden einzelner Nachrichten. Ein einzelner Knoten startet die Verteilung, indem er die Nachricht an seine direkten Nachbarn verschickt. Diese schicken dann die Nachricht weiter, so dass schrittweise immer mehr Knoten erreicht werden. Wenn diese Art der Verteilung gewählt wird, können nur Knoten erreicht werden, die sich in derselben Partition wie der Startknoten befinden. Knoten außerhalb dieser Partition, die sich nicht in Funkreichweite eines Knotens dieser Partition befinden, erhalten die Nachricht nicht. Besonders in MANETs mit geringer Knotendichte stellt diese Einschränkung einen großen Nachteil dar. Ein Ansatz, um diesen Nachteil zu umgehen, ist das erneute Versenden der Nachrichten (Hyperflooding) nach einer gewissen Zeit.

5.2.1 Stand der Forschung

Die Datenverteilung durch Flutungsprotokolle ist eine häufig verwendete Methode. Aus diesem Grund liegt heute eine Vielzahl an Protokollen vor [KM05, WC02]. Die Protokolle können nach ihrem grundlegenden Ansatz und ihren unterschiedlichen Voraussetzungen klassifiziert werden (siehe [WC02, NTCS99]). Um einen aussagekräftigen Vergleich mit unserem adaptiven Flutungsprotokoll zu erhalten, stellen wir in den nächsten Abschnitten aus jeder Protokollklasse stellvertretend ein relevantes Protokoll vor. Umgebungsbasierte Protokolle[1] werden nicht vorgestellt, da die mobilen Knoten in unseren Szenarien keine Kenntnis über ihre geographischen Koordinaten (GPS) oder ihren Abstand zu anderen Knoten besitzen. Die vorgestellten Protokolle werden im Evaluationsteil (Abschnitt 5.4) als Vergleich verwendet.

[1] Protokolle, die Informationen über die Position eines Knotens (z. B. GPS Daten) und dessen näherer Umgebung (z. B. Entfernung zu anderen Knoten) verwenden.

Ein aktueller und allgemeiner Hyperflooding Ansatz ist das sogenannte Hypergossiping [Khe07]. Diese Methode nutzt eine effiziente Partitionserkennung und einen Parameter für die Lebensdauer der Nachricht, um Netzlast einzusparen. Die Einschränkung der Lebensdauer stellt ein Problem für die von uns betrachteten Replikationsszenarien dar, da hier die Transaktionen an alle Teilnehmer verteilt werden müssen und somit keine Nachrichten verloren gehen dürfen. Stattdessen verwenden wir zusätzlich zum Flutungsprotokoll ein synchronisationsbasiertes Broadcastprotokoll, das sich besser für die Datenverteilung in Replikationsszenarien anbietet und ebenfalls eine Verteilung über mehrere Netzpartitionen realisiert (siehe Abschnitt 2.1.2 und 5.3).

Eine Adaption von Flutungsprotokollen an die lokale Umgebung eines Knotens ist bereits für mehrere Klassen von Protokollen in [TNS03, RKHH04] und [CS03] betrachtet worden. Die in [CS03] beschriebenen Protokolle ähneln unserem Ansatz am meisten und folgen der gleichen Grundidee wie unser Protokoll: Sie passen die Wahrscheinlichkeit für das Weitersenden einer Nachricht an die Anzahl der direkten Nachbarn (single-hop) eines Knotens an. Im Unterschied zu unserem Ansatz wird dabei jedoch eine Gleichverteilung der Knoten auf der Grundfläche vorausgesetzt. Bei dem Random Waypoint Modell [BW02] und auch bei anderen Bewegungsmodellen ist jedoch eine Gleichverteilung der mobilen Knoten nur eingeschränkt vorhanden. Insbesondere bei dem von uns in den Untersuchungen verwendeten Area Graph-based Modell ist eine Gleichverteilung nicht gegeben. Aus diesem Grund wird das Protokoll in den Untersuchungen nicht betrachtet und ein eigener Ansatz vorgestellt.

Simple Flooding Protokoll

Der einfachste Ansatz, Daten in einem MANET zu verteilen, ist das einfache Fluten (engl. Simple Flooding) [HOTV99]. Bei diesem Protokoll werden alle Nachrichten Schritt für Schritt ohne Einschränkungen an die jeweiligen Nachbarn der Knoten weiterversendet, so dass am Ende alle Knoten der Partition die Nachricht erhalten haben.

Simple Flooding und alle nachfolgenden Flutungsprotokolle verwenden eine Nachrichtenhistorie, um feststellen zu können, ob Teilnehmer eine Nachricht bereits erhalten haben oder nicht. Ist dies der Fall, so wird die Nachricht nicht erneut versendet und ein endloses Weiterversenden verhindert. Somit endet die Verteilung, wenn alle erreichbaren Knoten die Nachricht erhalten haben. Durch seine Einfachheit und die redundante Verteilung ist das Simple Flooding ein sehr robustes Protokoll, das sich für die Datenverteilung in MANETs eignet.

Probabilistisches Flutungsprotokoll

Das probabilistische Flutungsprotokoll [NTCS99] gehört zur Klasse der *wahrscheinlichkeitsbasierten Methoden*. Das Protokoll funktioniert ähnlich wie das Simple Flooding. Das Weiterversenden der Nachricht geschieht jedoch nicht immer, sondern nur mit einer festgelegten Wahrscheinlichkeit p. Im ersten Schritt der Verteilung sendet der initiierende Knoten die Nachricht an alle seine Nach-

barn. Im nächsten und den folgenden Schritten wird die Nachricht von den Nachbarn jedoch nur mit der gegebenen Wahrscheinlichkeit weiterversendet.

In MANETs mit einer hohen Knotendichte überschneiden sich die Funkbereiche der Knoten oft, dadurch sind die Knoten häufig im Funkbereich mehrerer Nachbarknoten. Aus diesem Grund ist das Weiterversenden der Nachricht von allen Knoten meistens nicht notwendig. In diesem Fall spart das probabilistische Protokoll Nachrichten ein, ohne jedoch zwingend weniger Knoten zu erreichen. In MANETs mit einer geringen Knotendichte erreicht das Protokoll jedoch häufig nur vergleichsweise wenige Knoten, da einige notwendige Knoten die Nachricht zum Teil nicht weiterversenden. Wenn die Wahrscheinlichkeit p auf 100% [2] gesetzt ist, so verhält sich das Protokoll identisch zum Simple Flooding Protokoll.

Self-Pruning Flutungsprotokoll

Das Self-Pruning[3] Protokoll [LK00] gehört zur Klasse der *Methoden mit Nachbarschaftswissen*. Diese Protokolle nutzen die Informationen über ihre aktuellen Nachbarn. Das Self-Pruning Protokoll benötigt nur Informationen über seine direkten (single-hop) Nachbarn. Diese werden durch periodische „Hello“ Nachrichten zur Verfügung gestellt, die eine eindeutige Kennung des sendenden Knoten enthalten. Des Weiteren enthalten alle gesendeten Nachrichten zusätzlich eine Liste der Kennungen der Nachbarn des sendenden Knotens.

Wenn ein Knoten eine Nachricht von einem anderen Knoten erhält, so vergleicht er seine Nachbarn mit der Liste der Nachbarn des sendenden Knotens. Stellt seine eigene Liste eine Untermenge der Liste des sendenden Knotens dar, so wird die Nachricht nicht weiterversendet, da nicht zu erwarten ist, dass ein weiterer Knoten erreicht wird. Im anderen Fall versendet der empfangende Knoten die Nachricht weiter.

5.2.2 Adaptives Flutungsprotokoll

Das von uns entwickelte adaptive Flutungsprotokoll berücksichtigt die lokale Dichte des Netzes, um die Daten effizienter zu verteilen. Ebenso wie beim probabilistischen Flutungsprotokoll basiert die Verteilung unseres Protokolls auf einem Wahrscheinlichkeitswert p. Der Wert p ist im Gegensatz dazu jedoch nicht statisch, sondern passt sich jeweils an die lokale Dichte des Netzes an. Der aktuelle Wahrscheinlichkeitswert hängt von der Anzahl der Nachbarn n_s des sendenden Knotens s und der Anzahl der Nachbarn n_r des empfangenden (engl. receiving) Knotens r ab. Wir nehmen an, dass dem Protokoll die Information über die Anzahl der Nachbarn zur Verfügung steht (siehe Abschnitt 2.4.3).

[2] Aufgrund der häufigen Verwendung von Prozentwerten wird in dieser Arbeit das Prozentzeichen verwendet. Bei Vergleichen von Prozentwerten wird das Prozentzeichen für relative Prozentangaben verwendet. Absolute Vergleiche von Prozentwerten werden mit Hilfe des Begriffes „Prozentpunkt“ beschrieben.

[3] Reduktion

Diese wird ebenso wie beim Self-Pruning Protokoll mit Hilfe periodischer „Hello“ Nachrichten bereitgestellt.

Der Wert p, der die Wahrscheinlichkeit für ein Weiterversenden einer Nachricht angibt, wird so berechnet, dass er die Anzahl der weitersendenden Knoten beschränkt. Diese Beschränkung ist für Bereiche mit hoher Knotendichte gedacht, wo eine gewisse Anzahl von Knoten ausreicht, um eine Nachricht an alle Knoten in der Nachbarschaft zu verteilen. Diese Schranke ist durch den vorher festgelegten Schwellwert $x > 0$ gegeben. Ein Wert von $x = 10$ resultiert zum Beispiel bei einer Anzahl von 20 Nachbarn in $p = 50\%$. Daraus ergibt sich folgende Funktion.

Adaptive Verteilungsfunktion (einfache Version)

$$\begin{aligned} f &: \mathbb{N}_{>0} \rightarrow (0, 1] \\ f &: n_s \mapsto p = \min\left(1, \frac{x}{n_s}\right) \end{aligned} \tag{5.1}$$

Dieser erste Ansatz ist identisch mit der in [CS03] beschriebenen Grundidee und ist noch sehr einfach. Er funktioniert wie folgt: Wenn die Anzahl der Nachbarn n_s unter der gegebenen Schranke x liegt, wird die Wahrscheinlichkeit für das Weiterversenden für alle empfangenden Knoten auf 1 (100%) gesetzt, da die Verteilung in einem Gebiet mit niedriger Knotendichte ($< x$) stattfindet. Im anderen Fall wird der Wahrscheinlichkeitswert so gewählt, dass die zu erwartende Anzahl der weiterversendenden Knoten x entspricht.

Für viele Netztopologien reicht dieser Ansatz bereits aus, es gibt jedoch Fälle, in denen diese einfache Funktion nicht differenziert genug ist. So zum Beispiel, wenn der sendende Knoten n_s sehr viele und ein empfangender Knoten n_r sehr wenige Nachbarn besitzt, die zum Teil nur von ihm erreicht werden können. In diesem Fall wird für p ein sehr niedriger Wert bestimmt, und die Nachricht wird nur von sehr wenigen Nachbarn weiterversendet. Wenn nun weiter entfernte Knoten nur von einem dieser Nachbarn erreicht werden können, ist die Wahrscheinlichkeit sehr hoch, dass diese nicht erreicht werden.

Aus diesem Grund muss ebenfalls die Anzahl der Nachbarn des empfangenden Knotens berücksichtigt werden. Deshalb wird in der erweiterten Funktion $\min(n_s, n_r)$ verwendet, um den Wert von p zu bestimmen. Experimente haben gezeigt, dass die Verteilung in den Übergangsgebieten zwischen hoher und niedriger Knotendichte besonders häufig abbricht. Aus diesem Grund verwenden wir zusätzlich einen Faktor $\frac{\max(n_s,n_r)}{\min(n_s,n_r)}$, um den Wert von p in diesen Fällen weiter anzuheben. Diese Erweiterungen führen zu der endgültigen Formel des adaptiven Broadcastprotokolls.

Adaptive Verteilungsfunktion Die Wahrscheinlichkeit p für das Weiterversenden eines empfangenden Knotens n_r, der eine Nachricht von Knoten n_s erhalten hat, wird durch folgende

Funktion bestimmt:

$$f : (\mathbb{N}_{>0}, \mathbb{N}_{>0}) \rightarrow (0, 1]$$
$$f : (n_s, n_r) \mapsto p = \min\left(1, \frac{x \cdot \max(n_s, n_r)}{\min(n_s, n_r)^2}\right) \tag{5.2}$$

Mit Hilfe dieser Formel wird vom adaptiven Flutungsprotokoll für jede eingehende Nachricht eines Knotens entschieden, ob diese weiterversendet wird oder nicht.

5.3 Synchronisationsprotokolle

Diese Art der Protokolle verteilt die Daten im Netz durch eine paarweise durchgeführte Synchronisation der Netzteilnehmer. Schritt für Schritt werden so durch die einzelnen Synchronisationen die lokal ausgeführten Transaktionen an alle anderen Teilnehmer verteilt. Das ist zwar erheblich langsamer als eine auf Fluten basierende Verteilung [Khe07], jedoch werden im Gegensatz dazu alle Netzteilnehmer nach einer gewissen Zeit erreicht, da Synchronisationsprotokolle durch temporäre Netzpartitionen kaum beeinträchtigt werden.

5.3.1 Stand der Forschung

Stellvertretend für ähnliche Synchronisationsprotokolle anderer Systeme besprechen wir in Abschnitt 5.3.1 das von Bayou verwendete Anti-Entropy Protokoll [PST+97], da es in beliebigen Netzen verwendbar ist. Protokolle anderer Systeme wie Coda [SKK+90], Ficus [Ric92] oder das von Golding [Gol92] eingeführte Zeitstempel Anti-Entropy Protokoll besitzen jeweils nur einen Teil der Funktionalität und werden deshalb an dieser Stelle nicht weiter behandelt.

In [LW04] und [DQA04] sind zwei weitere Ansätze synchronisationsbasierter Datenverteilung beschrieben. Beide Ansätze verfolgen jedoch nicht eine vollständige Verteilung der Daten an alle Teilnehmer, sondern versuchen, die Daten wie bei einem Cache selektiv zu verteilen, um bestimmte Anfragen der Knoten möglichst lokal zu beantworten. In den jeweiligen Untersuchungen fehlt eine Betrachtung der Netzlast der Protokolle.

In [PS00] wird ein System zur Datenverteilung und des Prefetching in mobilen Umgebungen vorgestellt. Eine experimentelle Untersuchung des Systems in [PS01] berücksichtigt zwar viele Aspekte, wie z. B. die Funkreichweite und die Kooperation der Knoten, es wird jedoch nicht die verursachte Netzlast des Protokolls betrachtet. Ebenso gibt es keine Adaption der Datenverteilung.

Ein Ansatz, der auf einem hybriden Flutungs- und Synchronisationsprotokoll basiert, wird von Datta, Hauswirth und Aberer in [DHA03] vorgestellt. Das beschriebene generische Protokoll ist jedoch nicht speziell auf die Verteilung in MANETs ausgelegt und nutzt daher nicht die Eigenschaften von MANETs wie die lokale Knotendichte oder die funkbasierte Broadcast Übertragung, wie es unsere Protokolle tun.

Zwei weitere Protokolle, die für Sensor- bzw. ad-hoc Netze konzipiert wurden, sind in [HKB99] und [HBR03] beschrieben. Die Protokolle ähneln beide den von uns vorgestellten Protokollen, nutzen aber die Eigenschaften der funkbasierten Broadcast Übertragung nicht aus.

Zusammenfassend kann man sagen, dass es bereits viele Ansätze für die synchronisationsbasierte Datenverteilung gibt. Diese berücksichtigen jedoch entweder nicht die erzeugte Netzlast, beinhalten keine Adaption oder nutzen nicht die Eigenschaften der funkbasierten Broadcast Übertragung für ein „Mithören“ der Funknachrichten aus.

Anti-Entropy Protokoll

Das Anti-Entropy Protokoll [PST+97] von Bayou basiert auf einer paarweise durchgeführten Synchronisation und implementiert eine einseitige Synchronisation der aktuellen Operationshistorien zweier verbundener Knoten. Der Ablauf des Protokolls ist einfach. Zuerst schickt ein bestimmter Teilnehmer (Empfänger) eine Liste mit den Kennungen der ihm bekannten Schreiboperationen an den Sender. Der Sender vergleicht diese mit den Kennungen seiner lokalen Schreiboperationen und sendet danach alle Operationen, die dem Empfänger unbekannt sind, an denselbigen. Nach Abschluss des Protokolls sind dem Empfänger alle Schreiboperationen des Senders bekannt.

Obwohl das Anti-Entropy Protokoll in MANETs einsetzbar ist, ist dessen Verfahren sehr allgemein und berücksichtigt keine MANET spezifischen Aspekte. In einem MANET ist es z. B. nicht wie im Festnetz möglich, als Synchronisationspartner einen beliebigen Netzteilnehmer zu wählen, da ein mobiler Knoten ohne Verwendung eines Routingprotokolls nur Kontakt zu seinen direkten Nachbarn hat. Darüberhinaus ist das Protokoll eng mit dem Bayou System verwoben und beinhaltet Funktionalitäten (Synchronisation mit Hilfe von Datenträgern, Hinzufügen und Entfernen eines Replikationsteilnehmers), die im Rahmen dieser Arbeit nicht benötigt werden.

Aus diesem Grund haben wir ein Synchronisationsprotokoll implementiert, das auf den Ideen des Anti-Entropy Protokolls basiert. Wir verwenden dieses statische Protokoll, um es mit unserem adaptiven Synchronisationsprotokoll zu vergleichen.

5.3.2 Statisches Synchronisationsprotokoll

Wie bereits erwähnt, basiert das von uns entwickelte statische Synchronisationsprotokoll auf Bayous Anti-Entropy Protokoll. Es verwendet für alle Netzteilnehmer ein vorher festgelegtes Synchronisationsintervall, das bestimmt, in welchen Abständen ein Synchronisationsvorgang von den Teilnehmern gestartet wird.

Ist das Synchronisationsintervall für einen Teilnehmer abgelaufen, so sendet dieser eine Synchronisationsaufforderung an alle direkten Nachbarn. Diese enthält eine Liste der Kennungen der Transaktionen, die der Teilnehmer bereits besitzt. Das Versenden an alle Nachbarn erzeugt im Vergleich zum Versand an einen Nachbarn keine Extrakosten, da bei funkbasierter Datenübertragung die Daten immer als lokaler Broadcast (ungerichtet) versendet werden. Die Nachbarn schicken daraufhin

dem Knoten (Empfänger) ihre jeweiligen Kennungslisten zurück. Dieser wählt dann den Teilnehmer mit den meisten ihm unbekannten Transaktionen und fordert diesen (Sender) auf, ihm die genannten Transaktion zuzuschicken. Ebenso wie beim Anti-Entropy Protokoll sendet der Sender die angeforderten Transaktion an den Empfänger, so dass dieser am Ende ebenfalls alle Transaktionen des Senders kennt. Alle Nachbarn des Senders empfangen die „mitgehörten“ Transaktionen ebenfalls und speichern diese, sofern sie ihnen zuvor nicht bekannt waren. Um unnötige Datenübertragungen zu vermeiden, werden nach einem globalen Commit die Kennungen der festgeschriebenen Transaktionen aus den Kennungslisten gelöscht.

5.3.3 Adaptives Synchronisationsprotokoll

Das von uns entwickelte adaptive Synchronisationsprotokoll macht sich die Eigenschaften funkbasierter Datenübertragung zunutze. In einem MANET ist gerichtete Kommunikation nicht möglich. Jede gesendete Nachricht ist ein lokaler Broadcast und kann somit von allen anderen Teilnehmern in Funkreichweite empfangen werden. Diese Eigenschaft wird ausgenutzt, um das Synchronisationsintervall eines Teilnehmers dynamisch an die Aktualität seiner Daten anzupassen.

Die grundlegende Idee dieses Verfahrens ist wie folgt: Alle Teilnehmer merken sich in der Nachbarschaft übertragene Kennungslisten, auch wenn diese nicht direkt für den Teilnehmer bestimmt waren. Bei jedem Synchronisationsvorgang verschicken ein Teilnehmer und all seine Nachbarn ihre Kennungslisten. Diese Informationen sind nicht nur für die direkt an der Synchronisation beteiligten Teilnehmer nützlich. Die anderen Netzteilnehmer vergleichen die „mitgehörten“ Kennungslisten ebenfalls mit der eigenen Historie und stellen fest, wie viele Transaktionen (u) ihnen davon unbekannt sind. Die Initiierung eines Synchronisationsvorgangs wird durch einen Schwellwert y bestimmt. Gilt $u > y$, startet der Teilnehmer einen Synchronisationsvorgang. Der Schwellwert y des adaptiven Synchronisationsprotokolls beeinflusst somit die Reaktionsgeschwindigkeit des Protokolls.

Die eigentliche Synchronisation wird identisch zum statischen Synchronisationsprotokoll durchgeführt. Das adaptive Protokoll ändert nur das Synchronisationsintervall aber nicht die Art der Synchronisation. Für den Fall, dass nur sehr wenig Kommunikation zwischen den Teilnehmern stattfindet und somit nicht genug Kennungslisten „mitgehört“ werden können, verwendet das adaptive Protokoll ebenfalls ein festes Synchronisationsintervall. Dieses ist jedoch erheblich länger als das der statischen Protokolle, da es nur in Grenzfällen (seltene Initiierung von Transaktionen) notwendig ist.

5.4 Evaluation der Broadcastprotokolle

In diesem Abschnitt evaluieren wir die beiden von uns entwickelten Broadcastprotokolle. Dazu untersuchen wir das Verhalten der Protokolle in verschiedenen Szenarien mit unterschiedlichen Para-

metern (Anzahl der mobilen Knoten, Knotendichte, Bewegungsmodell, etc.) und vergleichen diese mit anderen bereits vorgestellten Protokollen. Aufgrund der Tatsache, dass Flutungs- und Synchronisationsprotokolle weitestgehend unabhängig voneinander arbeiten, werden die Protokolle jeweils getrennt bewertet.

5.4.1 Experimentelle Untersuchung des adaptiven Flutungsprotokolls

Um unser adaptives Flutungsprotokoll zu evaluieren, führen wir Simulationen mit den folgenden Flutungsprotokollen durch:

- Simple Flooding Protokoll
- Self-Pruning Flutungsprotokoll
- Probabilistisches Flutungsprotokoll mit den Wahrscheinlichkeitswerten 20%, 40%, 60% und 80%
- Adaptives Flutungsprotokoll mit den Schwellwerten 5, 6, 7, 8 ,9 und 10 [4]

Das Ziel der Untersuchung ist, die Protokolle bezüglich ihrer *Effizienz* zu vergleichen. Um die Effizienz eines Protokolls zu bewerten, betrachten wir die folgenden Zielgrößen:

- **Erreichte Knoten** Diese Größe gibt den Anteil der erreichten Knoten an, d. h., wie weit eine Transaktion im Netz verteilt wurde. Sie repräsentiert den *Nutzen* eines Protokolls. Je höher der Anteil der erreichten Knoten, desto größer der Nutzen eines Protokolls. Diese Zielgröße wird stark durch die Topologie und Knotendichte eines Netzes beeinflusst.
- **Netzlast** Die Netzlast gibt das durchschnittliche Datenvolumen (Summe der Nachrichtengrößen) an, das von einem Knoten während der Verteilung einer Transaktion empfangen und gesendet wurde. Sie repräsentiert die *Kosten* eines Protokolls. Je geringer der Wert der Netzlast, desto geringer sind die Kosten eines Protokolls. Die Netzlast wird relativ (pro Knoten) angegeben.

Das Verhältnis dieser beiden Zielgrößen zueinander gibt die Effizienz eines Protokolls an. Somit hat ein effizientes Protokoll eine gutes Kosten-Nutzen Verhältnis, d. h., es erreicht einen hohen Anteil der Knoten im Netz und verursacht eine geringe Netzlast.

In der Evaluation liegt der Fokus auf folgenden Aspekten:

- *Einfluss der Knotendichte* auf die Protokolle.

[4] Adaptive Protokolle mit anderen Werten werden nicht weiter untersucht, da vorausgegangene Experimente gezeigt haben, dass diese entweder zu wenige Knoten erreichen (Werte < 5) oder in zu hohen Netzlasten (Werte > 10) resultieren.

- *Einfluss der Bewegungsmodelle* auf die Protokolle.

Wir vermuten für beide Parameter einen starken Einfluss auf die Ergebnisse unseres adaptiven Protokolls. Daraus ergeben sich folgende Hypothesen.

Hypothese 1 (Einfluss der Knotendichte) Wir erwarten, dass das adaptive Flutungsprotokoll in Szenarien mit einer hohen Knotendichte im Vergleich zu den anderen Protokollen die besten Ergebnisse liefert. Aufgrund seiner dynamischen Anpassung der Verteilungswahrscheinlichkeit kann das Protokoll in diesen Szenarien viele unnötige Nachrichten einsparen und verursacht nur eine geringe Netzlast. In Szenarien mit einer geringen Knotendichte verschickt das adaptive Protokoll aufgrund der wenigen Nachbarn die Nachrichten mit sehr hoher Wahrscheinlichkeit. Aus diesem Grund erwarten wir im Vergleich zu den anderen Protokollen in diesen Szenarien ähnliche Ergebnisse.

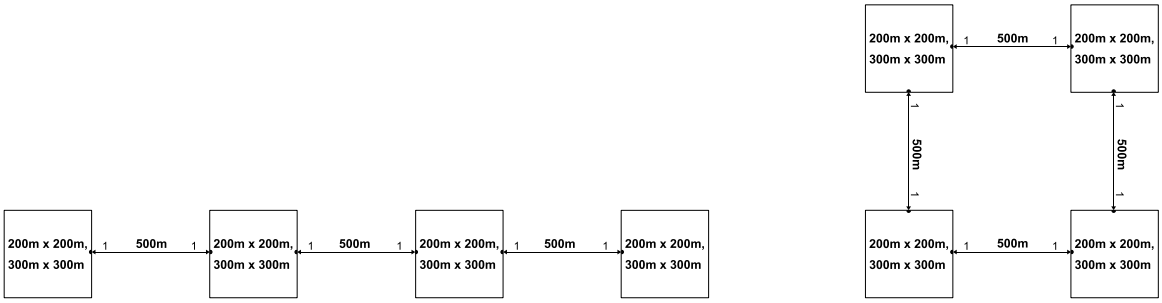
Hypothese 2 (Einfluss des Bewegungsmodells) Wir vermuten, dass das verwendete Bewegungsmodell einen starken Einfluss auf den Vergleich der verschiedenen Flutungsprotokolle hat. Besonders bei der Verwendung des Area Graph-based Bewegungsmodells erwarten wir signifikante Effizienzunterschiede zwischen unserem adaptiven und den anderen Protokollen. Aufgrund der heterogenen Topologie des AGMs und der daraus resultierenden heterogenen Knotendichte wirkt sich die Adaption unseres Protokolls im Gegensatz zu den anderen Protokollen stärker aus.

Aufbau der Experimente

Die experimentelle Untersuchung betrachtet die zuvor beschriebenen Protokolle in einem MANET-Szenario. Aufgrund der Größe der untersuchten Szenarien (siehe unten) wird als Simulator OMNeT++ [Var01] eingesetzt, da dieser im Vergleich zu anderen Simulatoren besser mit einer großen Anzahl von mobilen Knoten umgehen kann [WLW09]. Das simulierte Netz verwendet für die Kommunikation ein IEEE 802.11 [Com99] konformes MAC Protokoll mit einer Bandbreite von 11Mbit/s und einer Funkreichweite von 30m. Das MAC Protokoll verwendet für die Vermeidung von Funkbereichüberlappungen ein CSMA/CA[5] Schema. Somit ist die Überlappung von Funksignalen eingeschränkt, das hidden Node Problem[6] bleibt jedoch bestehen. Um Funkbereichüberlappungen aufgrund parallel sendender Knoten zu vermeiden, wird das Senden jeweils um eine zufällig gewählte Dauer (0s - 0,3s) verzögert.

Wir nehmen an, dass jedes Protokoll die Adressen und somit auch die Anzahl seiner direkten Nachbarn kennt. Praktisch kann dies durch „Hello“ Nachrichten gewährleistet werden (siehe z. B. AODV [PR99]). Für jedes einzelne Experiment (Kombination von Protokoll und Szenario) geben wir die Ergebnisse von 500 Einzelläufen an.[7] In jedem Durchlauf wird die Datenverteilung einer

[5] Carrier Sense Multiple Access/Collision Avoidance (Kollisionsvermeidung für Knoten in Funkreichweite)
[6] Überlappung zweier Funkbereiche von Knoten, die sich gegenseitig nicht in Funkreichweite befinden.
[7] Der Standardfehler ist für alle Ergebnisse kleiner als 1 Prozentpunkt.



(a) Linienförmiges Modell (b) Kreisförmiges Modell

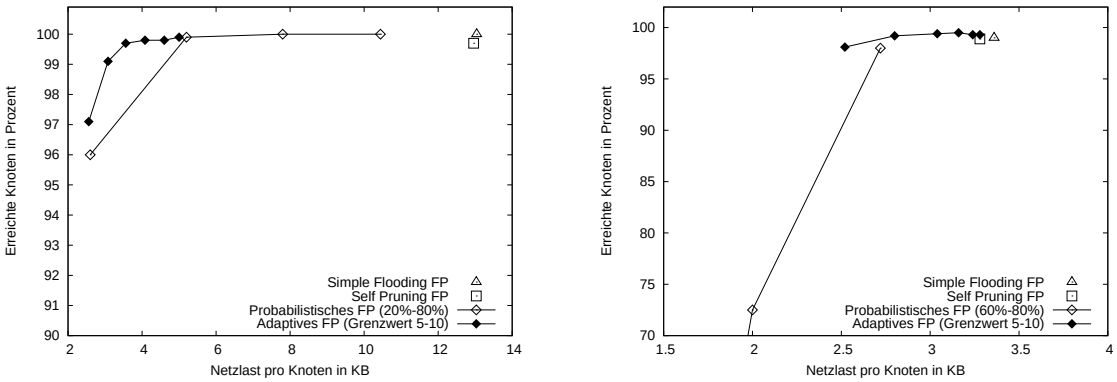
Abbildung 5.1: Verwendete Area Graph-based Modelle

Nachricht mit einer Größe von 400 Byte betrachtet, die von einem zufällig und gleichverteilt gewählten Knoten startet, der eine Transaktion lokal ausgeführt hat. Alle Szenarien simulieren ein Netz mit 2000 mobilen Knoten, die jeweils einen Fußgänger darstellen, der mit einem mobilen Gerät ausgerüstet ist. Wir betrachten die Szenarien als ein isoliertes System, in dem die Knoten das Szenario nicht verlassen bzw. ausfallen können. Diese Annahme hat keine signifikante Auswirkung auf die Ergebnisse, da in einem Einzellauf nur jeweils ein einzelnes Fluten durchgeführt wird, das nur wenige Sekunde benötigt, um alle Knoten der betreffenden Partition zu erreichen. Dabei ist nicht relevant, ob ein Knoten temporär oder dauerhaft unerreichbar ist.

Nr.	**Bewegungsmodell**	**Beschreibung**
1	Random Waypoint Modell	Größe: 400m $\times$ 400m (hohe Knotendichte)
2	Random Waypoint Modell	Größe: 600m $\times$ 600m (mittlere Knotendichte)
3	Random Waypoint Modell	Größe: 800m $\times$ 800m (niedrige Knotendichte)
4	Area Graph-based Modell (Linie)	Clustergröße: 200m $\times$ 200m mit kurzer Aufenthaltsdauer
5	Area Graph-based Modell (Linie)	Clustergröße: 200m $\times$ 200m mit langer Aufenthaltsdauer
6	Area Graph-based Modell (Linie)	Clustergröße: 300m $\times$ 300m mit kurzer Aufenthaltsdauer
7	Area Graph-based Modell (Linie)	Clustergröße: 300m $\times$ 300m mit langer Aufenthaltsdauer
8	Area Graph-based Modell (Kreis)	Clustergröße: 200m $\times$ 200m mit kurzer Aufenthaltsdauer
9	Area Graph-based Modell (Kreis)	Clustergröße: 200m $\times$ 200m mit langer Aufenthaltsdauer
10	Area Graph-based Modell (Kreis)	Clustergröße: 300m $\times$ 300m mit kurzer Aufenthaltsdauer
11	Area Graph-based Modell (Kreis)	Clustergröße: 300m $\times$ 300m mit langer Aufenthaltsdauer

Tabelle 5.1: Übersicht über die untersuchten Szenarien

In Tabelle 5.1 ist eine Übersicht über alle untersuchten Szenarien gegeben. Drei Szenarien basieren auf dem Random Waypoint Modell. Sie besitzen unterschiedliche Knotendichten, die aufgrund der verschieden großen Grundflächen entstehen. Weiterhin wurden in der Untersuchung zwei Area Graph-based Modelle verwendet (siehe Abbildung 5.1). Diese wurden jeweils in ihrer Clustergröße und der Aufenthaltsdauer in den Clustern variiert. Dadurch wird sowohl die Knotendichte in den



(a) Szenario mit hoher Knotendichte

(b) Szenario mit niedriger Knotendichte

Abbildung 5.2: Untersuchungsergebnisse der Flutungsprotokolle für das Random Waypoint Modell

Clustern als auch die Differenz der Knotendichte eines Clusters und einer Verbindung verändert. In den folgenden zwei Abschnitten diskutieren wir die Ergebnisse von vier repräsentativen Szenarien. Die Ergebnisse anderer Szenarien werden bei Bedarf erwähnt.

Untersuchungsergebnisse (Random Waypoint Modell)

In Abbildung 5.2 sind die Ergebnisse der Szenarien 1 und 3 dargestellt. Auf der x-Achse ist die Netzlast aufgetragen und die y-Achse stellt Anteil der erreichten Knoten dar. Somit steigt die Effizienz in diesem Koordinatensystem von rechts unten nach links oben. Punkte stellen jeweils eine einzelne Konfiguration des Protokolls dar. Protokolle einer Gruppe (probabilistische und adaptive Protokolle) werden zur Verdeutlichung verbunden. Die Protokolle sind jeweils entsprechend ihres Parameterwertes von links unten (Wert 5 für adaptive und 20% für probabilistische Protokolle) nach rechts oben (Wert 10 und 80%) angeordnet.

In Abbildung 5.2(a) (hohe Knotendichte, Ø 32,9 Nachbarn), ist zu erkennen, dass fast alle Protokolle mehr als 99% der mobilen Knoten erreichen. Die Netzlast liegt zwischen 3,1 KB (adaptives Flutungsprotokoll-6)[8] und 13 KB (Simple Flooding). Im Gegensatz zu allen anderen Protokollen produzieren die adaptiven Flutungsprotokolle wesentlich weniger Netzlast und erreichen ähnlich viele oder im Vergleich zum Self-Pruning Protokoll sogar mehr mobile Knoten im Netz. Nur das probabilistische Protokoll-40%[9] erzeugt ebenfalls eine vergleichsweise geringe Datenlast (5,2 KB).

Dieses Verhalten lässt sich direkt aus den Eigenschaften der Protokolle ableiten. Während die adaptiven Protokolle die Anzahl der weitersendenden Knoten auf eine ausreichende Anzahl begrenzen,

[8] Zur Verbesserung der Lesbarkeit verwenden wir für die Benennung eines Protokolls, das einen bestimmen Parameter verwendet, eine Kurzschreibweise. In diesem Fall wird ein adaptives Flutungsprotokoll mit Schwellwert 6 benannt.

[9] Bei der Benennung von probabilistischen Protokollen wird ebenfalls die Kurzschreibweise verwendet.

erlauben die anderen Protokolle zu vielen mobilen Knoten das erneute Versenden der Nachricht. Das führt zu den beobachteten Unterschieden in der Netzlast der Protokolle. Obwohl das Self-Pruning Protokoll ebenfalls seine Nachbarknoten kennt und seine Verteilung daran anpasst, erzeugt es mehr Netzlast als die adaptiven Protokolle.

In Abbildung 5.2(b) sind die Ergebnisse des Szenarios mit einer geringen Knotendichte (Ø 8,5 Nachbarn) dargestellt. Außer den probabilistischen Protokollen mit geringer Wahrscheinlichkeit (20% - 60%) erreichen alle Protokolle einen Anteil von über 98% der mobilen Knoten. Erneut arbeiten die adaptiven Protokolle effizienter als die anderen Protokolle. Sie erreichen eine vergleichbare Anzahl an mobilen Knoten, sparen dabei jedoch im Gegensatz zum Simple Flooding Protokoll bis zu 25% der Netzlast ein. Das Self-Pruning Flutungsprotokoll verursacht die gleiche Netzlast wie das adaptive Flutungsprotokoll-10, erreicht jedoch etwas weniger Knoten.

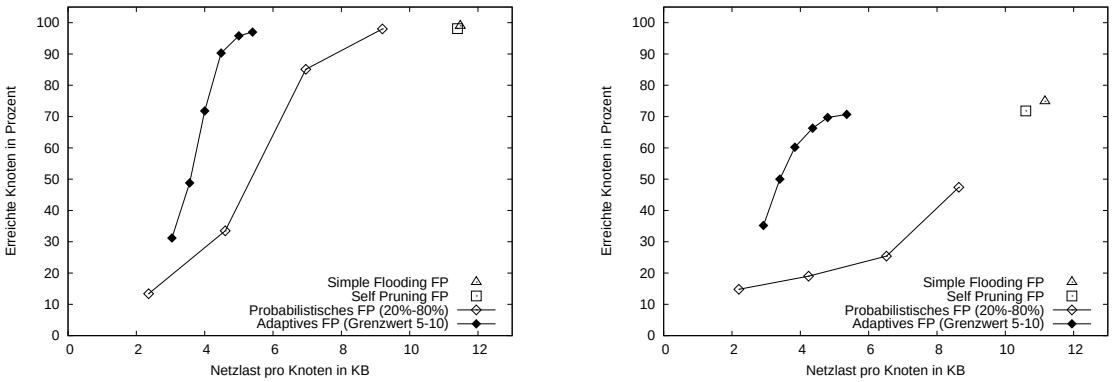
Der Vergleich der beiden Szenarien zeigt, dass die adaptiven Protokolle bei einer hohen Knotendichte mehr Netzlast einsparen. Da die erwartete Anzahl der weiterversendenden Knoten durch das Protokoll je nach Schwellwert auf 5 bis 10 limitiert wird, ergibt sich bei einer hohen Knotendichte im ersten Szenario (Ø 32,9 Nachbarn) auch eine stärkere Reduktion. Dahingegen werden bei der Knotendichte des zweiten Szenarios (Ø 8,5 Nachbarn) die adaptiven Protokolle mit den Schwellwerten 8 bis 10 in den meisten Fällen nicht limitiert, da die Schwellwerte über der tatsächlichen Anzahl der Nachbarn liegen. Trotzdem wird auch dort von den Protokollen Netzlast eingespart, da die lokale Dichte zum Teil höher ist als der Mittelwert von 8,5 Nachbarn. Dieser Fall tritt jedoch aufgrund der eher homogenen Knotenverteilung des Random Waypoint Modells nur relativ selten auf.

Schlussfolgerung Die Ergebnisse der probabilistischen Protokolle (20% - 80%) variieren sehr stark und es gibt keine einzige Variante, die in beiden Szenarien genügend Knoten erreicht und effizient arbeitet. Die probabilistischen Protokolle erzeugen entweder zu viel Netzlast oder erreichen nicht genügend mobile Knoten, um effektiv eingesetzt zu werden. Das Simple Flooding und das Self-Pruning Protokoll zeigen ähnliche Ergebnisse. Beide erreichen einen hohen Anteil an Knoten, aber produzieren eine hohe Netzlast. Die durch das Self-Pruning Verfahren eingesparte Netzlast ist gering (< 3%).

Die adaptiven Protokolle erreichen vergleichbar viele Knoten wie das Simple Flooding Protokoll, produzieren dabei aber weniger Netzlast (3% - 70% weniger) als die Vergleichsprotokolle. Aufgrund der limitierten Anzahl der weiterversendenden Knoten sparen die adaptiven Protokolle in Szenarien mit höheren Knotendichten mehr Netzlast ein als in Szenarien mit geringerer Knotendichte.

Untersuchungsergebnisse (Area Graph-based Modell)

In Abbildung 5.3 sind die Ergebnisse der Szenarien 8 und 9 dargestellt. Die linke Abbildung zeigt das Szenario, in dem die mobilen Knoten nur kurz in den Clustern verweilen. Dies resultiert in einem



(a) Szenario mit kurzer Aufenthaltsdauer im Cluster

(b) Szenario mit langer Aufenthaltsdauer im Cluster

Abbildung 5.3: Untersuchungsergebnisse der Flutungsprotokolle für das kreisförmige Area Graph-based Modell

geringen Unterschied zwischen der Knotendichte in den Clustern (Ø 32,7 Nachbarn) und auf den Kanten (Ø 20,9 Nachbarn). In Szenario 9 (rechte Abbildung) halten sich die Knoten in den Clustern wesentlich länger auf. Dies resultiert in einem größeren Unterschied zwischen der Knotendichte in den Clustern (Ø 30,3 Nachbarn) und auf den Kanten (Ø 11,7 Nachbarn).

Die Ergebnisse der oberen Abbildung zeigen große Unterschiede im Anteil der erreichten Knoten, der von 13% (probabilistisches Protokoll-20%) bis 99% (Simple Flooding Protokoll) reicht. Ein akzeptabler Anteil (mehr als 95%) wird vom Simple Flooding, dem Self-Pruning, dem probabilistischen Protokoll-80% und den adaptiven Protokollen-9/10 erreicht.

Während der Anteil der erreichten Knoten dieser Protokolle nur um wenige Prozentpunkte variiert, gibt es signifikante Unterschiede in der resultierenden Netzlast. Das Simple Flooding (11,48 KB) und das Self-Pruning Protokoll (11,4 KB) verursachen die höchste Netzlast. Das probabilistische Protokoll spart ungefähr 19% dieser Netzlast ein. Die adaptiven Protokolle-9/10 benötigen jedoch weniger als 50% der Netzlast des Simple Flooding und des Self-Pruning Protokolls. Aufgrund der heterogenen Topologie und Dichte des Area Graph-based Modells kommt die Adaption unseres Protokolls stärker zum Tragen und spart damit noch mehr Netzlast ein als in Szenarien mit homogener Topologie.

In der unteren Abbildung sind die Ergebnisse des AGMs mit einer geringen Knotendichte auf den Verbindungen dargestellt. Aufgrund dieser geringer Knotendichte wird eines der vier Cluster häufig nicht erreicht. Der Abstand zwischen den mobilen Knoten auf den Verbindungen ist zum Teil größer als deren Funkreichweite und somit wird die Verteilung an dieser Stelle unterbrochen. Diese Eigenschaft des Szenarios bewirkt, dass weniger Knoten im Netz erreicht werden. Somit erreicht selbst das Simple Flooding Protokoll nur 75% und das Self-Pruning Protokoll nur 72% aller mobilen Knoten. Die adaptiven Protokolle-9/10 erreichen eine etwas geringere Anzahl an Knoten (70% und 71%),

verursachen jedoch weniger als die Hälfte der Datenlast des Simple Flooding oder des Self-Pruning Protokolls. Sämtliche probabilistischen Protokolle erreichen einen nicht mehr ausreichenden Anteil der Knoten von weniger als 50%.

Schlussfolgerung Die heterogene Topologie (Cluster gegenüber Verbindungen) und Knotendichte des Area Graph-based Bewegungsmodells wirkt sich stark auf die Leistung der Protokolle aus. Probabilistische Protokolle erweisen sich für solche Szenarien als nicht geeignet, da sie zu wenige Knoten erreichen. Der Grund dafür ist die statische Festlegung der Wahrscheinlichkeit für das Weiterversenden, die sich nicht an die heterogene Dichte des Szenarios anpasst. Das Simple Flooding und das Self-Pruning Protokoll erreichen einen großen Anteil der Knoten, produzieren jedoch viel unnötige Netzlast. Die adaptiven Protokolle verhalten sich im Vergleich zu den anderen Protokollen erneut am effizientesten. Sie benötigen weniger als 50% der Netzlast, um vergleichbar viele oder etwas weniger (1% - 4%) Knoten zu erreichen. Im Gegensatz zu den Szenarien des Random Waypoint Modells erreichen bei den AGM Szenarien nur die adaptiven Protokolle mit den Schwellwerten 9 und 10 einen ausreichenden Anteil der Knoten.

Zusammenfassung und Überprüfung der Hypothesen

Die Evaluation unserer adaptiven Flutungsprotokolle bestätigt weitestgehend Hypothese 1 (Einfluss der Knotendichte) und Hypothese 2 (Einfluss des Bewegungsmodells). Unser Protokoll profitiert von einer höheren Knotendichte, da in diesem Fall im Vergleich zu den anderen Protokollen mehr Datenlast eingespart werden kann. Sogar in Szenarien mit einer geringen Knotendichte arbeiten die adaptiven Protokolle am effizientesten, die Ergebnisse heben sich dabei jedoch nicht so stark von denen der anderen Protokolle ab.

Während die Ergebnisse der adaptiven Protokolle in den Random Waypoint Szenarien noch zum Teil ähnlich zu den Ergebnissen der anderen Protokolle sind, gibt es bei den Szenarien des Area Graph-based Modells signifikante Unterschiede. Das bestätigt den ersten Teil unserer zweiten Hypothese. Der zweite Teil der Hypothese trifft jedoch nicht zu, da die adaptiven Protokolle nicht zusätzlich von Szenarien mit einer besonders starken Heterogenität (siehe Abbildung 5.3(b)) profitieren. In diesen Szenarien erreichen die Protokolle sogar im Vergleich zum Simple Flooding etwas weniger Knoten (bis zu 4%).

Während andere Protokolle durch verschiedene Einflussgrößen (Knotendichte, Bewegungsmodell) stark beeinflusst werden, zeigt sich bei den adaptiven Protokollen ein geringerer Einfluss. Besonders die adaptiven Protokolle mit den Schwellwerten 9 und 10 erreichen vergleichbar viele Knoten und eine geringe Netzlast in allen untersuchten Szenarien. Während sie im Vergleich zum Simple Flooding und Self-Pruning Protokoll ähnlich viele oder etwas weniger Knoten erreichen, sparen sie mindestens 20% bis 50% (abhängig vom Szenario) der Netzlast ein. Protokolle mit geringeren Schwellwerten sind im Hinblick auf die Replikationsszenarien nicht sinnvoll, da sie zu wenige

Knoten erreichen. Protokolle mit höheren Werten eignen sich ebenfalls nicht, da siehe eine zu hohe Netzlast verursachen (siehe Fußnote S. 69).

5.4.2 Experimentelle Untersuchung des adaptiven Synchronisationsprotokolls

Um unser adaptives Synchronisationsprotokoll zu evaluieren, führen wir experimentelle Untersuchungen mit den folgenden Protokollen bzw. Protokollkombinationen durch:

- Statisches Synchronisationsprotokoll mit einem Synchronisationsintervall von 30s, 60s, 90s, 120s, 180s, 240s, 300s, 360s
- Simple Flooding Protokoll in Kombination mit dem statischen Synchronisationsprotokoll (Intervalle: 30s, 60s, 90s, 120s, 180s, 240s, 300s, 360s). Dabei werden zunächst die Transaktionen nach der lokalen Ausführung per Fluten im Netz verteilt. Zusätzlich wird das Synchronisationsprotokoll verwendet, um weitere Teilnehmer zu erreichen, welche die Transaktion durch das Flutungsprotokoll noch nicht erhalten haben.
- Adaptives Synchronisationsprotokoll mit den Schwellwerten 0, 1, 2, 4 und 8.
- Simple Flooding Protokoll in Kombination mit dem adaptiven Synchronisationsprotokoll (Schwellwerte: 0, 1, 2, 4 und 8).

Das Ziel dieser Untersuchung ist, die Protokolle bezüglich ihrer *Effizienz* zu vergleichen. Um die Effizienz eines Protokolls zu bewerten, betrachten wir die folgenden Zielgrößen:

- **Durchschnittliche Datenverteilungsdauer** Die Datenverteilungsdauer gibt die Zeit an, die benötigt wird, um eine Nachricht von einem an einen anderen mobilen Knoten zu senden. Sie repräsentiert den *Nutzen* eines Protokolls. Je kürzer die Datenverteilungsdauer, desto größer der Nutzen eines Protokolls.

  Der Durchschnittswert wird über die Einzelwerte aller Knotenpaare und Nachrichten gebildet. Bei der Untersuchung von Synchronisationsprotokollen ist die Verteilung der Daten über die Zeit relevant, deshalb wird eine andere Zielgröße verwendet als zuvor bei der Bewertung der Flutungsprotokolle.

- **Netzlast** Die Netzlast pro Knoten gibt das durchschnittliche Datenvolumen an, das von einem Knoten pro Transaktion empfangen und versendet wurde. Im Gegensatz zu den Untersuchungen der Flutungsprotokolle werden im Verlauf eines Simulationsdurchlaufs eine Reihe von Transaktionen von verschiedenen Teilnehmern initiiert und an die anderen Teilnehmer verteilt. Die Netzlast repräsentiert die *Kosten* eines Protokolls. Je geringer die Netzlast, desto geringer sind die Kosten eines Protokolls.

Das Verhältnis dieser beiden Zielgrößen zueinander gibt die Effizienz eines Protokolls an. In der folgenden Evaluation liegt der Fokus auf folgenden Aspekten:

- *Einfluss der Knotendichte* auf die Protokolle.
- *Einfluss der Transaktionsfrequenz*[10] auf die Protokolle.

Wir vermuten für beide Parameter einen starken Einfluss auf die Ergebnisse unseres adaptiven Protokolls und kommen somit zu den folgenden Hypothesen.

Hypothese 1 (Einfluss der Knotendichte) Wir erwarten, dass sich unser adaptives Synchronisationsprotokoll in Szenarien mit einer hohen Knotendichte effizienter verhält als das statische Protokoll. Aufgrund der hohe Knotendichte können auch mehr Statusinformationen „mitgehört“ werden, wovon das adaptive Protokoll stark profitiert. In Szenarien mit niedriger Knotendichte ist anzunehmen, dass sich der Einfluss der Adaption verringert und die adaptiven und die statischen Protokolle ähnliche Ergebnisse liefern.

Hypothese 2 (Einfluss der Transaktionsfrequenz) Wir nehmen an, dass sich eine höhere Transaktionsfrequenz positiv auf das Verhalten des adaptiven Synchronisationsprotokolls auswirkt. Das adaptive Protokoll reagiert auf die hohe Frequenz, und somit werden Synchronisationsvorgänge dynamisch angestoßen, was eine höhere Aktualität der Daten bewirkt. In Szenarien mit einer geringen Transaktionsfrequenz tritt dies nur selten auf und deshalb ist zu erwarten, dass sich das adaptive Protokoll ähnlich verhält wie das statische Protokoll. Wir nehmen außerdem an, dass sich die Verteilung (über die Zeit) der Initiierung der Transaktionen ebenfalls auf die Ergebnisse der Protokolle auswirkt. Während eine zufällige und gleichverteilte Initiierung der Transaktionen für statische Protokolle einfach zu handhaben ist, so benötigen andere Verteilungen (z. B. Normalverteilung) eine dynamische Anpassung der Synchronisationsintervalle, wie sie beim adaptiven Protokoll gegeben ist. Wir erwarten somit, dass sich die Unterschiede zwischen dem statischen und dem adaptiven Protokoll stärker in Szenarien zeigen, denen *keine* gleichverteilte Initiierung der Transaktionen zugrunde liegt.

Aufbau der Experimente

Im Rahmen der Untersuchung werden die im letzten Abschnitt beschriebenen Synchronisationsprotokolle für verschiedene MANET-Szenarien simuliert. Als Simulator wird ns2 [ns10] verwendet, da aufgrund seiner weiten Verbreitung die Vergleichbarkeit der Messergebnisse verbessert wird und nur MANETs mittlerer Größe simuliert werden, bei denen sich die schlechte Leistung des Simulators bei großen Knotenmengen [WLW09] nur wenig auswirkt. Das simulierte Netz verwendet für

[10] Frequenz, mit der die Transaktionen von den Teilnehmern initiiert werden.

die Kommunikation ein IEEE 802.11 [Com99] konformes MAC Protokoll mit einer Bandbreite von 11 Mbit/s und einer Funkreichweite von 100m. Ein Knoten kann nur mit anderen Knoten kommunizieren, die sich innerhalb dieser Funkreichweite befinden. Jede Nachricht, die eine Transaktion beinhaltet, besitzt eine eindeutige Kennung und hat eine Gesamtgröße von 256 Byte. Dies entspricht in etwa der Datenmenge, die für die Übertragung einer einfachen Transaktion benötigt wird. Die Szenarien haben jeweils eine Dauer von einer Stunde, in der eine gegebene Anzahl von Transaktionen verteilt wird. Für jede zu verteilende Transaktion wird jeweils ein mobiler Knoten zufällig und gleichverteilt als Initiator aus der Gesamtmenge aller Knoten ausgewählt. Die Initiierung der Transaktionen über die Dauer des Szenarios wird durch eine Zufallsvariable bestimmt.

Alle untersuchten Szenarien basieren auf einem Random Waypoint Bewegungsmodell[11] mit einer Größe von 1000m $\times$ 1000m. Wie in den vorangegangenen Untersuchungen betrachten wir ein isoliertes System. Eine Übersicht über die untersuchten Szenarien ist in Tabelle 5.2 gegeben. Wir haben insgesamt 12 Szenarien mit einer unterschiedlichen Anzahl an Knoten (50, 100, 150), Transaktionen (100, 400) und unterschiedlichen Verteilungsfunktionen (gleichverteilt, normalverteilt) für die Initiierung der Transaktionen untersucht.

Szenario	**Anz. d. mobilen Knoten**	**Anz. d. Transaktionen**	**Verteilung d. Transaktionen**
1	50	100	Gleichverteilt
2	50	100	Normalverteilt
3	50	400	Gleichverteilt
4	50	400	Normalverteilt
5	100	100	Gleichverteilt
6	100	100	Normalverteilt
7	100	400	Gleichverteilt
8	100	400	Normalverteilt
9	150	100	Gleichverteilt
10	150	100	Normalverteilt
11	150	400	Gleichverteilt
12	150	400	Normalverteilt

Tabelle 5.2: Übersicht der untersuchten Szenarien

Im Folgenden diskutieren wir die Ergebnisse von drei repräsentativen Szenarien. Die Ergebnisse der anderen Szenarien werden bei Bedarf erwähnt.

[11] Entsprechende Untersuchungen mit dem Area Graph-based Modell wurden von Danny Tschirner in [Tsc06] durchgeführt. Dabei wurde ein Szenario jeweils mit dem Area Graph und dem Random Waypoint Modell simuliert. Es hat sich gezeigt, dass das adaptive Synchronisationsprotokoll bei der Verwendung des Random Waypoint Modells effizienter ist.

Untersuchungsergebnisse

In Abbildung 5.4 sind die Ergebnisse der Synchronisationsprotokolle für die Szenarien 6, 10 und 12 gezeigt. Jeder Punkt im Koordinatensystem stellt die Konfiguration eines Protokolls mit einem bestimmten Parameterwert (Synchronisationsintervall, Schwellwert) dar. Die Netzlast ist auf der x-Achse und die durchschnittliche Datenverteilungsdauer auf der y-Achse aufgetragen. Die Effizienz steigt in dem Koordinatensystem somit von rechts oben nach links unten.

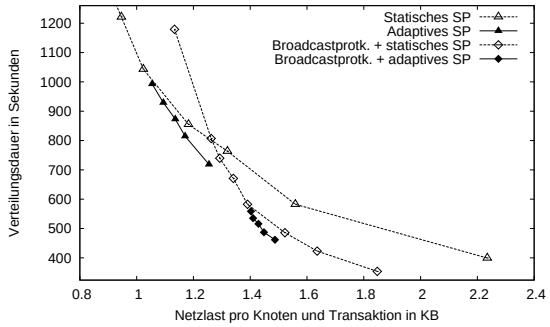
Jedes Diagramm beinhaltet vier Kurven, die jeweils die vier Protokollvarianten darstellen. Jede Kurve besteht aus den unterschiedlich parametrisierten Protokollen einer bestimmten Variante. So bestehen die Kurven, welche die adaptiven Protokolle darstellen immer aus fünf Punkten, die jeweils die Protokolle mit den verschiedenen Schwellwerten (0,1,2,4,8) darstellen. Erwartungsgemäß sind die Protokolle jeweils entsprechend ihrem Schwellwert von unten (Schwellwert 0) nach oben (Schwellwert 8) angeordnet. Analog sind die Kurven, welche die statischen Protokollvarianten repräsentieren, entsprechend ihren Synchronisationsintervallen geordnet.

Die Ergebnisse der Protokolle bilden jeweils eine Kurve mit abnehmendem Wachstum. Der Grund dafür ist, dass der Austausch von Synchronisationsinformationen ebenfalls eine gewisse Datenlast verursacht, unabhängig davon, ob Transaktionen ausgetauscht werden. Häufiger durchgeführte Synchronisationsversuche verursachen zum Teil unnötige Datenlast ohne dabei die Datenverteilungsdauer zu verringern.

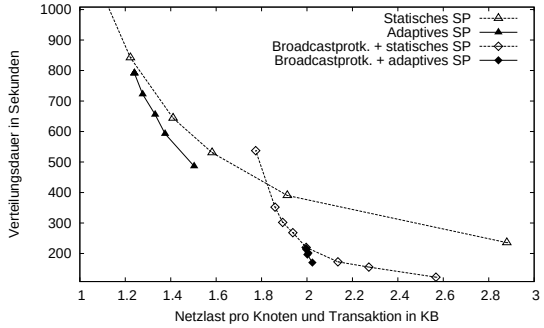
Betrachten wir zunächst den Unterschied zwischen den „reinen“ Synchronisationsprotokollen und den mit Flutungsprotokollen kombinierten Varianten. Zunächst ist zu erkennen, dass die Kombination mit einem Flutungsprotokoll in einer schnelleren Datenverteilung resultiert aber auch eine größere Netzlast verursacht. Dies liegt an der Tatsache, dass Flutungsprotokolle alle Teilnehmer in derselben Partition sehr schnell und ohne zusätzliche Netzlast für die Abstimmung der Synchronisation erreichen. In Szenarien mit einer höheren Knotendichte (5.4(b) und 5.4(c)) verstärkt sich dieser Effekt noch weiter. Das Flutungsprotokoll profitiert von den resultierenden größeren Partitionen und hat somit einen stärkeren Einfluss.

Ein Vergleich zwischen dem statischen und dem adaptiven Synchronisationsprotokoll (sowohl einzeln als auch in Kombination) zeigt deutliche Unterschiede. In allen vorgestellten Szenarien zeigt das adaptive Protokoll bessere Ergebnisse als das statische Protokoll. Besonders in den Szenarien mit einer höheren Knotendichte (5.4(b) und 5.4(c)) sind die adaptiven Protokolle deutlich effizienter und sparen bei gleicher Datenverteilungsdauer bis zu 13% der Netzlast ein. Aufgrund der hohen Knotendichte können die Knoten mehr Synchronisationsinformationen „mithören“ und somit kann das adaptive Protokoll schneller reagieren. Simulationen mit geringen Knotendichten (50 mobile Knoten) zeigen keinen signifikanten Unterschied zwischen den Ergebnissen des statischen und des adaptiven Synchronisationsprotokolls.

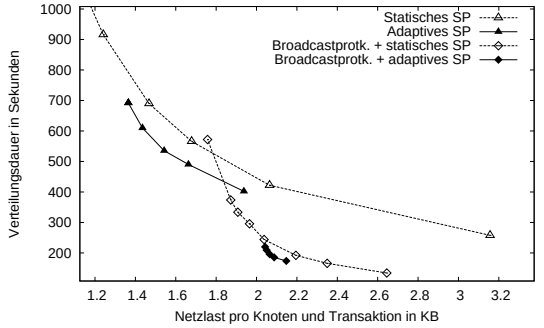
Weiterhin ist zu beobachten, dass die adaptiven Protokolle von einer höheren Transaktionsfrequenz profitieren (siehe Abbildung 5.4(c)). Der Grund dafür ist, dass die höhere Anzahl an verteil-



(a) Szenario mit 100 mobilen Knoten und 100 Transaktionen



(b) Szenario mit 150 mobilen Knoten und 100 Transaktionen



(c) Szenario mit 150 mobilen Knoten und 400 Transaktionen

Abbildung 5.4: Untersuchungsergebnisse der Synchronisationsprotokolle für die Szenarien 6, 10, und 12

ten Transaktionen auch mehr Statusinformationen verursacht; diese können wiederum vom adaptiven Protokoll genutzt werden. Der Unterschied zwischen den beiden Protokollvarianten (statisch, adaptiv) in Kombination mit einem Flutungsprotokoll ist etwas geringer (bis zu 7%). Dies liegt daran, dass das adaptive Protokoll nur die Verteilung für die Knoten verbessern kann, die nicht in der Partition des initiierenden Knoten liegen, denn diese wurden bereits durch das Flutungsprotokoll erreicht. Das erklärt auch, warum der positive Einfluss des adaptiven Protokolls in Szenarien mit einer höheren Knotendichte geringer ist.

Es wurden ebenfalls Szenarien mit einer gleichverteilten Initiierung (über die Dauer des Szenarios) der Transaktionen untersucht. Diese Verteilung ist für die statischen Protokolle einfacher zu handhaben und deshalb ist der Unterschied zwischen den beiden Synchronisationsprotokollen (statisch, adaptiv) in diesen Fällen sehr gering. Trotzdem verhält sich das adaptive Protokoll auch in diesen Szenarien effizienter als das statische Protokoll.

Schlussfolgerung und Überprüfung der Hypothesen

Die Untersuchungsergebnisse bestätigen die aufgestellten Hypothesen. Wie erwartet profitiert das adaptive Protokoll von einer höheren Knotendichte, da diese ebenfalls eine höhere Informationsdichte zur Folge hat, die „mitgehört“ werden kann. Auch der Einfluss einer höheren Transaktionsfrequenz und der Verteilung der Transaktionsinitiierung zeigt sich deutlich. Aus diesem Grund ist der Effizienzgewinn der adaptiven Synchronisationsprotokolle in dem Szenario mit einer hohen Transaktionsfrequenz und einer normalverteilten Transaktionsinitiierung am höchsten. So werden bei gleicher Datenverteilungsdauer von den adaptiven Protokollen bis zu 13% der Netzlast der statischen Protokolle eingespart.

Auch wenn die Synchronisationsprotokolle mit einem Flutungsprotokoll kombiniert werden, wird immer noch bis zu 7% der Netzlast eingespart. In Szenarien mit geringeren Knotendichten und einer Gleichverteilung der Initiierung der Transaktionen ist die Einsparung durch die adaptiven Protokolle geringer. Trotzdem verhalten sich die adaptiven Protokolle in allen untersuchten Szenarien effizienter als die statischen Synchronisationsprotokolle.

Im Hinblick auf die Replikation wird nur das Protokoll mit dem Schwellwert 0 verwendet. Dadurch werden die Daten möglichst schnell im Netz verteilt und die Wahrscheinlichkeit für die nebenläufige Ausführung von Transaktionen verringert.

5.5 Zusammenfassung

In diesem Kapitel wurde die für die Replikation notwendige Datenverteilung in MANETs betrachtet. Dazu haben wir sowohl Flutungs- als auch Synchronisationsprotokolle untersucht. Insbesondere wurden von uns zwei neue adaptive Protokolle für die Datenverteilung in MANETs vorgestellt: das *adaptive Flutungsprotokoll* und das *adaptive Synchronisationsprotokoll*. Beide Protokolle wurden

von uns entwickelt, um die Datenverteilung in MANETs möglichst *effizient* durchzuführen.

Um die Leistung der beiden Protokolle zu evaluieren, wurden experimentelle Untersuchungen mit unterschiedlichen MANET-Szenarien durchgeführt. In den Untersuchungen wurden weitere bekannte Protokolle betrachtet und mit unseren Protokollen verglichen. Die Ergebnisse der Experimente zeigen, dass das adaptive Flutungsprotokoll in den untersuchten Szenarien im Vergleich zu den anderen Protokollen bis zu 50% der Netzlast einspart. Auch das adaptive Synchronisationsprotokoll verteilt die Daten in allen untersuchten Szenarien effizienter als das Vergleichsprotokoll und spart bis zu 13% der Netzlast ein. Der Nutzen der Verteilung (erreichte Knoten, Datenverteilungsdauer) kann durch diese Protokolle jedoch nicht gesteigert werden. Das liegt daran, dass die Datenverteilung durch die Topologie des MANETs stark begrenzt wird. Während im Festnetz die Datenverteilung durch die hohe Konnektivität bei Bedarf beschleunigt werden kann, so sind in einem MANET durch die häufigen und langen Partitionierungen des Netzes den Protokollen wesentlich weniger Einflussmöglichkeiten gegeben. Mit sinkender Knotendichte des Netzes werden diese Möglichkeiten noch weiter eingeschränkt.

Beide adaptiven Broadcastprotokolle werden von dem von uns in dieser Arbeit ebenfalls vorgestellten Replikationssystem SYMORE verwendet. Dadurch können die Daten effizienter verteilt werden. Im Hinblick auf die Replikation werden die Schwellwerte der beiden Protokolle so festgelegt, dass die Daten möglichst weit und schnell im Netz verteilt werden (adaptives Flutungsprotokoll Schwellwert 10, adaptives Synchronisationsprotokoll Schwellwert 0). Somit wird die Wahrscheinlichkeit, dass Konflikte auftreten, möglichst gering gehalten. Die Schwellwerte werden aus den oben genannten Gründen (starker Einfluss der MANET Topologie) nicht angepasst.

Kapitel 6

Uhrensynchronisation

Die von uns vorgestellten Replikationsmethoden sind für Echtzeit-Zeitstempel konzipiert. Diese Zeitstempel werden von den Konfliktlösungsmethoden verwendet, um Konflikte zwischen den Transaktionen auf eine für den Benutzer transparente und intuitive Art zu lösen. Für jeden Benutzer ist nachvollziehbar, wenn im Falle eines Konflikts die älteste oder (je nach Konfliktlösung) die jüngste Transaktion gewinnt.

Für die Verwendung von Echtzeit-Zeitstempeln ist die Synchronisation der Uhren der Replikationsteilnehmer notwendig[1]. In diesem Kapitel stellen wir ein für MANETs konzipiertes Synchronisationsverfahren für Echtzeituhren vor. Dieses ermöglicht eine leichtgewichtige Synchronisation der Uhren der Replikationsteilnehmer. Weiterhin kann mit Hilfe des Verfahrens für beliebige Zeitstempel bestimmt werden, ob diese tatsächlich in Echtzeitreihenfolge angeordnet sind oder ob aufgrund der Ungenauigkeit der Uhren keine Echtzeitreihenfolge garantiert werden kann.

6.1 Problemstellung

Wenn die Uhren mehrerer Teilnehmer eines Netzes synchronisiert werden sollen, so ist die grundsätzliche Vorgehensweise meist die, eine (extern oder intern) vorgegebene Zeit an alle Teilnehmer zu verteilen. Das Problem bei dieser Verteilung ist, dass bei jeder Übertragung der Uhrzeit eine leichte Verzögerung (engl. message delay) auftritt, die nicht exakt zu bestimmen ist. Aufgrund dieser Verzögerung kann es vorkommen, dass Ereignisse (z. B. pre-Commit einer Transaktion), die einen Echtzeit-Zeitstempel besitzen, nicht in Echtzeitreihenfolge geordnet sind. Ein Beispiel dafür ist in Abbildung 6.1 angegeben.

Teilnehmer s_1 schickt um 12:00:00 Uhr seiner lokalen Zeit diese Uhrzeit an den Teilnehmer s_2. Dieser erhält den Wert zwei Sekunden später und übernimmt ihn, so dass die lokale Zeit der beiden Teilnehmer um zwei Sekunden versetzt ist. Beide Teilnehmer führen jeweils eine Transaktion aus, die mit einem Zeitstempel der lokalen Zeit versehen ist. Anschließend werden die Transaktionen jeweils an den anderen Teilnehmer geschickt. Die Transaktionen werden nun entsprechend

[1] Wir betrachten im Rahmen dieser Arbeit Szenarien, in denen die mobilen Geräte nicht mit GPS Empfängern ausgestattet sind (siehe Abschnitt 2.4.3).

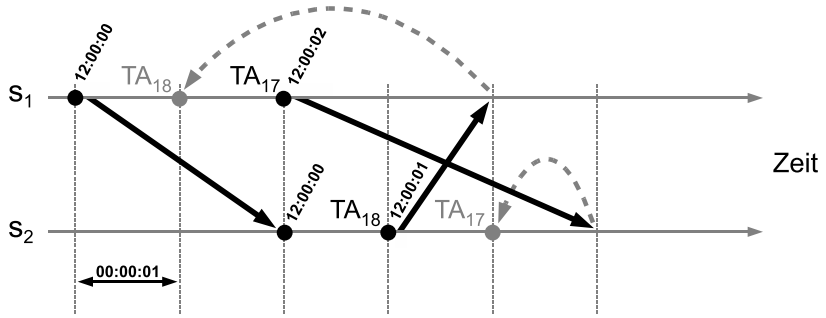


Abbildung 6.1: Problemstellung der Uhrensynchronisation

ihren Zeitstempeln geordnet. Aufgrund des Versatzes der Uhren wird Transaktion TA_{17} bei beiden Teilnehmern hinter TA_{18} eingeordnet, obwohl sie zeitlich vor TA_{18} ausgeführt wurde.

Obwohl diese Ordnung nicht einer Echtzeitreihenfolge entspricht, ist sie jedoch bei allen Teilnehmern gleich und kann somit für die Konfliktlösung eines Replikationssystems verwendet werden. Trotzdem ist es besonders für die Nutzer eines Replikationssystems wichtig zu wissen, ob für eine gegebene Menge von Transaktionen die Echtzeitreihenfolge garantiert werden kann. Wird z. B. ein Objekt überwacht und der aktuelle Zustand jeweils von der letzten Transaktion überschrieben, kann es vorkommen, dass der aktuelle Zustand nicht dem zuletzt beobachteten Wert entspricht. In einem Katastrophenszenario wäre ein Lager denkbar, dessen Vorräte überwacht werden. Kann für die Beobachtungsergebnisse des Lagers keine Echtzeitreihenfolge garantiert werden, so informiert das System die Einsatzkräfte darüber. Aus diesem Grund haben wir ein einfaches Uhrensynchronisationsverfahren entwickelt, das für zwei gegebene Echtzeit-Zeitstempel unterschiedlicher Teilnehmer bestimmen kann, ob eine Echtzeitreihenfolge garantiert ist. Bevor das eigentliche Verfahren erklärt wird, geben wir kurz eine Einführung in die Thematik und beleuchten den Stand der Forschung.

6.2 Einführung

Die Synchronisation von Echtzeituhren ist ein bekanntes Problem. Einen Überblick über bisherige Synchronisationsverfahren wie sie in lokalen Netzen oder im Internet verwendet werden, ist in [CDK02] gegeben. Die Güte der Synchronität zweier Uhren wird durch zwei Parameter gekennzeichnet: *Versatz* und *Drift*. Der **Versatz** (engl. skew) gibt die Differenz zweier Uhren zu einem bestimmten Zeitpunkt an. Er liegt für Uhren von Rechnern eines synchronisierten Netzes typischerweise im Bereich von wenigen Sekunden, kann aufgrund von fehlender Synchronisation jedoch auch mehrere Minuten groß werden. Die **Drift** gibt den Unterschied der Laufgeschwindigkeit einer Uhr zu einer Referenzuhr an, d. h. wie schnell diese Uhr im Vergleich zur anderen Uhr weiterzählt. Die Drift einer typischen Quartzuhr, wie sie heutzutage in Rechnern üblicherweise verwendet wird, beträgt laut [CDK02] 10^{-6} Sekunden pro Sekunde, das entspricht etwa einer Sekunde in 11,6 Tagen.

Aufgrund der kurzen Dauer (< 24 h) der in dieser Arbeit fokussierten MANET-Szenarien wird die Drift in den folgenden Betrachtungen vernachlässigt.

Das nachfolgende Verfahren betrachtet deshalb den Versatz der Uhren der einzelnen Teilnehmer zueinander. Obwohl die Uhren der Netzteilnehmer aneinander angeglichen werden können, ist eine exakte Synchronisation zweier Uhren grundsätzlich nicht möglich (siehe [LL84]). Dies liegt an der unbekannten Laufzeit bzw. an der Verzögerung von Nachrichten.

Die Synchronität von Uhren kann in zwei Klassen eingeteilt werden: *extern* und *intern* synchronisierte Uhren. Extern synchronisierte Uhren haben einen vorgegebenen maximalen Versatz im Bezug zu einer externen Referenzuhr. Intern synchronisierte Uhren haben einen vorgegebenen maximalen Versatz im Bezug zueinander.

Das bekannteste Synchronisationsprotokoll ist das Network Time Protocol (NTP) [Mil92]. NTP ist ein externes Synchronisationsprotokoll und eignet sich deshalb nicht für die in dieser Arbeit betrachteten MANET-Szenarien, da die Teilnehmer keine Verbindung zu einer externen Uhr besitzen. In einer Replikationsgruppe ist jedoch eine interne Synchronisation vollkommen ausreichend. Ein einfaches internes Uhrensynchronisationsprotokoll für MANETs ist die in [Com99] vorgestellte Timing Synchronisation Function. Differenziertere Protokolle für Sensornetze sind in [MBT04] und [Röm01] beschrieben. Diese beiden Ansätze führen eine kontinuierliche Synchronisation durch, um den Einfluss der Drift so gering wie möglich zu halten, was jedoch in den von uns betrachteten Szenarien nicht notwendig ist.

Das von uns vorgestellte Verfahren verwendet ein einfaches Synchronisationsprotokoll, das auf dem in [Cri89] vorgestellten Ansatz basiert. Es erweitert jedoch die bisherigen Ansätze durch die Möglichkeit, die Echtzeitreihenfolge der vergebenen Zeitstempel zu prüfen.

6.3 Synchronisationsverfahren

Die Uhrensynchronisation aller Teilnehmer der Replikationsgruppe ähnelt den Synchronisationsprotokollen der Datenverteilung. Es synchronisieren sich nur jeweils zwei Teilnehmer, wodurch die Uhrensynchronisation des gesamten Netzes epidemisch verläuft. Das hier dargestellte Verfahren ist eine vereinfachte Form des von uns in [SBH07] vorgestellten Ansatzes.

Die Synchronisation zweier Teilnehmer, zwischen denen eine direkte Verbindung (siehe Abschnitt 2.1) besteht, wird als **direkte Synchronisation** bezeichnet. Dafür wird folgendes leichtgewichtiges Protokoll verwendet:

Uhrensynchronisationsprotokoll

Sei $\delta_{\max}$ eine vorher festgelegte Obergrenze für die Dauer des Synchronisationsvorgangs. Sei $l(s_x)$ eine Funktion, die jeweils die aktuelle lokale Zeit eines Teilnehmers s_x zurückgibt. Das Synchronisationsprotokoll zwischen zwei Teilnehmern (s_1 und s_2) besteht aus den folgenden Einzelschritten:

1. Teilnehmer s_1 schickt seine aktuelle lokale Zeit $t_1 = l(s_1)$ an s_2.

2. Teilnehmer s_2 erhält den Wert t_1 um δ_{12} verzögert zum Zeitpunkt $t_2 = l(s_2)$ und berechnet anschließend die Differenz der beiden Uhren: $\mathrm{diff}_{12} = t_1 - t_2$. Danach schickt er eine Bestätigung mit seiner aktuellen lokalen Zeit $t_3 = l(s_2)$ und der Differenz diff_{12} an s_1 zurück.

3. Teilnehmer s_1 erhält die Bestätigung um δ_{21} verzögert zum Zeitpunkt $t_4 = l(s_1)$. Mit Hilfe der Differenz der beiden Zeiten diff_{12} kann s_1 die Summe der beiden Verzögerungen δ_{12} und δ_{21} berechnen: $\delta_{121} = \delta_{12} + \delta_{21} = t_4 - t_3 - \mathrm{diff}_{12}$.

4. Gilt $\delta_{121} \leq \delta_{\max}$, so sendet s_1 eine Abschlussbestätigung (ack) an s_2, wodurch s_2 die Differenz diff_{12} zu seiner lokalen Zeit addiert und somit die lokale Zeit von s_1 (inklusive Verzögerung) übernimmt. Gilt $\delta_{121} > \delta_{\max}$, so startet s_1 den Synchronisationsvorgang erneut.

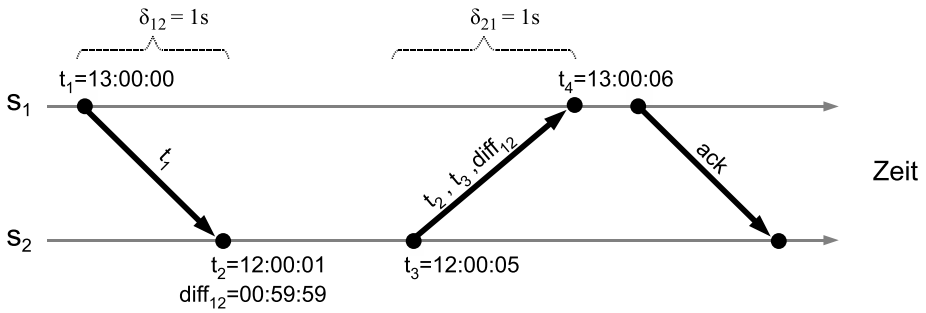


Abbildung 6.2: Ablauf des Synchronisationsprotokolls. Das Beispiel ergibt eine Gesamtverzögerung von $\delta_{121} = 2\mathrm{s} = 13{:}00{:}06 - 12{:}00{:}05 - 00{:}59{:}59$

Ein Beispiel für den Protokollablauf ist in Abbildung 6.2 angegeben. Nach einer *erfolgreichen* Ausführung des Protokolls ist durch $\delta_{\max}$ eine obere Grenze für die Sendeverzögerung δ_{12} von s_1 nach s_2 gegeben. Obwohl diese Abschätzung recht grob ist, stellt dies für den betrachteten Aufgabenbereich kein Problem dar. Untersuchungen einer Java Implementierung des Protokolls auf Windows PDAs haben eine Verzögerung (Hin- und Rücksenden) von 200ms – 300ms ergeben (siehe [RÖ06]). Wird das Protokoll auf einer unteren Abstraktionsschicht implementiert, kann dieser Wert jedoch noch erheblich verringert werden. Das Protokoll benötigt im Normalfall nur drei Schritte und ist somit leichtgewichtig und für den Einsatz in MANETs geeignet.

Durch die Mobilität der Teilnehmer wird bestimmt, welche Teilnehmer sich in Funkreichweite befinden und direkt synchronisieren können. Auf das Protokoll selbst hat die Mobilität keinen Einfluss, da eine Synchronisation (siehe [RÖ06]) im Normalfall weniger als eine Sekunde benötigt. Bewegt sich ein Teilnehmer innerhalb dieser Zeit aus dem Funkbereich des anderen, was bei den beschriebenen Szenarien mit Fußgängern sehr unwahrscheinlich ist, wird dies als Fehlerfall behandelt.

Im *Fehlerfall*, d. h. die Sendeverzögerung ist zu lang ($> \delta_{\max}$) oder falls es nach einer vorbestimmten Zeitspanne keine Rückmeldung gibt, wird das Protokoll erneut ausgeführt. Liegt eine länger anhaltende Störung der direkten Verbindung der beiden Teilnehmer vor, so wird das Protokoll je nach Konfiguration nach einer gewissen Anzahl von Fehlversuchen abgebrochen. Mögliche Ursachen dafür sind der Ausfall eines Teilnehmers oder ein zu großer Abstand, so dass keine direkte Funkverbindung mehr zwischen den Teilnehmern hergestellt werden kann. Die Synchronisation muss in diesem Fall zu einem späteren Zeitpunkt (ggf. mit einem anderen Teilnehmer) erneut durchgeführt werden. Ein dauerhafter Ausfall eines Teilnehmers wird von der Gruppenverwaltung behandelt (siehe Abschnitte 2.4.4 und 2.4.5).

Die Synchronisation der gesamten Replikationsgruppe baut auf diesem Protokoll auf und wird schrittweise durchgeführt. Ausgehend von einem bestimmten Teilnehmer (**Referenzknoten**) werden die Teilnehmer in **Synchronisationsschritten** paarweise synchronisiert, bis alle Teilnehmer der Replikationsgruppe erreicht sind. Da in den einzelnen Synchronisationsschritten jeweils die aktuelle Zeit synchronisiert wird, ist der zeitliche Abstand zwischen diesen Schritten aus Sicht der Gesamtsynchronisation nicht relevant. Ein Beispiel für die Gesamtsynchronisation wird Abbildung 6.3 gezeigt.

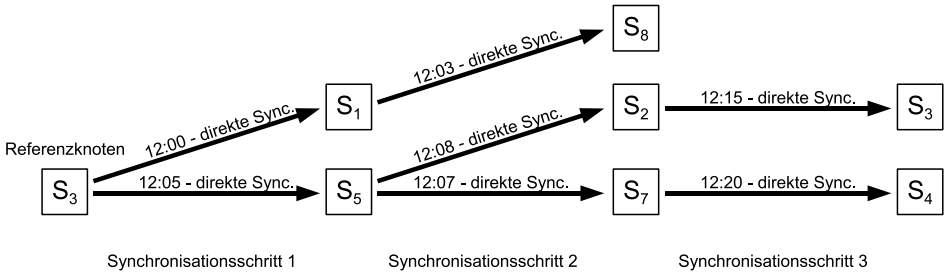


Abbildung 6.3: Schematischer Ablauf der Gesamtsynchronisation. Bei den einzelnen Synchronisationsschritten sind die Zeiten angegeben. Da jeweils die *aktuelle* Uhrzeit synchronisiert wird, kann sich die Gesamtsynchronisation auch über einen längeren Zeitraum erstecken.

Bei jedem Synchronisationsschritt ergibt sich durch die Sendeverzögerung eine größere Ungenauigkeit der Gesamtsynchronisation der Gruppe. Diese Ungenauigkeit lässt sich für eine gegebene Replikationsgruppe wie folgt abschätzen.

Gesamtverzögerung einer Replikationsgruppe

Für die Synchronisation einer Replikationsgruppe über maximal n Synchronisationsschritte und einer Verzögerung von maximal $\delta_{\max}$ für eine direkte Synchronisation, kann die **Gesamtverzögerung** wie folgt abgeschätzt werden:

$$\delta_{\mathrm{gmax}} = n \cdot \delta_{\max}$$

Entspricht $n + 1$ der Größe der Replikationsgruppe, so können sich die Teilnehmer beliebig syn-

chronisieren, da im ungünstigsten Fall eine Kette gebildet wird, die n Synchronisationsschritte benötigt. Um eine Erweiterung der Replikationsgruppe zu erlauben, muss n entsprechend größer gewählt oder darauf geachtet werden, dass sich die neuen Teilnehmer nur mit Teilnehmern synchronisieren, deren Uhrensynchronisation maximal $n-1$ Synchronisationsschritte vom Referenzknoten entfernt ist.

Basierend auf der Abschätzung für die Gesamtverzögerung, kann für zwei beliebige Transaktionen bestimmt werden, ob diese in Echtzeitreihenfolge angeordnet sind oder ob keine Garantie für die Echtzeitreihenfolge gegeben werden kann.

Garantie der Echtzeitreihenfolge

Seien ts_1 und ts_2 die Echtzeit-Zeitstempel unterschiedlicher lokaler Uhren. Gilt $|ts_1 - ts_2| > \delta_{\mathrm{gmax}}$, kann garantiert werden, dass die Zeitstempelreihenfolge von ts_1 und ts_2 der Echtzeitreihenfolge entspricht. Gilt dies nicht, kann die Echtzeitreihenfolge für die beiden Zeitstempel nicht garantiert werden.

Beweis Die Korrektheit des Verfahrens zur Überprüfung der Echtzeitreihenfolge lässt sich anhand seiner Konstruktion zeigen. Die tatsächliche Sendeverzögerung zweier Teilnehmer s_x und s_y, die sich direkt synchronisieren, beträgt δ_{xy} und es gilt $\delta_{xy} > 0$. Die gesamte Verzögerung des Synchronisationsvorgangs $\delta_{xy} + \delta_{yx}$ wird durch $\delta_{\max}$ abgeschätzt, es gilt also:

$$\delta_{xy} + \delta_{yx} \leq \delta_{\max}$$

Daraus folgt, dass die tatsächliche Sendeverzögerung δ_{xy} kleiner als die abgeschätzte Verzögerung ist, also:

$$\delta_{xy} < \delta_{\max}$$

Für die Anzahl der Synchronisationsschritte m zwischen der Referenzuhr und einem Teilnehmer der Replikationsgruppe gilt $m \leq n$ (siehe oben). Ebenfalls gilt, dass jede einzelne Sendeverzögerung δ_{xy} kleiner als δ_{max} ist. Schätzt man die einzelnen Sendeverzögerungen jeweils mit $\delta_{\max}$ ab, so gilt für die Gesamtverzögerung:

$$m \cdot \delta_{\max} \leq n \cdot \delta_{\max} = \delta_{\mathrm{gmax}}$$

□

Aufgrund der groben Abschätzung ergeben sich bei mittelgroßen Replikationsgruppen für δ_{gmax} Werte, die bis in den Sekundenbereich hineinreichen können. Dies stellt jedoch kein Problem dar, da die Reihenfolge von für wenige Sekunden auseinander liegenden Transaktionen aus Benutzersicht nicht wahrgenommen wird.

Integration in den Replikationsablauf

Das Uhrensynchronisationsverfahren wird während der Initialisierung der Replikation durchgeführt. Deshalb wird im Normalfall davon ausgegangen, dass die Replikationsteilnehmer erst Transaktionen durchführen (und damit Zeitstempel anfordern), nachdem die Uhrensynchronisation durchgeführt wurde. Trotzdem können auch schon während der Uhrensynchronisation Transaktionen von den Teilnehmern initiiert werden. Diese werden zwar lokal ausgeführt, die Replikation wird jedoch so lange verzögert, bis die Uhrensynchronisation durchgeführt wurde. Die zuvor vergebenen Zeitstempel der Transaktionen werden dann entsprechend dem Versatz der synchronisierten Uhr angepasst. Treten der Replikationsgruppe neue Teilnehmer bei, so wird die Synchronisation im Rahmen des Datenaustausches mit einem Teilnehmer der Replikationsgruppe durchgeführt.

Totale Ordnung der Zeitstempel

In der einleitenden Problemstellung des Kapitels wurde ein Beispiel gezeigt, bei dem die Ordnung der Transaktionen zwar nicht der Echtzeitreihenfolge entspricht, jedoch trotzdem bei allen Teilnehmern die gleiche Ordnung eingehalten wird. Wie in Kapitel 2 bereits beschrieben, bezeichnet man dies als eine totale Ordnung. Für dieses Ordnungskriterium ist nicht relevant, ob der Zeitstempel der Echtzeit entspricht oder nicht. So können die Uhren der einzelnen Teilnehmer zueinander sogar einen beliebig großen Versatz haben, ohne dass dies eine Auswirkung auf die totale Ordnung hat. Entscheidend ist nur, dass sich die Echtzeit-Zeitstempel nach dem Synchronisationsvorgang nicht weiter verändern oder angepasst werden.

Ebenso unproblematisch ist, dass aufgrund der beschriebenen Ungenauigkeiten die totale Ordnung der Zeitstempel im Gegensatz zur Kausalordnung stehen kann. So kann z. B. eine Transaktion TA_2, die kausal nach einer anderen Transaktion TA_1 stattgefunden hat, laut der Zeitstempelreihenfolge vor TA_1 ausgeführt worden sein. Die Zeitstempel werden jedoch nur berücksichtigt, wenn die Transaktionen kausal parallel stattgefunden haben und ein Konflikt besteht, ansonsten gilt primär die kausale Ordnung der Transaktionen (siehe Abschnitt 2.3.4).

6.4 Zusammenfassung

In diesem Kapitel haben wir ein Verfahren für die Uhrensynchronisation in MANETs vorgestellt. Dieses Verfahren verwendet ein leichtgewichtiges Synchronisationsprotokoll, das zwei direkt verbundene Teilnehmer synchronisiert. Durch die wiederholte Ausführung des Protokolls im MANET werden schrittweise alle Teilnehmer der Replikationsgruppe synchronisiert. Zusätzlich schätzt das Synchronisationsprotokoll die Sendeverzögerung der Uhrensynchronisation ab. Dadurch ist es möglich für zwei beliebige Zeitstempel unterschiedlicher Teilnehmer zu bestimmen, ob deren Echtzeitreihenfolge garantiert werden kann.

Basierend auf den synchronisierten Uhren der Teilnehmer können vom Replikationssystem Echtzeit-Zeitstempel vergeben werden. Diese Zeitstempel werden von den im Folgenden beschriebenen Replikationsmethoden verwendet. Besonders bei der Konfliktlösung stellen Echtzeit-Zeitstempel einen für die Benutzer transparenten und intuitiven Lösungsansatz dar.

Kapitel 7

Konflikterkennung und Konfliktlösung

Bei der Verwendung von optimistischer Replikation können aufgrund kausal parallel ausgeführter Transaktionen Konflikte entstehen (siehe Konfliktdefinition in Abschnitt 2.4.1). Um einen konsistenten Zustand innerhalb der Replikationsgruppe wiederherzustellen, ist eine einheitliche Erkennung und Lösung solcher Konflikte notwendig.

In diesem Kapitel stellen wir die von uns verwendeten Methoden zur dezentralen Konflikterkennung und -lösung vor.[1] Sie basieren darauf die kausalen Abhängigkeiten der Transaktionen zu nutzen. Jeder Replikationsmanager verwaltet diese lokal mit Hilfe einer Datenstruktur, dem *Vorgängergraphen*. Für jede neu hinzukommende Transaktion wird in die Datenstruktur stellvertretend ein Knoten mit den entsprechenden Abhängigkeiten eingefügt und geprüft, ob ein Konflikt entstanden ist. Im Falle eines Konflikts wird dieser, basierend auf den lokal bekannten Informationen, sofort gelöst. Dadurch wird die Datenbank zeitnah aktualisiert und für den Replikationsteilnehmer eine hohe Datenverfügbarkeit gewährleistet.

Das Kapitel gliedert sich wie folgt: Um ein Verständnis für das zugrunde liegende Problem zu schaffen und die Verwendung der Datenstruktur zu motivieren, wird zunächst der Ablauf unseres Replikationsansatzes erläutert. Darauf folgend wird in Abschnitt 7.2 der Vorgängergraph definiert und dessen Funktionsweise erläutert. Anschließend wird der Konfliktlösungsalgorithmus vorgestellt und seine Korrektheit nachgewiesen. Das von uns verwendete Commitverfahren wird in Abschnitt 7.4 beschrieben.

7.1 Ablauf der Replikation

Der zentrale Ablauf unseres Replikationsansatzes ist in Abbildung 7.1 dargestellt. Im Folgenden erklären wir die einzelnen Schritte des Ablaufs im Hinblick auf die von uns verwendeten Replikationsmethoden. Wir verwenden dabei die bereits in Abschnitt 2.4.1 eingeführten Definitionen des Systemmodells. Zusätzlich wird die kausale Abhängigkeit von Transaktionen definiert:

[1] Die Entwicklung der Methoden wurde in Zusammenarbeit mit Frank Bregulla im Rahmen seiner Diplomarbeit durchgeführt [Bre06].

Kausale Abhängigkeit Eine Transaktion TA_i ist von einer anderen Transaktion TA_j kausal abhängig, falls folgendes gilt:

- TA_j wurde kausal vor TA_i ausgeführt, d. h. TA_j wurde bei einem Replikationsteilnehmer lokal vor TA_i ausgeführt oder der Teilnehmer hat TA_j vor der Ausführung von TA_i erhalten (siehe Abschnitt 2.3.4).
- TA_i und TA_j greifen auf die gleichen Daten zu, d.h. es gilt: $(\mathcal{R}(TA_i) \cup \mathcal{W}(TA_i)) \cap (\mathcal{R}(TA_j) \cup \mathcal{W}(TA_j)) \neq \emptyset$.

Die Abhängigkeiten werden als Tupel beschrieben und sind durch folgende Relation gegeben:

$$dep \ : \ TA_x \mapsto \{(TA_x, TA_m), \ldots, (TA_x, TA_n)\}$$

Die Menge aller Abhängigkeiten einer bestimmten Transaktion TA_x ist durch $dep(TA_x)$ gegeben.

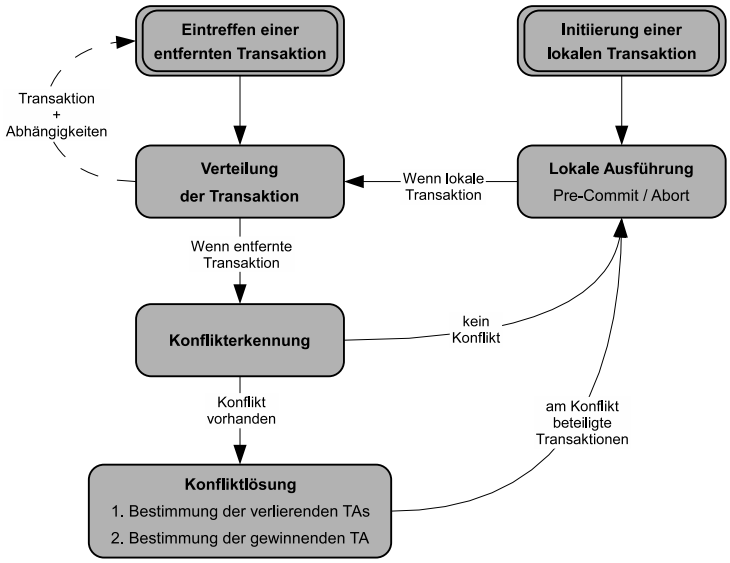


Abbildung 7.1: Ablauf der Replikation

Initialisierung Bei der Initialisierung des Systems wird sichergestellt[2], dass alle Replikationsteilnehmer das gleiche Schema und den gleichen Datenbestand aufweisen. Ebenso besitzen die Replikationsmanager der Teilnehmer eindeutige Kennungen (siehe S. 27) und kennen alle anderen Teilnehmer der Replikationsgruppe (siehe Abschnitt 2.4.4 und 2.4.5).

[2]Das Verfahren zum Abgleich der Datenschemata ist in der Arbeit von Sebastian Martens [Mar07] beschrieben.

Lokale Ausführung Wird eine lokale Transaktion von einem Teilnehmer initiiert, so wird sie zuerst vorläufig auf der lokalen Datenbank (siehe S. 28) ausgeführt. Erst *nach* der lokalen Ausführung der Transaktion durch ein pre-Commit (siehe S. 28) wird die Transaktion an die anderen Replikationsteilnehmer verteilt. Durch diese Vorgehensweise werden Transaktionen immer atomar ausgeführt und verteilt, d.h., es kann *keine Überschneidung von lokalen und entfernten Transaktionen* entstehen. Sowohl entfernte als auch lokale Transaktionen werden auf der lokalen Datenbank isoliert ausgeführt. Wenn zum Beispiel eine entfernte Transaktion von einem anderen Teilnehmer eintrifft während eine lokale Transaktion ausgeführt wird, so wird die Ausführung der entfernten Transaktion entsprechend nach hinten verschoben. Dadurch kann bei der Ausführung lokaler Transaktionen kein Konflikt entstehen. Zusätzlich zu den Transaktionen werden auch deren kausale Abhängigkeiten festgehalten und übertragen.

Verteilung Eine Transaktion wird nach ihrer lokalen Ausführung atomar an die anderen Teilnehmer verteilt. Damit ein Konflikt festgestellt werden kann, ist es notwendig, dass zu jeder Transaktion TA ebenfalls die lokal bekannten Abhängigkeiten $dep(TA)$ verteilt werden. Die Verteilung geschieht asynchron mit Hilfe der in Kapitel 5 beschriebenen Broadcastprotokolle.

Konflikterkennung / -lösung Wenn eine entfernte Transaktion bei einem Teilnehmer eintrifft, wird zunächst mit Hilfe der Abhängigkeiten (siehe Abschnitt 7.2) überprüft, ob die Transaktion mit anderen Transaktionen in Konflikt steht. Liegt ein Konflikt vor, so werden alle beteiligten Transaktionen entsprechend ihrem Ausführungszeitstempel (pre-Commit Zeitstempel) geordnet. Die Transaktion mit dem ältesten Zeitstempel gewinnt den Konflikt und die anderen Transaktionen werden abgebrochen. Die entsprechenden Änderungen werden anschließend lokal ausgeführt. Aufgrund der globalen Eindeutigkeit der Echtzeit-Zeitstempel ist eine identische Konfliktlösung bei allen Replikationsteilnehmern gewährleistet. Analog kann auch die Transaktion mit dem jüngsten Zeitstempel gewinnen oder Prioritätswerte verwendet werden, dies ändert jedoch nichts an der grundsätzlichen Vorgehensweise. Wenn alle Transaktionen an alle Replikationsteilnehmer verteilt sind, führt das zu einem identischen Datenbankzustand bei allen Teilnehmern. Dadurch kann Eventual Consistency gewährleistet werden. Aufgrund der asynchronen Verteilung kann es trotzdem vorkommen, dass sich Zwischenstände der einzelnen Teilnehmer unterscheiden.

Ablauf des Commitverfahrens

Das Commitverfahren hat das Ziel, einen stabilen globalen Zustand herzustellen. Dieser Zustand wird mit Hilfe einer Sicht im System festgehalten (siehe S. 28). Das Verfahren besitzt einen eigenen Ablauf, der unabhängig vom zentralen Replikationsablauf ist.

Zunächst wird ein Commitpunkt (siehe S. 29) gesetzt, um global einheitlich zu markieren, welche Transaktionen festgeschrieben werden sollen. Der Commitpunkt kann von einem beliebigen

Teilnehmer oder durch einen vorher festgelegten Zeitplan vorgegeben werden. Wenn ein Replikationsteilnehmer alle Transaktionen bis zum Commitpunkt erhalten hat, kann er die Transaktionen davor endgültig festschreiben bzw. abbrechen und die Historie entsprechend kürzen. Um sicherzustellen, dass er alle Transaktionen erhalten hat, muss er zunächst von den anderen Teilnehmern jeweils über ein lokales Ereignis informiert werden, das nach dem Commitpunkt stattgefunden hat. Dies können weitere Transaktionen aber auch sogenannte Statusnachrichten sein, die signalisieren, dass ein Teilnehmer noch an der Replikation beteiligt ist, jedoch zu diesem Zeitpunkt keine Transaktionen ausgeführt hat. Zusätzlich muss mit Hilfe der Sequenznummern (siehe S. 28) überprüft werden, ob Transaktionen eines Teilnehmers ausgelassen wurden.

7.1.1 Beispielablauf

Zum besseren Verständnis wird in Abbildung 7.2 ein Beispielablauf mit der Verwendung von Echtzeit-Zeitstempeln gezeigt. Das Beispiel zeigt eine Replikationsgruppe mit den drei Teilnehmern s_1, s_2 und s_3. Zu Beginn werden von den Teilnehmern s_1 und s_2 die Transaktionen TA_{17} und TA_{18} nebenläufig ausgeführt. Beide Transaktionen bestehen aus einer einzigen Schreiboperation auf dem Datenelement e und einer lokalen pre-Commit Operation. Die Transaktion TA_{17} besitzt einen kleineren Zeitstempel und ist somit älter als TA_{18}. Nachdem beide Transaktionen lokal ausgeführt wurden, werden sie an die anderen Teilnehmer verteilt. Bei allen Teilnehmern wird der Konflikt erkannt und daraufhin werden beide Transaktionen entsprechend ihrer Zeitstempel geordnet. Da TA_{18} den jüngeren Zeitstempel besitzt, wird TA_{18} abgebrochen (abort).

Danach setzt s_2 einen globalen Commitpunkt cp (13:05) und verteilt diesen an alle anderen Teilnehmer. Um TA_{17} endgültig festzuschreiben, müssen die Teilnehmer jeweils über ein neues Ereignis der anderen Teilnehmer informiert werden, das nach cp stattgefunden hat.

Ist ein Teilnehmer längere Zeit nicht erreichbar, wird die Commit Entscheidung entsprechend verzögert (siehe Abschnitt 2.4.5). Nachdem die Teilnehmer die Statusnachrichten empfangen haben, können alle abschließend TA_{17} endgültig festschreiben.

Damit ein Replikationssystem die kausal parallele Ausführung von Transaktionen mit Hilfe der gegebenen Abhängigkeiten feststellen kann, wird eine Datenstruktur benötigt. Im folgenden Abschnitt stellen wir die von uns verwendete Datenstruktur, den *Vorgängergraphen*, vor.

7.2 Die Vorgängergraph Datenstruktur

Der Vorgängergraph ist eine Datenstruktur, die von unserem Replikationsansatz für die dezentrale Konflikterkennung und -lösung verwendet wird. Der Vorgängergraph ist eine Erweiterung der Vorgängerliste [KRSD01, FLS01] (siehe Abschnitt 2.3.5) und repräsentiert die Transaktionen und

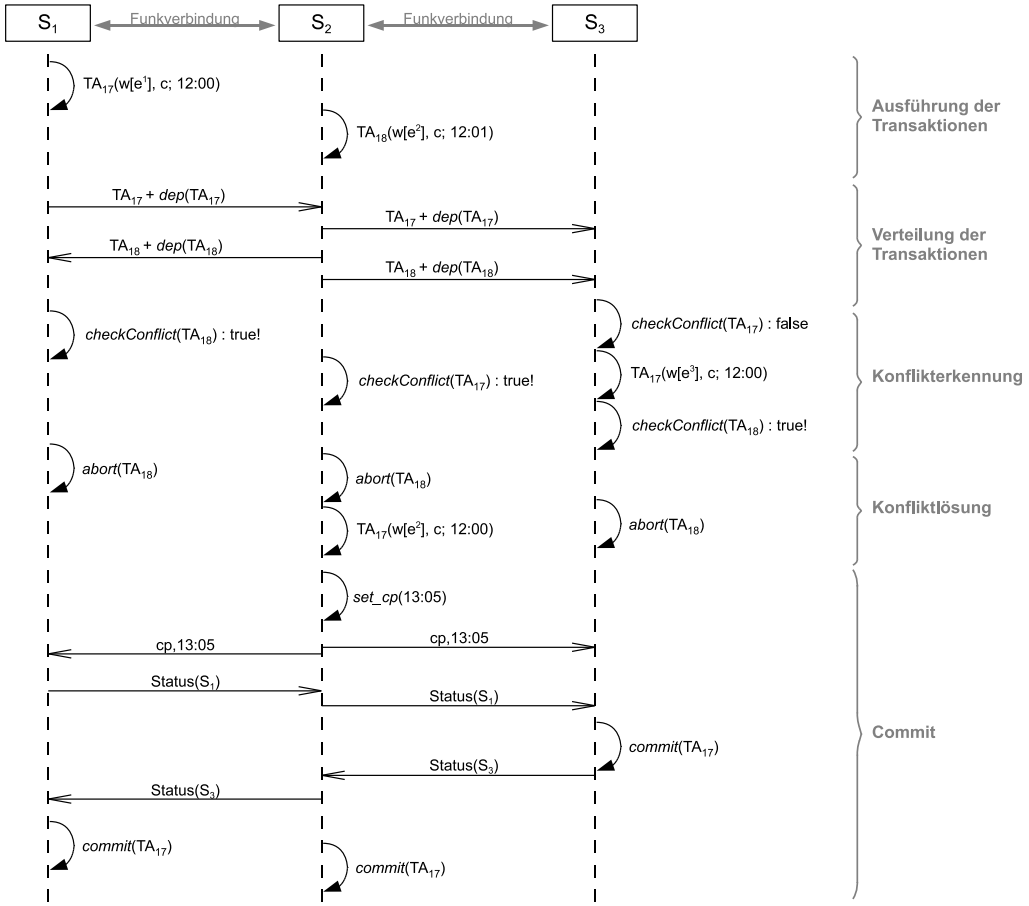


Abbildung 7.2: Ablauf der Replikation (alle dargestellten Datenübertragungen sind asynchron)

deren Abhängigkeiten (dep), die einem Replikationsmanager lokal zu einem bestimmten Zeitpunkt bekannt sind. Im Gegensatz zu anderen Replikationssystemen (siehe Kapitel 3) können mit Hilfe des Vorgängergraphen nicht nur Operationen, sondern auch Transaktionen verwaltet werden.

Als Voraussetzung für die Definition eines Vorgängergraphen müssen zunächst die notwendigen Begriffe eingeführt werden. Damit Konflikte nicht nur auf Datenelementen (siehe Abschnitt 2.4.1) erkannt werden, sondern auch eine zeilenweise oder tabellenweise Konflikterkennung möglich ist, werden sogenannte Konfliktelemente verwendet. Bei der Entwicklung von verteilten Applikationen ist es teilweise notwendig, für die Konflikterkennung eine größere Granularität zu verwenden.

Konfliktelement An Stelle der Datenelemente werden bei der Konflikterkennung und -lösung Konfliktelemente verwendet. Die Menge aller Konfliktelemente wird als K bezeichnet. Ein Konfliktelement $k \in K$ ist die kleinste Einheit der Konflikterkennung und repräsentiert eine nicht leere Menge von Datenelementen. Im einfachsten Fall entspricht ein Konfliktelement genau

einem Datenelement.

Nachfolgend werden kommutative Operationen und Transaktionen erläutert. Durch die Berücksichtigung von kommutativen Transaktionen können Transaktionsabbrüche vermieden werden. Hat z. B. kausal parallel jeweils ein Inkrement auf einem Konfliktelement stattgefunden, so können die in Konflikt stehenden Transaktionen beide erhalten bleiben.

Die kausal parallelen Transaktionen werden sequentiell ausgeführt. Die Reihenfolge ist aufgrund der Kommutativität beliebig.

Kommutativitätsgruppe Mit Hilfe von Kommutativitätsgruppen wird die Kommutativität von Schreiboperationen festgelegt. Eine Kommutativitätsgruppe C_x ist eine Menge von Schreiboperationen $\{w_1^x, ..., w_n^x\}$ der Form $w(e \leftarrow f(D))$. Die Funktion f verwendet die Elemente der Datenbank D und enthält nur Operatoren (z. B. +) und Leseoperationen (z. B. $r(x)$). Die Ausführungsreihenfolge der Operationen einer Kommutativitätsgruppe hat keinen Einfluss auf das Ergebnis, d. h. es gilt: $\forall\ (w_i^x, w_j^x) \in C_x \times C_x : w_i^x(w_j^x) = w_j^x(w_i^x)$. Inkrementoperationen können z. B. durch eine Kommutativitätsgruppe mit Schreiboperationen der Form $w(e \leftarrow r(e) + k)$ repräsentiert werden.

Kommutative Schreiboperationen Zwei Schreiboperationen sind kommutativ zueinander, wenn sie derselben Kommutativitätsgruppe angehören. Schreiboperationen der Kommutativitätsgruppe C_0 bilden eine Ausnahme. Die Gruppe C_0 ist definiert als Menge, die alle Schreiboperationen enthält, die zu keiner anderen Schreiboperation kommutativ sind. Blinde Schreiboperationen (engl. blind write) fallen typischerweise in diese Gruppe, da sie auf einem Element schreiben ohne es vorher zu lesen.

Kommutative Transaktionen Die Kommutativität von Transaktionen beruht auf der Kommutativität ihrer Schreiboperationen. Basierend auf [JM93, JMR97] werden Transaktionen als kommutativ zueinander bezeichnet, wenn alle Operationen der Transaktionen, die auf gemeinsamen Konfliktelementen ausgeführt werden, kommutativ zueinander sind.

Ein Sonderfall besteht, wenn eine Transaktion zwei oder mehr Operationen enthält, die auf dem gleichen Konfliktelement arbeiten und nicht kommutativ zueinander sind. Da die Kommutativitätsgruppen nur für einzelne Operationen angegeben werden, kann keine Aussage über die Kommutativität der Ausführung einer solchen Operationsfolge getroffen werden. Aus diesem Grund wird eine nicht kommutative Operationsfolge innerhalb einer Transaktion für einen Vergleich mit den Operationen anderer Transaktionen als nicht kommutativ zu allen anderen Operationen bewertet.

Nachdem die grundlegenden Begriffe eingeführt wurden, betrachten wir nun die zentrale Datenstruktur unseres Replikationsansatzes. Diese stellt eine Repräsentation der Transaktionsabhängig-

keiten dar und ermöglicht somit eine Konflikterkennung. Jeder Replikationsmanager verwendet einen Vorgängergraphen und verwaltet damit die Beziehungen der Transaktion untereinander.

Vorgängergraph Ein Vorgängergraph G_d ist ein gerichteter azyklischer Graph, der explizit die Abhängigkeiten aller Transaktionen darstellt, die einem Teilnehmer s_d zu einem bestimmten Zeitpunkt bekannt sind. Der Graph $G_d = (V, E)$ besteht aus einer Menge von Knoten $V = \{t_1, ..., t_n\}$ und einer Menge gerichteter Kanten $(t_i, t_j) \in E$.

Jeder **Knoten** t_i repräsentiert eine Transaktion, die dem Teilnehmer s_d bekannt ist. Jede **Kante** (t_i, t_j) repräsentiert eine kausale Abhängigkeit (TA_i, TA_j) und verweist auf den Knoten, der die in der Kausalreihenfolge jeweils vorangegangene Transaktion repräsentiert. Jeder Kante (t_i, t_j) ist durch die Funktion $on : E \rightarrow \mathcal{P}(K \times \{r, w^x\})\backslash\emptyset$ die Menge der Konfliktelemente zugeordnet für die die entsprechende Abhängigkeit besteht. Ebenso wird vermerkt, ob der Zugriff der Transaktion t_i lesend (r) oder schreibend (w^x) ist. Wird innerhalb einer Transaktion ein Konfliktelement sowohl gelesen als auch geschrieben, so wird nur der Schreibzugriff vermerkt.

Die Knoten des Graphen, die keine ausgehenden Kanten besitzen, werden als **Wurzelknoten** bezeichnet. Die Wurzelknoten beinhalten leere Transaktionen, die stellvertretend für jedes Konfliktelement vorhanden sind. Die Knoten des Graphen, die keine eingehenden Kanten besitzen, werden als **Blattknoten** bezeichnet.

In Abbildung 7.3 sind zwei Beispiele eines Vorgängergraphen gegeben. In Abbildung 7.3(a) ist ein Graph dargestellt, der keine kommutativen Transaktionen enthält, da alle Schreiboperationen aus der Kommutativitätsgruppe C_0 stammen. Die Wurzelknoten des Graphen sind grau eingefärbt und den Konfliktelementen k_1, k_2 und k_3 zugeordnet. Es folgen die Knoten t_1 und t_2, die schreibend auf die Konfliktelemente zugreifen.[3] Während t_1 nur auf ein Konfliktelement zugreift, wird von t_2 sowohl k_2 als auch k_3 geschrieben. Die Knoten t_3, t_4 und t_5 greifen auf die Ergebnisse der ersten beiden Knoten zu und referenzieren diese deshalb. Knoten t_5 greift auf beide Konfliktelemente zu, die auch von t_2 geschrieben wurden, aus diesem Grund enthält die Kante von t_5 nach t_2 sowohl k_2 als auch k_3. Zuletzt greifen drei weitere Knoten auf die Ergebnisse von t_4 zu.

In Abbildung 7.3(b) ist ein Graph gezeigt, der auch kommutative Knoten enthält. Die Knoten t_1, t_2 und t_3 greifen auf das Konfliktelement k_1 zu und gehören alle der gleichen Kommutativitätsgruppe an. Die Transaktion des Knotens t_4 hat kausal nach den anderen Transaktionen stattgefunden. Aus diesem Grund referenziert t_4 alle anderen drei Knoten als Vorgänger. Seine Schreiboperation gehört der Kommutativitätsgruppe C_2 an. Abschließend führt t_5 eine Schreiboperation der gleichen Gruppe auf k_1 aus. Es folgen nun die Erläuterungen zu den Beziehungen, die zwischen den Knoten eines Vorgängergraphen bestehen.

[3] Zur Verbesserung der Lesbarkeit wird ein Knoten des Vorgängergraphen stellvertretend für die von ihm repräsentierte Transaktion genannt. Die Begriffe werden im Folgenden synonym verwendet.

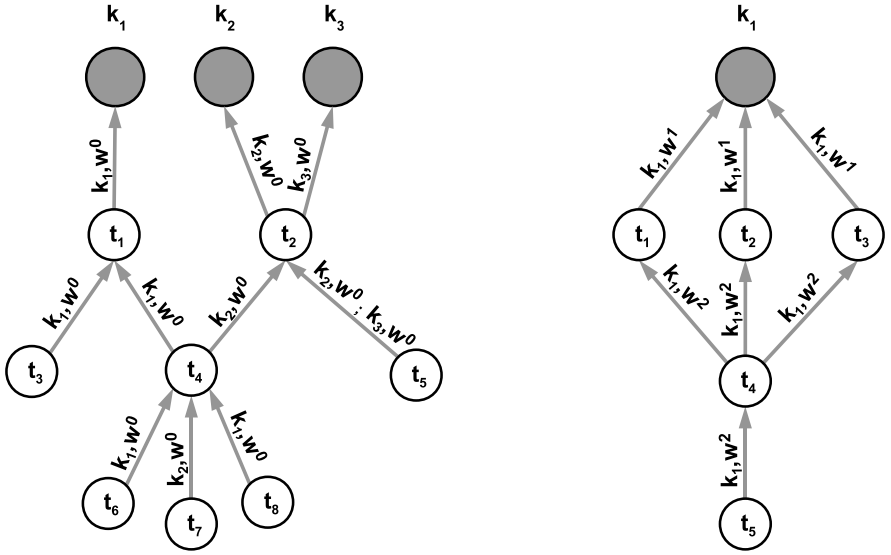


Abbildung 7.3: Vorgängergraphen mit kommutativen und nicht-kommutativen Schreiboperationen. Die Konfliktelemente werden mit k_y, die Schreiboperationen einer Kommutativitätsgruppe C_x mit w^x bezeichnet.

7.2.1 Funktionsdefinitionen

Elternknoten Die ausgehenden Kanten eines Knotens $t_i \in V$ verweisen auf die Eltern des Knotens in Bezug auf ein Konfliktelement. Betrachtet man keine kommutativen Transaktionen, so besitzt jede Transaktion pro Konfliktelement maximal einen Elternknoten. Bei der Verwendung von kommutativen Transaktionen sind auch mehrere Elternknoten möglich. Die Elternknoten sind durch die Funktion $parents : V \times K \rightarrow \mathcal{P}(V)$ gegeben. Diese liefert die Elternknoten in Bezug auf ein Konfliktelement und ist wie folgt definiert:

$$parents(t_i, k) := \{t_j \in V | \exists (t_i, t_j) \in E \wedge ((k, w^x) \in on(t_i, t_j) \vee (k, r) \ \in \ on(t_i, t_j))\}$$

Die Menge aller Elternknoten eines Knotens ist durch die Funktion $allparents : V \rightarrow \mathcal{P}(V)$ gegeben. Diese ist für einen Knoten t_i wie folgt definiert:

$$allparents(t_i) := \{t_j \in V \mid \exists (t_i, t_j) \in E\}$$

Vorgängerknoten Für die Erkennung von Konflikten ist die Menge der Vorgängerknoten eines oder mehrerer Knoten relevant. Für eine Knotenmenge $T \subseteq V$ ist die Menge der Vorgängerknoten in Bezug auf ein Konfliktelement k durch die Funktion $pred : \mathcal{P}(V) \times K \rightarrow \mathcal{P}(V)$ wie folgt

definiert:

$$pred(T,k) := T \cup \bigcup_{t \in T} pred(parents(t,k))$$

Die Menge aller Vorgängerknoten ist durch die Funktion $allpred : \mathcal{P}(V) \rightarrow \mathcal{P}(V)$ definiert:

$$allpred(T) := T \cup \bigcup_{t \in T} allpred(allparents(t))$$

Aus technischen Gründen ist jeder Knoten auch als ein Vorgänger von sich selbst definiert.

Kindknoten Die eingehenden Kanten eines Knotens sind die Verweise der Kindknoten. Die Kindknoten eines Knotens $t_j \in V$ für ein bestimmtes Konfliktelement k sind durch die Funktion $children : V \times K \rightarrow \mathcal{P}(V)$ gegeben. Die Funktion ist wie folgt definiert:

$$children(t_j,k) := \{t_i \in V | (t_i,t_j) \in E \wedge ((k,w^x) \in on(t_i,t_j) \vee (k,r) \in on(t_i,t_j))\}$$

Die Menge aller Kindknoten eines Knotens ist durch die Funktion $allchildren : V \rightarrow \mathcal{P}(V)$ gegeben. Diese ist für einen Knoten t_j wie folgt definiert:

$$allchildren(t_j) := \{t_i \in V | (t_i,t_j) \in E\}$$

Nachfolgerknoten Für eine Knotenmenge $T \subseteq V$ ist die Menge der Nachfolgerknoten in Bezug auf ein Konfliktelement k durch die Funktion $succ : \mathcal{P}(V) \times K \rightarrow \mathcal{P}(V)$ wie folgt definiert:

$$succ(T,k) := T \cup \bigcup_{t \in T} succ(children(t,k))$$

Die Menge aller Nachfolgerknoten ist durch die Funktion $allsucc : \mathcal{P}(V) \rightarrow \mathcal{P}(V)$ definiert:

$$allsucc(T) := T \cup \bigcup_{t \in T} allsucc(allchildren(t))$$

Aus technischen Gründen ist jeder Knoten auch als ein Nachfolger von sich selbst definiert.

Geschwisterknoten Die Geschwisterknoten eines Knotens t sind Knoten, die die gleichen Elternknoten für dasselbe Konfliktelement besitzen. Die Geschwisterknoten eines Knotens t in Bezug auf ein Konfliktelement k werden durch die Funktion $siblings : V \times K \rightarrow \mathcal{P}(V)$ bestimmt. Die Funktion ist wie folgt definiert:

$$siblings(t,k) := \bigcup_{t' \in parents(t,k)} children(t',k) \setminus t$$

Verwandte Knoten Die verwandten Knoten eines Knotens t sind die Knoten, die einen Vorgänger haben, der die gleichen Elternknoten wie ein Vorgänger des Knotens t für dasselbe Konfliktelement k besitzt. Die verwandten Knoten eines Knotens t werden durch die Funktion $relatives : V \rightarrow \mathcal{P}(V)$ definiert:

$$relatives(t,k) := \bigcup_{t' \in pred(t,k)} succ(t',k) \setminus (pred(t,k) \cup succ(t,k))$$

Wendet man die Funktionen auf die Knoten des Graphen aus Abbildung 7.3(a) an, ergeben sich folgende Werte:

$$\begin{aligned}
&parents(t_4,k_2) = \{t_2\} \quad allparents(t_4) = \{t_1,t_2\}\\
&children(t_4,k_2) = \{t_7\} \quad allchildren(t_4) = \{t_6,t_7,t_8\}\\
&pred(\{t_6\},k_1) = \{t_1,t_4,t_6\} \quad allpred(\{t_6\}) = \{t_1,t_2,t_4,t_6\}\\
&succ(\{t_1\},k_1) = \{t_1,t_3,t_4,t_6,t_8\} \quad allsucc(\{t_1\}) = \{t_1,t_3,t_4,t_6,t_7,t_8\}\\
&siblings(t_3,k_1) = \{t_4\} \quad relatives(t_3,k_1) = \{t_4,t_6,t_8\}
\end{aligned}$$

Nachdem der Vorgängergraph und die darauf aufbauenden Funktionen definiert sind, wird im nächsten Abschnitt die Konflikterkennung im Vorgängergraph besprochen.

7.2.2 Konflikte im Vorgängergraph

Wie bereits im Systemmodell erläutert (siehe Abschnitt 2.4), liegt zwischen zwei Transaktionen ein Konflikt vor, wenn diese kausal parallel ausgeführt werden und beide Transaktionen mindestens auf ein gleiches Datenelement zugreifen, wobei mindestens ein Zugriff davon schreibend ist.

Anhand eines Vorgängergraphen können solche Konflikte erfasst werden. Dabei werden zwei orthogonale Kriterien – die Art und die Kommutativität von Konflikten – unterschieden, die jeweils zwei Ausprägungen besitzen. Das erste Kriterium (Konfliktart) unterscheidet zwischen *direkten* und *indirekten* Konflikten. Das zweite Kriterium (Kommutativität) unterscheidet zwischen *kommutativen* und *nicht-kommutativen* Konflikten.

Direkter Konflikt Zwischen den Knoten (und somit auch zwischen deren Transaktionen) t_i und t_j besteht ein direkter Konflikt $t_i \leftrightarrow t_j$, wenn auf mindestens ein gemeinsames Konfliktelement k ein Schreibzugriff erfolgt und die beiden Knoten in Bezug auf dasselbe Element Geschwister sind:

$$t_i \in siblings(t_j,k) \wedge \Big(\exists t_m : \exists(k,w^x) \in on(t_i,t_m) \vee \exists(k,w^x) \in on(t_j,t_m)\Big)$$

Indirekter Konflikt Zwischen den Knoten t_i und t_j besteht ein indirekter Konflikt $t_i \tilde{\leftrightarrow} t_j$, wenn auf mindestens ein gemeinsames Konfliktelement k ein Schreibzugriff erfolgt und die beiden Knoten in Bezug auf dasselbe Element verwandt sind:

$$t_i \in relatives(t_j, k) \wedge \Big(\exists t_m : \exists (k, w^x) \in on(t_i, t_m) \vee \exists t_n : \exists (k, w^x) \in on(t_j, t_n) \Big)$$

Kommutativer Konflikt Sei K_t die Menge der Konfliktelemente, auf die die Transaktion t zugegriffen hat. Sei $W_{t,k}$ die Menge aller Schreiboperationen, die von Transaktion t auf Konfliktelement k durchgeführt wurden. Zwischen den Knoten t_i und t_j besteht ein kommutativer Konflikt $t_i \overset{+}{\longleftrightarrow} t_j$, wenn ein direkter oder indirekter Konflikt besteht und alle am Konflikt beteiligten Paare von Schreiboperationen kommutativ zueinander sind:

$$(t_i \leftrightarrow t_j \vee t_i \tilde{\leftrightarrow} t_j) \wedge \Big(\forall k \ \in \ K_{t_i} \cap K_{t_j} : \forall (w_i^x, w_j^y) \in (W_{t_i,k} \times W_{t_j,k}) : x = y \neq 0 \Big)$$

Nicht-kommutativer Konflikt Zwischen den Knoten t_i und t_j besteht ein nicht-kommutativer Konflikt $t_i \overset{-}{\longleftrightarrow} t_j$, wenn ein direkter oder indirekter Konflikt besteht und mindestens ein am Konflikt beteiligtes Paar von Schreiboperationen nicht kommutativ zueinander ist:

$$(t_i \leftrightarrow t_j \vee t_i \tilde{\leftrightarrow} t_j) \wedge \Big(\exists k \ \in \ K_{t_i} \cap K_{t_j} : \exists (w_i^x, w_j^y) \in (W_{t_i,k} \times W_{t_j,k}) : x \neq y \vee x = 0 \Big)$$

Für alle beschriebenen Ausprägungen von Konflikten gilt die Symmetrie, d. h. gilt $t_i \leftrightarrow t_j$, so gilt auch $t_j \leftrightarrow t_i$ (analog für die anderen Ausprägungen).

Konfliktgruppe Eine Menge von Transaktionen, die direkt oder indirekt miteinander in Konflikt stehen, wird als Konfliktgruppe bezeichnet.

Betrachtet man erneut den Graphen aus Abbildung 7.3(a), kann man mehrere Konflikte erkennen. Die Knoten t_3 und t_4 stehen in direktem Konflikt zueinander, ebenso t_4 und t_5. Die Transaktionen der Knoten t_3 und t_5 wurden zwar nebenläufig ausgeführt, arbeiten jedoch nicht auf dem gleichen Konfliktelement und somit besteht kein Konflikt. Ein indirekter Konflikt besteht zwischen den Knoten t_6 und t_3; t_8 und t_3; ebenso zwischen t_7 und t_5. Es ergeben sich dadurch die zwei Konfliktgruppen $\{t_3, t_4, t_6, t_8\}$ und $\{t_4, t_5, t_7\}$. Alle Konflikte sind nicht-kommutativ, da in diesem Beispiel keine kommutativen Schreiboperationen verwendet werden.

Der Graph aus Abbildung 7.3(b) enthält kommutative Konflikte. Da die Schreiboperationen der Knoten t_1, t_2 und t_3 alle aus derselben Kommutativitätsgruppe sind, besteht ein direkter kommutativer Konflikt zwischen diesen drei Knoten.

Relationen Mit Hilfe der Echtzeit-Zeitstempel der Knoten und den zuvor definierten Bedingungen können zwischen den Knoten des Vorgängergraphen folgende Relationen gebildet werden. Diese sind analog zu den Relationen, die zwischen den Transaktionen bestehen.

- **Zeitliche Ordnung:** $t_i < t_j$, wenn der Zeitstempel von t_i älter als der von t_j ist.
- **Kausale Abhängigkeit:** t_i ist abhängig von t_j ($t_i \rightarrow t_j$), wenn t_i auf Konfliktelemente zugreift, auf die bereits von t_j zugegriffen wurde. Dies ist gleichbedeutend mit $\exists k : t_j \in pred(t_i, k)$.
- **Konflikt:** t_i steht mit t_j in Konflikt, wenn ein direkter $t_i \leftrightarrow t_j$ oder indirekter Konflikt $t_i \tilde{\leftrightarrow} t_j$ besteht. Ein Konflikt kann jeweils kommutativ $t_i \overset{+}{\longleftrightarrow} t_j$ oder nicht-kommutativ $t_i \overset{-}{\longleftrightarrow} t_j$ sein.

7.2.3 Einfügen von Transaktionen

Die Funktion $insertNode$ fügt einen neuen Knoten in den Vorgängergraphen ein. Als Eingabe erhält sie eine Transaktion. In Listing 1 ist der Pseudocode der Funktion dargestellt.

Listing 1 Einfügealgorithmus

```
1:  function INSERTNODE(ta : Transaction)
2:
3:      Set writeSet := ta.getWriteSet()
4:      Set readSet := ta.getReadSet()
5:      List conflictElements := getConflictElements(writeSet, readSet)
6:      List lastNodes := getLastNodes(conflictElements)
7:
8:      Node t := createNewNode(ta, lastNodes)
9:      updateLastNodes(t)
10:     updateConflictElements(t)
11:
12: end function
```

In den Zeilen 3 - 5 werden die Konfliktelemente bestimmt, die von der Transaktion betroffen sind. Anschließend werden die Knoten bestimmt, die zuletzt ebenfalls auf mindestens ein gleiches Konfliktelement zugegriffen haben. In Zeile 8 wird ein neuer Knoten mit den entsprechenden Kanten erstellt. Die Funktion $getLastNodes$ verwendet eine Map, die jeweils die Knoten enthält, die zuletzt auf ein Konfliktelement zugegriffen haben. Diese Map muss beim Einfügen eines neuen Knotens aktualisiert werden (Zeile 9). Ebenso verhält es sich beim Zuordnen der Konfliktelemente.

Alle verwendeten Funktionen haben im Erwartungsfall eine lineare Laufzeit in Bezug auf die Anzahl der Eingabeelemente, da die Maps als Hashtabellen implementiert sind.

7.2.4 Konflikterfassung im Vorgängergraphen

Betrachtet man die Vorgängergraphen im Kontext der Replikation, so können Transaktionen bei einem Replikationsteilnehmer nicht nur lokal eingefügt werden. Auch die Transaktionen anderer Replikationsteilnehmer werden zusammen mit ihren Abhängigkeiten verteilt und die jeweiligen Knoten werden dann in den lokalen Vorgängergraphen des empfangenden Teilnehmers eingefügt. Im Gegensatz zu dem oben beschriebenen Einfügen einer lokalen Transaktion sind die Transaktionen anderer Teilnehmer bereits in deren Vorgängergraphen eingefügt worden. Aus diesem Grund wurden für diese Transaktionen bereits Knoten mitsamt der entsprechenden Verweise auf die Elternknoten im Graphen erstellt. Dadurch ist es mit Hilfe des Vorgängergraphen möglich, kausale Parallelität festzustellen und somit Konflikte zu erfassen. Anhand eines Beispiels in Abbildung 7.4 soll dies verdeutlicht werden. Zur einfacheren Veranschaulichung werden in diesem Beispiel nur Transaktionen mit Schreiboperationen auf genau einem Konfliktelement betrachtet. Die Kantenbeschriftung wird deshalb weggelassen.

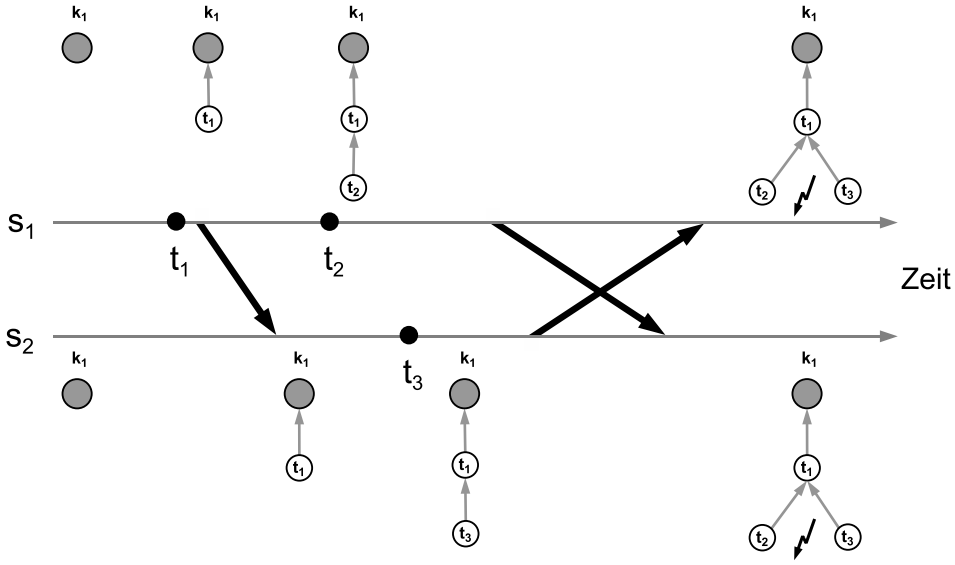


Abbildung 7.4: Erfassung eines Konflikts durch den Vorgängergraphen

Die Replikationsteilnehmer s_1 und s_2 besitzen beide das Konfliktelement k_1. Zu Beginn besteht der Vorgängergraph bei beiden Teilnehmern nur aus einem Wurzelknoten. Der Teilnehmer s_1 führt eine Transaktion aus, und der Knoten t_1 wird in den Vorgängergraph eingefügt. Anschließend wird der Knoten t_1 inklusive seiner ausgehenden Kante an s_2 weitergeschickt und in dessen Graphen eingefügt.

Beide Knoten führen danach kausal parallel die Transaktionen t_2 und t_3 lokal aus (siehe Replikationsablauf S. 93). Entsprechend werden jeweils lokal die Knoten t_2 und t_3 in den Vorgängergraphen

eingefügt. Zuletzt schicken sich beide Teilnehmer die neuen Knoten inklusive der ausgehenden Kanten zu, und diese werden in die Graphen eingefügt. Beim Einfügen tritt bei beiden Teilnehmern ein Konflikt zwischen den Knoten t_2 und t_3 auf. Wie dieser Konflikt gelöst werden kann, wird in Abschnitt 7.3 erklärt.

Bei dem Einfügen entfernter Transaktionen kann es vorkommen, dass neue einzufügende Knoten auf andere Knoten verweisen, die nicht vorhanden sind. Dies kann zum Einen daran liegen, dass die Nachrichten in vertauschter Reihenfolge angekommen sind. In diesen Fall wird das Einfügen des Knotens so lange lokal verzögert, bis der entsprechende Vorgängerknoten vom Teilnehmer empfangen wurde. Es ist ebenfalls möglich, dass ein Knoten bereits durch das Commitverfahren gelöscht wurde (siehe Abschnitt 7.4). Hierbei müssen die Verweise des Knoten entsprechend angepasst werden (siehe ebd.).

7.3 Konfliktlösung im Vorgängergraphen

In diesem Abschnitt wird erklärt, wie Konflikte mit Hilfe des Vorgängergraphen gelöst werden können. Zunächst definieren wir für den Vorgängergraphen Nebenbedingungen. Werden diese Bedingungen eingehalten, so stellt der Graph eine konfliktfreie Ausführungsreihenfolge von Transaktionen dar. Die Nebenbedingungen müssen somit auch nach der Ausführung des Konfliktlösungsalgorithmus gelten, den wir im Anschluss erläutern.

7.3.1 Nebenbedingungen

Ziel der Konfliktlösung ist es, für eine gegebene Menge von Transaktionen eine kausale Ausführungsreihenfolge zu ermitteln, die frei von nicht-kommutativen Konflikten ist. Im Falle eines Konflikts existiert für die entsprechenden Transaktionen keine eindeutige kausale Ausführungsreihenfolge mehr. Dadurch kann für die betroffenen Datenelemente der aktuelle Wert ebenfalls nicht mehr eindeutig bestimmt werden.

Kommutative Konflikte können bestehen bleiben, da ein eindeutiger Wert für das entsprechende Datenelement unabhängig von der Ausführungsreihenfolge der betroffenen Transaktionen bestimmt werden kann. Um die Konfliktfreiheit anhand des Graphen zu überprüfen, führen wir in diesem Abschnitt Nebenbedingungen für den Vorgängergraphen ein. Diese Nebenbedingungen müssen nach der Ausführung des Konfliktlösungsalgorithmus erfüllt sein. Dadurch wird eine eindeutige Ausführungsreihenfolge der Transaktionen sichergestellt und ein konsistenter Datenbankzustand erreicht.

Um mit Hilfe des Vorgängergraphen eine eindeutige Ausführungsreihenfolge darzustellen, müssen vorläufig ausgeführte und abgebrochene Transaktionen markiert werden. Ebenso müssen Transaktionen markiert werden, über deren vorläufige Ausführung noch nicht entschieden wurde. Aus diesem Grund besitzt jeder Knoten des Graphen einen Zustand, der angibt, ob die betreffende Transaktion bereits vorläufig ausgeführt (*aktiviert*), abgebrochen (*deaktiviert*) oder noch nicht entschieden

(*undefiniert*) wurde. Die Berücksichtigung dieser Zustände ergibt eine eindeutige Ausführungsreihenfolge.

Die folgenden Nebenbedingungen eines Vorgängergraphen gewährleisten, dass der Graph keine nicht-kommutativen Konflikte, sondern nur kommutative Konflikte enthält. Im Einzelnen sind die Nebenbedingungen eines Vorgängergraphen $G = (V, E)$ wie folgt:

1. Es existiert kein Paar von aktivierten Transaktionen $t_i, t_j \in V$, so dass zwischen t_i und t_j ein nicht-kommutativer Konflikt besteht.

2. Für jede aktivierte Transaktion $t_i \in V$ gilt, dass alle Transaktionen, von denen t_i abhängig ist, auch aktiviert sind.

3. Für jede aktivierte Transaktion t_i gilt, dass t_i von allen aktivierten Geschwisterknoten der Transaktionen, die Elternknoten von t_i sind, ebenfalls abhängig ist:

$$\forall t_i \in V, \forall k \in K_{t_i} : \forall t_p \in parents(t_i, k) : \forall t_s \in siblings(t_p, k) :$$
$$t_p \text{ ist aktiviert } \wedge\, t_s \text{ ist aktiviert} \Rightarrow t_s \in parents(t_i, k)$$

   Dieses Kriterium sorgt für die korrekte Konstellation kommutativer Konflikte, da Kriterium 1 nicht ausreichend ist. Durch diese Nebenbedingung wird sichergestellt, dass ein Knoten von allen an einem kommutativen Konflikt beteiligten Knoten abhängig ist.

4. Jede Transaktion $t_i \in V$ ist entweder aktiviert oder deaktiviert.

Die erste Nebenbedingung sorgt dafür, dass keine nicht-kommutativen Konflikte im Graphen bestehen bleiben. So darf z. B. im Graphen aus Abbildung 7.4 nur einer der beiden Knoten t_2, t_3 aktiviert sein, damit diese Nebenbedingung eingehalten wird. Die zweite Bedingung gewährleistet, dass eine Ausführungsreihenfolge nicht unterbrochen ist. So muss t_1 ebenfalls aktiviert sein, wenn t_3 aktiviert ist.

Die dritte Nebenbedingung stellt sicher, dass ein Knoten alle an einem kommutativen Konflikt beteiligten Knoten referenziert. Ein Beispiel dafür ist in Abbildung 7.5 gegeben: Der Knoten t_3 referenziert nur einen Teil der Knoten des kommutativen Konflikts zwischen t_1 und t_2 und darf somit nicht aktiviert sein, da er sonst die dritte Nebenbedingung verletzt.

Die letzte Bedingung gewährleistet, dass der Konfliktlösungsalgorithmus auf dem Vorgängergraphen vollständig durchgeführt wurde. Der Zustand *undefiniert* ist nur ein temporärer Zustand, der während der Durchführung der Konfliktlösung verwendet wird. Nachdem die Nebenbedingungen definiert wurden, wird in den nächsten Abschnitten erklärt wie diese durch den Konfliktlösungsalgorithmus eingehalten werden.[4]

[4] Der Konflikterkennungsalgorithmus wird nicht separat eingeführt. Er wird unter anderem auch innerhalb des Konfliktlösungsalgorithmus verwendet und deshalb im Rahmen des nächsten Abschnitts erläutert.

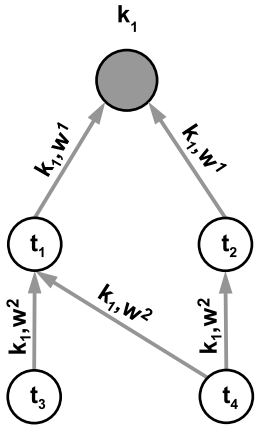


Abbildung 7.5: Vorgängergraph mit kommutativem Konflikt

7.3.2 Konfliktlösungsalgorithmus

Die Idee der Konfliktlösung besteht darin, miteinander in Konflikt stehende Transaktionen entsprechend eines Attributes (Zeitstempel, Priorität) zu ordnen und die Transaktion mit dem kleinsten bzw. größten Wert den Konflikt gewinnen zu lassen. In erster Linie sind dafür Echtzeit-Zeitstempel vorgesehen, es können aber auch andere extern vorgegebene Werte wie z. B. die Priorität verwendet werden. Wir betrachten folgende drei Konfliktlösungsverfahren:

1. Die älteste Transaktion gewinnt einen Konflikt.
2. Die jüngste Transaktion gewinnt einen Konflikt.
3. Die Transaktion mit der höchsten Priorität gewinnt einen Konflikt. Bei gleicher Priorität gewinnt die älteste Transaktion.

Alle drei Verfahren unterscheiden sich nur in der unterschiedlichen Ordnung der Transaktionen. Diese Eigenschaft macht sich der Konfliktlösungsalgorithmus zunutze. Als Eingabe erhält er eine Liste von Knoten, für die eine Konfliktlösung durchgeführt werden soll. Diese Knotenliste ist entsprechend der Ordnungsrelation des vorgesehenen Konfliktlösungsverfahrens geordnet und wird vom Konfliktlösungsalgorithmus in dieser Reihenfolge abgearbeitet. Dadurch werden die unterschiedlichen Konfliktlösungsverfahren im Konfliktlösungsalgorithmus umgesetzt und das Verfahren ist unabhängig von der Reihenfolge, in der die Transaktionen eingefügt wurden. Werden die Knoten z. B. nach ihren Prioritätswerten geordnet, ergibt sich daraus das prioritätsbasierte Konfliktlösungsverfahren.

Ablauf der Konflikterkennung und -lösung Um den Konfliktlösungsalgorithmus zu erklären, wird zunächst der Ablauf der Konfliktlösung im Rahmen des Replikationsverfahrens erläutert. Der

Ablauf ist im Folgenden dargestellt. Dabei wird zwischen dem Einfügen einer lokalen und einer entfernten Transaktion unterschieden.

- Lokale Transaktion:
  1. Lokale Ausführung einer Transaktion in der Datenbank des Teilnehmers
  2. Einfügen eines entsprechenden Knotens in den Vorgängergraphen ($insertGraph$)
- Entfernte Transaktion:
  1. Eintreffen der entfernten Transaktion
  2. Einfügen eines entsprechenden Knotens in den Vorgängergraphen ($insertGraph$)
  3. Überprüfen, ob die Transaktion einen Konflikt verursacht ($checkConflict$ anhand des Knotens)
  4. Im Falle eines Konflikts: Ausführung der Konfliktlösung auf allen Transaktionen ($solveConflicts$ auf den entsprechenden Knoten)

Bei der Ausführung einer lokalen Transaktion muss keine Konflikterkennung und -lösung vorgenommen werden, da lokale Transaktionen nicht zueinander nebenläufig ausgeführt werden können (siehe 2.4.3). Beim Einfügen einer entfernten Transaktion wird zunächst überprüft, ob diese mit anderen Transaktionen in Konflikt steht ($checkConflict$). Besteht ein Konflikt, so muss die Konfliktlösung durchgeführt werden. Der Konfliktlösungsalgorithmus ($solveConflicts$) erhält als Eingabe jeweils eine geordnete Liste aller Knoten. In Abbildung 7.6 ist ein Beispielablauf dargestellt.

Das Beispiel zeigt eine Replikationsgruppe mit den zwei Teilnehmern s_1 und s_2. Zu Anfang wird von Teilnehmer s_1 die Transaktion TA_1 ausgeführt. Dazu wird der entsprechende Knoten t_1 in den Vorgängergraphen von s_1 eingefügt. Anschließend wird t_1 gemeinsam mit seinen ausgehenden Kanten an s_2 versendet. Der Teilnehmer s_2 erhält den Knoten und überprüft, ob t_1 einen Konflikt verursacht. Da kein Konflikt besteht, wird t_1 aktiviert und somit vorläufig ausgeführt.

Etwas später führen die beiden Teilnehmer nebenläufig jeweils eine Transaktion aus. Nachdem die entsprechenden Knoten jeweils in den Vorgängergraphen eingefügt wurden, werden sie versendet. Die Knoten t_2 und t_3 werden jeweils in den Vorgängergraphen eingefügt, und anschließend wird überprüft, ob die Transaktionen einen Konflikt verursachen. Es besteht ein Konflikt und der Konfliktlösungsalgorithmus wird ausgeführt. Als Eingabe erhält der Algorithmus eine Knotenliste mit allen drei Knoten t_1, t_2 und t_3. Die Reihenfolge der Knoten in den Eingabelisten ist identisch, da als Ordnungskriterium die Zeitstempel der Transaktionen gelten. Aus diesen Grund wird t_3 vor t_2 eingeordnet.

Die Funktion *solveConflicts*

Die Funktion $solveConflicts$ stellt einen konfliktfreien Graphen her. Sie nimmt sich aus der geordneten Eingabeliste (NodeList) jeweils den nächsten Knoten und aktiviert oder deaktiviert diesen. Wird

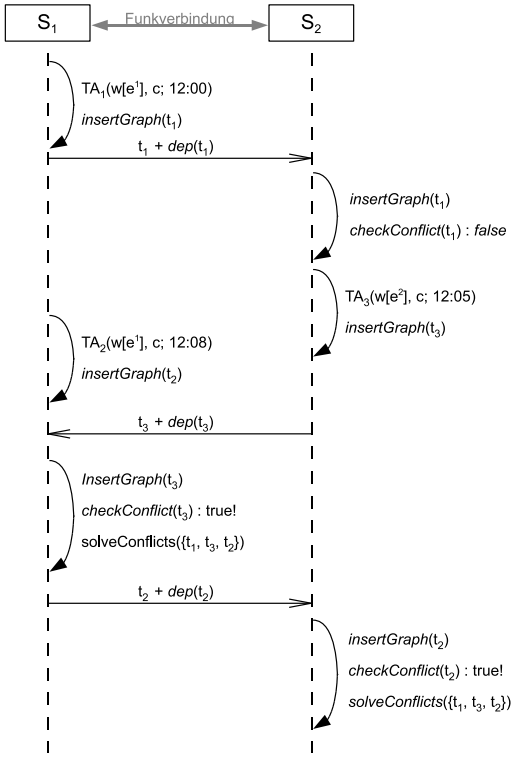


Abbildung 7.6: Ablauf der Konfliktlösung im Rahmen der Replikation (alle dargestellten Datenübertragungen sind asynchron)

ein Knoten aktiviert, werden ebenfalls alle Vorgängerknoten aktiviert ($activatePath$). Wird ein Knoten deaktiviert, werden ebenfalls alle Nachfolgerknoten deaktiviert ($deactivatePath$). Durch diese Vorgehensweise werden alle Knoten des Graphen partitionsweise abgearbeitet. In Listing 2 ist der Pseudocode der Konfliktlösungsfunktion $solveConflicts$ dargestellt.

Pseudocode In Listing 2 ist der Konfliktlösungsalgorithmus in Pseudocode dargestellt. In den Zeilen 3 - 7 werden zunächst alle Knoten der Liste auf undefiniert gesetzt, die noch nicht endgültig festgeschrieben wurden. Damit werden sie für den Konfliktlösungsalgorithmus als „zu bearbeiten“ markiert. Alle anderen Knoten werden ignoriert. Es folgt die eigentliche Konfliktlösung in den Zeilen 9 - 17. Entsprechend der Listenordnung werden die Knoten einzeln nacheinander betrachtet. Es werden nur Knoten behandelt, die noch undefiniert sind (Zeile 10).

In Zeile 11 wird zunächst überprüft, ob ein Knoten als kommutativ definiert wurde. Ist dies der Fall, so muss überprüft werden, ob alle aktivierten Geschwisterknoten aus derselben Kommutativitätsgruppe stammen wie der Knoten. Außerdem muss geprüft werden, ob bei einer Aktivierung des Knotens die dritte Nebenbedingung eingehalten wird, d. h. ob der Knoten auf allen aktivierten kom-

Listing 2 Konfliktlösungsalgorithmus

```
1:  function SOLVECONFLICTS(NodeList:List)
2:
3:      for all Node t ∈ NodeList do                ▷ Setze Zustand aller Knoten auf undefiniert
4:          if t.isNotCommitted() then              ▷ Nur wenn Knoten noch nicht festgeschrieben ist
5:              t.setUndefined()
6:          end if
7:      end for
8:
9:      for all Node t ∈ NodeList do                ▷ Prüfe, ob undefinierte Knoten aktiviert werden dürfen
10:         if t.isUndefined() then
                                                    ▷ Überprüfung der 3. Nebenbedingung
11:             if t.isCommutative() and !checkCommutativeCondition(t) then
12:                 deactivatePath(t)
13:             else
14:                 activatePath(t)                 ▷ Aufruf der Funktion activatePath (S. 110)
15:             end if
16:         end if
17:     end for
18:
19: end function
```

mutativen Knoten aufbaut. Diese Aufgabe wird von der Funktion $checkCommutativeCondition$ übernommen. Dabei werden die aktiven Geschwisterknoten der Eltern des Knotens t bestimmt und geprüft, ob diese ebenfalls Eltern von t sind.

Hält der Knoten die Nebenbedingungen für kommutative Knoten ein oder ist er nicht kommutativ, so darf er aktiviert werden. Dazu wird die Funktion $activatePath$ aufgerufen, die den Knoten und alle seine Vorgänger bis zur Wurzel aktiviert. Ist ein Knoten kommutativ und hält die Nebenbedingungen nicht ein, so werden der Knoten und alle seine Nachfolger deaktiviert. Die in Zeile 11 verwendete Funktion $activatePath$ ist ein zentraler Teil des Konfliktlösungsalgorithmus und wird deswegen im nächsten Abschnitt gesondert behandelt.

Die Funktion *activatePath*

Die Funktion $activatePath$ aktiviert den gesamten Pfad eines Knoten bis zu den Wurzeln und deaktiviert alle dazu in Konflikt stehenden Knoten. Dadurch werden in diesem Teil des Graphen alle Konflikte aufgelöst und sichergestellt, dass die erste Nebenbedingung eingehalten wird. Hierbei gewinnen immer die Knoten, die sich auf dem Pfad des Ursprungsknoten befinden, da dieser den mo-

mentan höchsten Rang in der Listenordnung besitzt. Der Algorithmus ist in Listing 3 in Pseudocode aufgeschrieben.

Listing 3 Aktivierung aller Vorgänger eines Knotens

```
 1: function ACTIVATEPATH(t : Node, firstStep : Boolean)
 2:
 3:     t.setActivated()                                  ▷ Aktiviere den Knoten
 4:     for all t_p ∈ t.getAllParents() do                ▷ Aktiviere alle Elternknoten
 5:         activatePath(t_p, false)
 6:     end for
 7:
 8:     if !firstStep then                                ▷ Prüfe, ob im ersten Rekursionschritt
 9:         for all t_cs ∈ t.getConflictingSiblings() do  ▷ Deakt. Pfade der Konfliktknoten
10:             if t_cs.isUndefined() then
11:                 deactivatePath(t_cs)
12:             end if
13:         end for
14:     end if
15:
16: end function
```

Pseudocode In Zeile 3 wird der übergebene Knoten aktiviert. In Zeile 4 - 6 wird die Funktion $activatePath$ rekursiv für alle Eltern des Knoten aufgerufen.

Anschließend werden alle mit dem Eingabeknoten in Konflikt stehenden Geschwisterknoten betrachtet und samt Nachfolgerknoten deaktiviert ($deactivatePath$) falls ihr Zustand undefiniert ist (Zeilen 9 –13). Dies darf erst ab dem zweiten Rekursionsschritt durchgeführt werden (Zeile 8). Beim ersten Aufruf der Funktion kann für den Eingabeknoten ein kommutativer Geschwisterknoten existieren, der ebenfalls noch aktiviert werden darf.

Korrektheit

In diesem Abschnitt wird gezeigt, dass der Konfliktlösungsalgorithmus korrekt arbeitet, d. h., dass die Nebenbedingungen nach dessen Anwendung auf einen Vorgängergraphen erhalten bleiben und dass das entsprechende Konfliktlösungsverfahren korrekt angewendet wird. Dies wird mit Hilfe einer Schleifeninvariante über der zentralen Schleife (Zeile 9 - 17) der Funktion $solveConflicts$ gezeigt.

Als Voraussetzung ist ein Vorgängergraph $G = (V, E)$ gegeben, der alle Nebenbedingungen erfüllt. Sei T^i die Menge der Knoten, die bis zu Beginn des i-ten Schrittes bereits aktiviert bzw.

deaktiviert wurden.

Folgende Invarianten entsprechen den in Abschnitt 7.3.1 angegebenen Nebenbedingungen und gelten vor dem i-ten Iterationsschritt für alle Knoten $t_j \in T^i$.

1. Wenn t_j aktiviert ist, so existiert kein $t_k \in T^i$, mit dem t_j in einem nicht-kommutativen Konflikt steht.
2. Wenn t_j aktiviert ist, so sind ebenfalls alle Vorgänger $t_p \in allpred(t_j)$ von t_j aktiviert.
3. Für alle Konfliktelemente von t_j, $k \in K_{t_j}$ und alle aktivierten Elternknoten $t_p \in parents(t_j, k)$ gilt, dass alle aktivierten Geschwisterknoten $t_s \in siblings(t_p, k)$ von t_p ebenfalls Elternknoten von t_j sind.
4. t_j ist entweder aktiviert oder deaktiviert.

Wir zeigen nun, dass diese Invarianten während der Iterationsschritte der Konfliktlösungsschleife erhalten bleiben.

Initialisierung Vor dem ersten Durchlauf ist $T^i = \emptyset$ und damit gelten trivialerweise alle Invarianten.

Iteration In jedem Einzelschritt der Schleife des Konfliktlösungsalgortihmus (Listing 2, Zeilen 9-17) wird ein neuer Knoten t_n betrachtet und am Ende für ihn die Funktion $activatePath$ oder $deactivatePath$ aufgerufen.

1. Fall (Aktivierung) Der neue Knoten t_n, die aktivierte Menge seiner Vorgänger T_{pred} und die Menge aller Knoten T_{conf}, die direkt oder indirekt mit den Knoten aus T_{pred} in Konflikt stehen, werden zur Menge T^i hinzugefügt, weil diese durch die Funktionen $activatePath$ und $deactivatePath$ aktiviert bzw. deaktiviert wurden. Somit muss gezeigt werden, dass die Menge $T^{i+1} = T^i \cup t_n \cup T_{pred} \cup T_{conf}$ die Invarianten erfüllt.

Aufgrund der Funktionsweise von $activatePath$ gilt für t_n und alle Knoten aus T_{pred} die zweite Invariante. Ebenso wird dadurch die erste Invariante eingehalten: Für jeden Knoten, der in direktem Konflikt mit einem aktivierten Knoten steht, wird $deactivatePath$ aufgerufen (Listing 3, Zeilen 9 - 13). Dadurch werden ebenfalls alle Knoten deaktiviert, die in indirektem Konflikt mit den Knoten aus T_{pred} stehen. Da in den Funktionen $activatePath$ und $deactivatePath$ alle betrachteten Knoten aktiviert bzw. deaktiviert werden, gilt somit auch die vierte Invariante.

Die dritte Invariante wird für den Knoten t_n direkt in der Funktion $solveConflicts$ in Zeile 11 überprüft. Für alle anderen Knoten wird die dritte Invariante beim Aktivieren der Pfade eingehalten: Nehmen wir an, dass die dritte Invariante für einen Knoten t_p aus T_{pred} nicht gilt. Als Veranschaulichung soll die Abbildung 7.5 auf Seite 106 dienen. Damit die dritte Invariante nicht gilt, müssten

beide Knoten t_3 und t_4 aktiviert sein. Damit beide Knoten von der Funktion $activatePath$ aktiviert werden könnten, müsste ein Kindknoten t_5 existieren, der sowohl von t_3 als auch t_4 abhängig ist. Dies kann jedoch nicht passieren, da der Konfliktlösungsalgorithmus vor der Ausführung von t_5 bereits einen der beiden Knoten t_3 oder t_4 deaktiviert haben muss, da sie nicht auf dieselben Elternknoten zugreifen. Somit kann innerhalb der Aktivierung eines Pfades immer nur einer der beiden Knoten t_3 und t_4 aktiviert werden. Es kann also durch die Aktivierung eines Pfades und die gleichzeitige Deaktivierung der in Konflikt stehenden Knoten kein Knoten existieren, der der dritten Invariante widerspricht.

Somit werden im ersten Fall alle vier Invarianten für die betrachteten Knoten eingehalten.

2. Fall (Deaktivierung) Wird t_n deaktiviert, so wird ebenfalls die Menge seiner Nachfolgerknoten T_{succ} deaktiviert. Von Knoten aus dieser Menge kann keine der vier Invarianten verletzt werden, da nur undefinierte Knoten deaktiviert werden. Auch t_n selbst verstößt durch die Deaktivierung gegen keine der Invarianten.

Schlussfolgerung Wir haben gezeigt, dass die neu eingefügte Transaktion t_n und die Menge der neu hinzugekommenen aktivierten bzw. deaktivierten Transaktionen nicht gegen die Invarianten verstoßen. Ebenso verstoßen die bisherigen Transaktionen T^i durch die Aktivierung bzw. Deaktivierung der neuen Transaktionen nicht gegen die Invarianten. Daraus folgt, dass T^{i+1} ebenfalls die Invarianten erfüllt.

Termination In jedem Schritt des $solveConflicts$ Algorithmus wird mindestens ein Knoten aktiviert bzw. deaktiviert. Jeder Knoten $t \in V$ des Vorgängergraphen wurde mindestens einmal aktiviert oder deaktiviert. Dies wurde entweder direkt getan, da t aus der abzuarbeitenden Knotenliste entnommen wurde oder indirekt bei der rekursiven Aktivierung der Vorgängerknoten oder Deaktivierung der Nachfolgerknoten eines Knotens durchgeführt. Ebenso wurde der Zustand jedes Knotens nicht mehrfach geändert, da der Zustand eines aktivierten oder deaktivierten Knotens nicht mehr geändert wird. Somit gelten bei der Termination des Algorithmus die Invarianten für alle Knoten des Graphen und dadurch sind die Nebenbedingungen für den Vorgängergraphen erfüllt.

Nachdem gezeigt wurde, dass der Konfliktlösungsalgorithmus die Invarianten eines Vorgängergraphen erfüllt, muss noch gezeigt werden, dass die Konflikte auch entsprechend des gewählten Konfliktlösungsverfahrens gelöst wurden. Dies kann mit Hilfe der Knotenliste gezeigt werden, die vom Konfliktlösungsalgorithmus schrittweise abgearbeitet wird.

Die Knotenliste des Konfliktlösungsalgorithmus ist entsprechend dem Ordnungskriterium des Konfliktlösungsverfahrens geordnet. Wenn die jüngste Transaktion gewinnen soll, so sind die Knoten der Liste absteigend entsprechend ihrer Echtzeit-Zeitstempel geordnet. Dadurch werden bei bestehenden Konflikten zuerst die jüngeren Knoten aktiviert, da sie zuerst aus der Knotenliste ent-

nommen werden. Durch diese Ordnung der Knotenliste ist immer eine Konfliktlösung im Sinne des Konfliktlösungsverfahrens gewährleistet.

Die Ausnahme bilden Konflikte, die Teil eines übergeordneten Konflikts sind. Hierbei kann es vorkommen, dass die Konfliktlösung nicht dem gewählten Verfahren entspricht. Dies ist jedoch irrelevant, da der übergeordnete Konflikt entscheidend ist. Hat man z. B. einen Konflikt zwischen zwei Transaktionen und wählt die Konfliktlösung mit Hilfe von Prioritätswerten, so zählt für die Konfliktlösung nur die Transaktion mit der höchsten Priorität. Dadurch kann es vorkommen, dass Transaktionen, die Vorgänger dieser Transaktion sind, eine geringere Priorität haben als die diejenige, die abgebrochen wurde.

Laufzeitverhalten

Bei der Betrachtung des Laufzeitverhaltens beziehen sich alle Angaben auf die Anzahl der Knoten im Graphen für die bisher nur ein pre-Commit durchgeführt wurde. Knoten, für die ein Commit durchgeführt wurde, müssen bei der Konfliktlösung nicht mehr betrachtet werden. Alle Hilfsfunktionen, die in den Listings 2 und 3 verwendet und in diesem Abschnitt nicht aufgeführt werden, benötigen eine konstante Laufzeit.

Bei der Initialisierung der Funktion $solveConflicts$ wird zunächst jeder Knoten einmal besucht. Anschließend wird in jedem Schritt der zentralen Schleife ein Teil des Graphen ausgehend von einem Ursprungsknoten von der Funktion $activatePath$ bzw. der Funktion $deactivatePath$ traversiert. Alle von diesen Funktionen besuchten Knoten sind anschließend aktiviert bzw. deaktiviert und werden somit nur noch einmal in der Funktion $solveConflicts$ betrachtet.

Aufgrund der Assoziativität und Symmetrie von Konflikten kann es nicht vorkommen, dass diese Knoten erneut von der Funktion $activatePath$ (inklusive $getAllParents$) oder der Funktion $deactivatePath$ (inklusive $getConflictingSiblings$) betrachtet werden. Das liegt daran, dass durch diese beiden Funktionen jeweils disjunkte Teilgraphen traversiert werden. Lässt man die Funktion $checkCommutativeCondition$ zunächst außer Betracht, ergibt sich für den Konfliktlösungsalgorithmus insgesamt eine lineare Laufzeit bezüglich der Knoten- und Kantenanzahl, da jeder Knoten dabei höchstens dreimal und jede Kante höchsten einmal besucht wird.

Durch die Funktion $checkCommutativeCondition$ kann es passieren, dass ein Knoten mehr als dreimal besucht wird, weil die Funktion alle am Konflikt beteiligten Knoten erneut betrachtet, um die dritte Nebenbedingung zu prüfen. Die Funktion wird jedoch nur selten aufgerufen, da sie nur bei der direkten Betrachtung eines Knotens in der Funktion $solveConflicts$ benötigt wird (siehe S. 111). Außerdem ist die Anzahl der an einem Konflikt beteiligten Knoten in der Praxis gering. In den experimentellen Untersuchungen dieser Arbeit sind in 98 Prozent der Fälle zwei bis vier Knoten an einem Konflikt beteiligt. Somit wirkt sich das asymptotisch quadratische Laufzeitverhalten nur sehr selten auf die gesamte Laufzeit des Konfliktlösungsalgorithmus aus.

Der Speicherplatzbedarf eines Vorgängergraphen ist proportional zur Anzahl der Transaktionen

und deren Abhängigkeiten, die durch ihn repräsentiert werden, da der Speicherbedarf eines Knotens und einer Kante jeweils konstant ist.

7.3.3 Globale Konsistenz

Bisher wurde die Konflikterkennung beim Einfügen von lokalen (siehe 7.2.3) und entfernten (siehe 7.2.4) Transaktionen und die Konfliktlösung innerhalb eines Vorgängergraphen erläutert. Nun betrachten wir, wie diese Methoden zusammenspielen, um einen globalen konsistenten Zustand zu erreichen. Die grundlegende Idee ist, die kausale Ordnung aller nicht in Konflikt stehenden Transaktionen und die totale Ordnung aller in Konflikt stehenden Transaktionen zu nutzen und somit zu einer global eindeutigen Lösung zu gelangen.

Um einen konsistenten Zustand zu erreichen, muss jeder Replikationsteilnehmer zunächst alle bis dahin durchgeführten Transaktionen erhalten haben. Somit ergibt sich für die jeweiligen Vorgängergraphen der Teilnehmer die gleiche Menge an Knoten. Durch die Verwendung des Vorgängergraphen ergibt sich für alle Transaktionen ebenfalls dieselbe kausale Ordnung. Das Attribut, das zur Konfliktlösung verwendet wird (z. B. Echtzeit-Zeitstempel) wird der Transaktion und somit auch dem Knoten lokal von dem Teilnehmer zugewiesen, auf dem die Transaktion initiiert wurde. Dies geschieht einmalig, danach wird das Attribut nicht mehr angepasst bzw. verändert. Somit besitzen alle Knoten aller Teilnehmer dieselben Ordnungsattribute und werden in der Knotenliste des Konfliktlösungsalgorithmus auf die gleiche Weise geordnet. Da der Konfliktlösungsalgorithmus deterministisch ist, liefert er für jede Eingabemenge ein eindeutiges Ergebnis.

Somit sind alle Transaktionen, die auf den gleichen Daten arbeiten entweder kausal geordnet oder es besteht ein Konflikt (kausale Parallelität). In diesem Fall werden bei allen Teilnehmern dieselben Transaktionen aktiviert bzw. deaktiviert, da für alle am Konflikt beteiligten Transaktionen aufgrund ihres Ordnungsattributes eine Totalordnung existiert, die für die Konfliktlösung verwendet wird. Die resultierenden Vorgängergraphen aller Replikationsteilnehmer sind dadurch gleich, wodurch ein global konsistenter Zustand erreicht wird.

7.4 Commitverfahren

Im letzten Abschnitt wurde gezeigt, wie für alle Replikationsteilnehmer ein konsistenter Zustand hergestellt werden kann, wenn alle Teilnehmer die gleichen Transaktionen erhalten haben. In diesem Abschnitt wird nun gezeigt, wie durch das Commitverfahren sichergestellt wird, dass alle Teilnehmer die Konfliktlösung auf den gleichen Transaktionen durchführen damit diese endgültig festgeschrieben bzw. abgebrochen werden können. Dadurch wird das Konsistenzkriterium *Eventual Consistency* erfüllt.

Das von uns verwendete Commitverfahren ist zeitstempelbasiert und verwendet die in Kapitel 6 beschriebenen Echtzeit-Zeitstempel. Transaktionen werden dabei nicht einzeln betrachtet, sondern

es werden alle Transaktionen in einem bestimmten Zeitintervall festgeschrieben. Ein Intervall wird durch seine Commitpunkte festgelegt und als **Commitintervall** bezeichnet.

Weiß der Replikationsmanager, dass er alle Transaktionen eines Commitintervalls erhalten hat, so führt er den Konfliktlösungsalgorithmus nur auf den Transaktionen des Intervalls aus.[5] Diese Entscheidung wird lokal anhand der Informationen getroffen, die von anderen Teilnehmern versendet wurden. Abschließend werden alle Transaktionen aus diesem Intervall endgültig festgeschrieben bzw. abgebrochen. Damit ist gewährleistet, dass alle Replikationsteilnehmer den Konfliktlösungsalgorithmus auf der gleichen Menge an Transaktionen ausführen und somit zu einem konsistenten Ergebnis kommen. In Abbildung 7.7 ist ein Beispiel für die verschiedenen Commitintervalle gezeigt, die sich aufgrund der Commitpunkte ergeben.

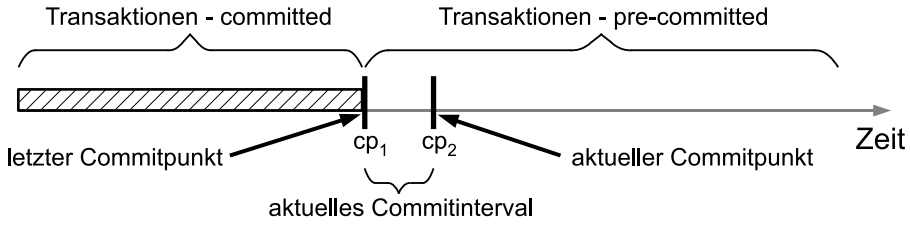


Abbildung 7.7: Verwendung von Commitpunkten

Durch die Commitpunkte cp_1 und cp_2 sind die Transaktionen in unterschiedliche Zeitintervalle eingeteilt. Die Transaktionen im Zeitintervall vor cp_1 konnten festgeschrieben werden, da der Teilnehmer bereits alle Transaktionen aus diesem Zeitintervall erhalten hat. Ob ein Teilnehmer alle Transaktionen eines anderen Teilnehmers bis zu einem bestimmten Zeitpunkt erhalten hat, kann er aufgrund der Zeitstempel und der lokalen Sequenznummer der Transaktionen bestimmen. Hat er bereits eine Transaktion von einem anderen Teilnehmer erhalten, deren Zeitstempel größer als der Commitpunkt ist und sind die Sequenznummern der Transaktionen lückenlos, hat er alle Transaktionen bis zum Commitpunkt erhalten. Damit die Inaktivität von Replikationsteilnehmern das Commit nicht unnötig verzögert, werden in bestimmten Abständen Statusnachrichten versendet.

Für das Intervall $[cp_1, cp_2]$ kann noch nicht bestimmt werden, ob alle Transaktionen erhalten wurden. Deswegen können diese Transaktionen noch nicht festgeschrieben werden. Aus diesem Grund werden alle Transaktionen, die einen Zeitstempel größer als cp_1 besitzen, nur vorläufig ausgeführt. Im Folgenden wird nun die Korrektheit dieses Commitverfahrens gezeigt, indem die Kriterien für *Eventual Consistency* überprüft werden.

[5] Der Konfliktlösungsalgorithmus kann ohne Einschränkung der Korrektheit auch auf Teilen des Vorgängergraphen ausgeführt werden. Bei der Initialisierung werden dann anstatt der leeren Menge die bereits festgeschriebenen und noch nicht gelöschten (siehe Abschnitt 7.4.5) Transaktionen verwendet. Da diese ebenfalls durch den Konfliktlösungsalgorithmus behandelt wurden, erfüllen sie die Nebenbedingungen und müssen nicht mehr betrachtet werden.

7.4.1 Gewährleistung von Eventual Consistency

Die entscheidende Voraussetzung für die korrekte Ausführung des Commitverfahrens ist eine global konsistente Wahl der Commitpunkte. Dies ist durch die Verwendung der Echtzeit-Zeitstempel (siehe Kapitel 6) gewährleistet. Somit sind die Commitintervalle bei allen Teilnehmern gleich und dadurch auch die Mengen der Transaktionen, auf denen der Konfliktlösungsalgorithmus abschließend ausgeführt wird. Dadurch werden bei allen Teilnehmern bis zum entsprechenden Commitpunkt die gleichen Transaktionen festgeschrieben bzw. abgebrochen.

Wir zeigen nun, wie dadurch die Kriterien von *Eventual Consistency* eingehalten werden. Zur Verbesserung der Lesbarkeit werden diese Kriterien basierend auf der in Abschnitt 2.3.7 gegebenen Definition an unser Systemmodell angepasst. Dafür muss zuerst der Präfix eines Vorgängergraphen definiert werden.

Präfix eines Vorgängergraphen Der Präfix eines Vorgängergraphen ist ein durch die Knotenmenge U induzierter Teilgraph, für den $U = allpred(U)$ gilt und der pro Konfliktelement mindestens einen Knoten enthält, der auf dieses Element zugreift.

Eventual Consistency Unter der Voraussetzung, dass alle Replikationsteilnehmer denselben Ausgangszustand besitzen, wird *Eventual Consistency* gewährleistet, wenn die folgenden Bedingungen gelten:

1. Zu jedem Zeitpunkt und für jeden Replikationsteilnehmer existiert ein Präfix eines Vorgängergraphen, das äquivalent zu einem Präfix eines Vorgängergraphen jedes anderen Replikationsteilnehmers ist. Dieses wird als *festgeschriebenes Präfix* (engl. committed prefix) bezeichnet.
2. Das festgeschriebene Präfix jedes Replikationsteilnehmers wächst monoton über die Zeit.
3. Alle nicht abgebrochenen Transaktionen in dem festgeschriebenen Präfix erfüllen die Nebenbedingungen des Vorgängergraphen.
4. Für jede initiierte Transaktion gilt, dass *irgendwann* entweder die festgeschriebene Transaktion oder die abgebrochene Transaktion im festgeschriebenen Präfix enthalten ist.

Die in Bedingung 1 genannten Präfixe ergeben sich aus der Menge der Transaktionen, die in den jeweiligen Commitintervallen enthalten sind. Da die Commitintervalle bei allen Teilnehmern gleich sind und aufeinander aufbauen, ist die erste Bedingung erfüllt. Die Commitintervalle sind durch Commitpunkte festgelegt, die einen Echtzeit-Zeitstempel besitzen. Da kein Commitintervall ausgelassen werden darf, wächst die Menge der festgeschriebenen Transaktionen monoton über die Zeit. Damit ist Bedingung 2 erfüllt.

Wie bereits gezeigt, sorgt der Konfliktlösungsalgorithmus dafür, dass die zu den Transaktionen gehörigen Knoten die Nebenbedingungen des Vorgängergraphen einhalten. Dies gilt somit ebenso

für die aktivierten Transaktionen des Commitintervalls, die festgeschrieben werden, wodurch Bedingung 3 erfüllt ist. Durch die Nebenbedingungen ist festgelegt, dass eine Transaktion entweder aktiviert oder deaktiviert sein muss. Wenn ein Commit durchgeführt wird, so werden die aktivierten Transaktionen endgültig festgeschrieben und die deaktivierten Transaktionen endgültig abgebrochen. Dadurch ist auch die letzte Bedingung erfüllt.

Somit wird mit Hilfe unseres Commitverfahrens Eventual Consistency gewährleistet. Wie eingangs erwähnt, wird für dieses Verfahren vorausgesetzt, dass global konsistente Commitpunkte festgelegt wurden. In den nächsten Abschnitten werden drei unterschiedliche Ansätze zur Festlegung der Commitpunkte erläutert.

7.4.2 Manuelles Commit

Das manuelle Commit ist das einfachste Verfahren zur Festlegung von global konsistenten Commitpunkten. Es besitzt jedoch den Nachteil, dass für die Verteilung des Commitpunkts zusätzlicher Nachrichtenaufwand entsteht. Bei diesem Verfahren können die Replikationsteilnehmer eigenständig Commitpunkte initiieren, die ebenso wie Transaktionen an alle anderen Teilnehmer verteilt werden. Dieses Verfahren bietet sich besonders an, wenn das Commit durch den Nutzer bzw. von der Applikation gesteuert werden soll. Die Commitpunkte werden jeweils an alle Teilnehmer verteilt und gewährleisten somit Eventual Consistency.

Die parallele Initiierung von Commitpunkten stellt kein Problem dar, da diese ebenfalls entsprechend ihrem Zeitstempel geordnet sind und abgearbeitet werden. Ebenso wie bei den Transaktionen (siehe 2.4.1) werden für die Commitpunkte Sequenznummern vergeben und berücksichtigt. So kann es nicht passieren, dass die einzelnen Teilnehmer die Commitpunkte in unterschiedlicher Reihenfolge abarbeiten oder Commitpunkte auslassen.

7.4.3 Intervallbestimmtes Commit

Bei dem intervallbestimmten Commit wird am Anfang für die Replikationsteilnehmer ein Startzeitpunkt $offset$ und eine Intervallgröße i_{size} festgelegt, die den Abstand zwischen zwei Commitpunkten definiert. Mit Hilfe dieser beiden Werte kann jeder Replikationsteilnehmer die Zeitpunkte ts_{commit} der Commitpunkte lokal berechnen und es entsteht kein zusätzlicher Nachrichtenaufwand. Die Berechnung des aktuellen Commitzeitpunktes zum Zeitpunkt $time$ ist wie folgt:

$$ts_{commit} = \left\lfloor \frac{time - offset}{i_{size}} \right\rfloor \cdot i_{size} + offset$$

Da alle Teilnehmer die gleichen Ausgangswerte (*offset*) verwenden, sind diese Commitpunkte global konsistent. Wie bisher werden alle Commits erst ausgeführt, wenn sichergestellt ist, dass von allen anderen Replikationsteilnehmern alle Ereignisse bis zu diesem Zeitpunkt erhalten wurden.

7.4.4 Implizites Commit

Das dritte Verfahren zur Festlegung der Commitpunkte legt diese nicht explizit zu bestimmten Zeitpunkten fest, sondern verwendet als Commitpunkte die Zeitstempel der Transaktionen. Aufgrund der unterschiedlichen Verteilung der Transaktionen auf die Replikationsteilnehmer können dadurch auch unterschiedliche Commitpunkte gewählt werden. Deshalb kann dieses Verfahren nur bei dem Echtzeit-Zeitstempel Konfliktlösungsverfahren verwendet werden, bei dem die älteren Transaktionen gewinnen. Bei allen anderen Verfahren kann das implizite Commit zu inkonsistenten Daten führen.

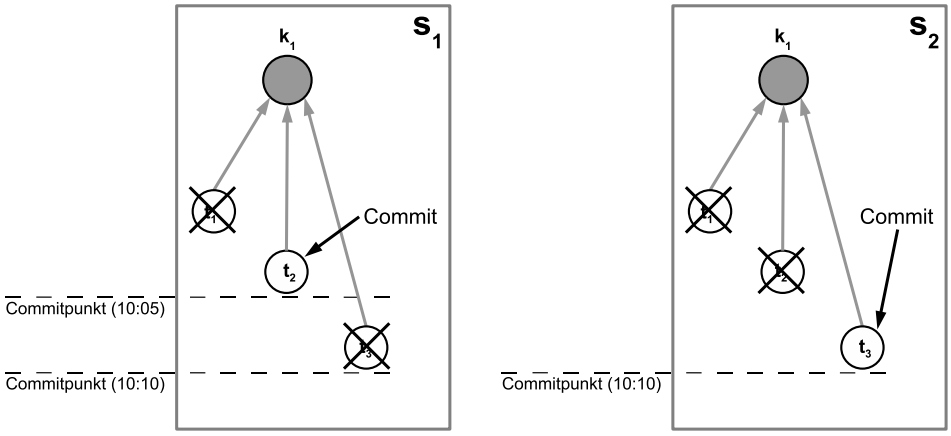


Abbildung 7.8: Verwendung unterschiedlicher Commitpunkte

Als Beispiel dafür betrachten wir einen Konflikt zwischen drei Transaktionen, die jeweils um 10:00 Uhr (t_1), 10:05 Uhr (t_2) und 10:10 Uhr (t_3) durchgeführt wurden (siehe Abbildung 7.8). Als Konfliktlösung werden die Echtzeit-Zeitstempel verwendet und die jeweils jüngste Transaktion gewinnt. Wenn der Teilnehmer s_1 nur die zwei Transaktionen t_1 und t_2 kennt und 10:05 Uhr als Commitpunkt verwendet, gewinnt t_2 diesen Konflikt und wird festgeschrieben. Wenn der Teilnehmer s_1 den Knoten t_3 erhält und ein Commit mit einem Commitpunkt um 10:10 Uhr durchgeführt wird, so muss t_3 abgebrochen werden, da t_2 bereits festgeschrieben ist. Erhält ein anderer Teilnehmer s_2 alle drei Transaktionen und verwendet dieser 10:10 Uhr als Commitpunkt, so gewinnt t_3 und wird anschließend festgeschrieben. Somit kommen die beiden Replikationsteilnehmer zu unterschiedlichen Ergebnissen, obwohl sie den gleichen aktuellen Commitpunkt verwenden.

Hätte man für das Konfliktlösungsverfahren die Ordnung verwendet, bei der die ältere Transaktion gewinnt, so wäre bei beiden Teilnehmern unabhängig vom jeweiligen Eintreffen der Transaktionen die Transaktion t_1 festgeschrieben worden. Im folgenden Abschnitt zeigen wir nun, dass bei diesem Konfliktlösungsverfahren die Commitpunkte unterschiedlich gewählt werden können, ohne Dateninkonsistenzen zu verursachen.

Beweis

Wir zeigen per Widerspruchsbeweis, dass keine Transaktion, deren Zeitstempel größer ist als der Zeitstempel einer anderen Transaktion die Commitentscheidungen für diese Transaktionen verändern kann. Dieses gilt nur für das zeitstempelbasierte Konfliktlösungsverfahren, bei dem die ältere Transaktion gewinnt.

Gegeben seien zwei Transaktionen t_i und t_j mit den Zeitstempeln ts_i und ts_j für die gilt $ts_i < ts_j$. Nehmen wir an, die Transaktion t_j verändert die Commitentscheidung, die für t_i getroffen wurde. Daraus folgt, dass t_j mit t_i in einem direkten bzw. indirekten Konflikt steht.

1. Fall Bei einem *direkten Konflikt* besitzen die beiden Knoten denselben Elternknoten. Damit die Commitentscheidung verändert werden müsste, wäre es notwendig, dass der Konfliktlösungsalgorithmus t_j als gewinnende Transaktion festlegt. Dafür muss $ts_i > ts_j$ gelten, dies steht jedoch im Widerspruch zur Annahme.

2. Fall Bei einem *indirekten Konflikt* besitzen t_j und t_i einen gemeinsamen Vorgänger t_p, für dessen Zeitstempel $ts_p < ts_i$ gelten muss. Da t_p einen kleineren Zeitstempel als t_i besitzt, wurde bereits vor t_i und t_j entschieden, ob t_p aktiviert wird oder nicht. Somit kann der Knoten t_j das Ergebnis von t_p nicht beeinflussen. Da t_j nur in einem indirekten Konflikt mit t_i steht, kann er auch nicht mehr das Ergebnis von t_i beeinflussen.

Damit ist gezeigt, dass das implizite Commit bei dem Konfliktlösungsverfahren, das auf Echtzeit-Zeitstempel aufbaut und die älteste Transaktion bevorzugt, trotz unterschiedlicher Commitpunkte keine dauerhaften Dateninkonsistenzen verursacht.

7.4.5 Trimmen des Vorgängergraphen

Durch das Festschreiben von Transaktionen wird nicht nur Eventual Consistency gewährleistet, sondern auch das Kürzen der Vorgängergraphen ermöglicht. Die Transaktionen wurden endgültig festgeschrieben bzw. abgebrochen und die Informationen über die Ausführungsreihenfolge der Transaktionen werden nicht mehr benötigt. Aus diesem Grund werden nach jedem Commit ebenfalls die unnötigen Knoten aus dem Vorgängergraphen gelöscht und die übrigen Verweise entsprechend angepasst. Die endgültig festgeschriebenen bzw. abgebrochenen Transaktionen werden von der Konflikterkennung und -lösung nicht mehr berücksichtigt (siehe S. 115) .

Beim Trimmen des Vorgängergraphen werden folgende Aktionen auf den Knoten der festgeschriebenen bzw. abgebrochenen Transaktionen ausgeführt:

- Es werden alle Knoten gelöscht, die deaktiviert sind.

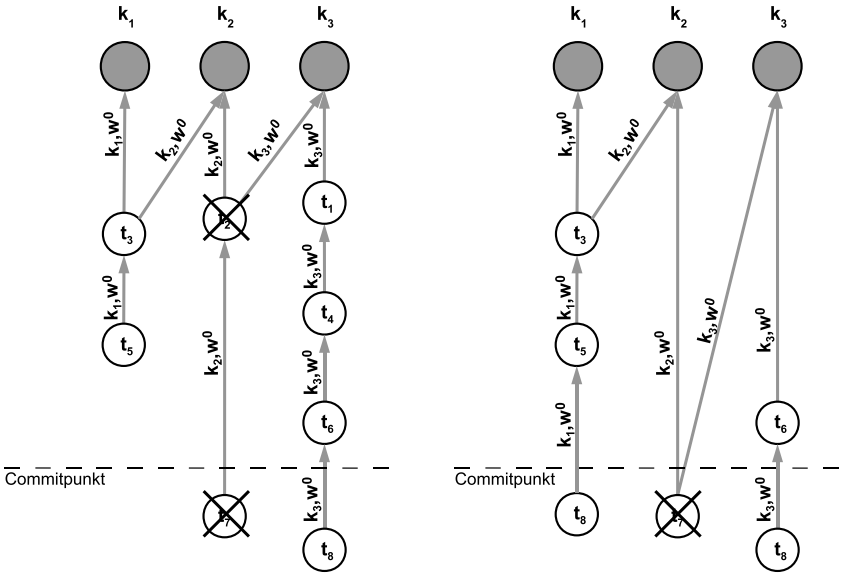
- Bis auf die Knoten, die jeweils zuletzt auf ein Konfliktelement zugegriffen haben, werden alle Knoten gelöscht.
- Die Kanten, die auf gelöschte Knoten verweisen, werden auf die entsprechenden Vorgänger umgesetzt. Ist kein Vorgänger vorhanden, so wird auf den Wurzelknoten verwiesen.

Das Verfahren löscht nicht alle Knoten, deren Zeitstempel vor dem Commitpunkt liegen. Das liegt daran, dass die Bezugspunkte für die Kanten jüngerer Knoten (nach dem Commitpunkt) erhalten bleiben sollen. Besonders für Knoten, die neu in den Graphen eingefügt werden, würde dies ein Problem darstellen, da ohne die Kenntnis vom Zustand des jeweiligen Elternknotens nicht entschieden werden kann, ob der Knoten aktiviert oder deaktiviert werden soll (siehe 7.3.1).

Trotzdem kann es vorkommen, dass sich ein Knoten auf einen bereits gelöschten Knoten bezieht. Entweder wurde dieser Knoten gelöscht, weil er deaktiviert war oder er besaß einen Kindknoten, der später auf dieselben Konfliktelemente zugegriffen hat. In beiden Fällen muss ein Knoten, der auf eine solchen gelöschten Knoten verweist, deaktiviert werden, da dies ansonsten der ersten bzw. zweiten Nebenbedingung widersprechen würde. Die entsprechenden Kanten des Knotens werden auf die Wurzelknoten der Konfliktelemente gesetzt.

Um festzustellen, ob ein Knoten auf einen Knoten verweist, der vor dem Commitpunkt liegt, wird für jede Kante ebenfalls der Zeitstempel des Knotens festgehalten, auf den die Kante zeigt. Wird auf einen fehlenden Knoten verwiesen, kann dadurch festgestellt werden, ob der Knoten bereits gelöscht wurde (Zeitstempel *vor* dem Commitpunkt) oder noch nicht vom Teilnehmer erhalten wurde (Zeitstempel *nach* dem Commitpunkt).

In Abbildung 7.9 ist ein Beispiel für das Trimmen eines Vorgängergraphen gegeben. Im linken Bild sieht man den noch vollständigen Graphen, in dem die Transaktionen t_2 und t_7 deaktiviert sind, da sie mit den anderen Transaktionen in Konflikt stehen. Im rechten Bild ist der bis zum Commitpunkt getrimmte Graph gezeigt. Knoten t_6 bleibt erhalten, da er der letzte Knoten des Konfliktelements k_3 ist, analog bleibt t_5 für Konfliktelement k_1 erhalten. Da t_5 nur auf das Element k_1 zugreift, bleibt t_3 ebenfalls erhalten. Der Knoten t_7 verweist auf den gelöschten Knoten t_2. Dadurch wird er deaktiviert und seine Verweise auf die Wurzelknoten von k_2 und k_3 gesetzt.



(a) Graph vor dem Trimmen (b) Getrimmter Graph

Abbildung 7.9: Trimmen eines Vorgängergraphen

7.5 Zusammenfassung

In diesem Kapitel haben wir gezeigt, auf welche Weise die dezentrale Konflikterkennung und Konfliktlösung in unserem Replikationsansatz realisiert werden. Das Verfahren basiert auf der Verwendung der kausalen Abhängigkeiten der Transaktionen. Die Abhängigkeiten werden im Replikationssystem von einer Datenstruktur, dem sogenannten Vorgängergraphen, verwaltet. Der Vorgängergraph wird für die Konflikterkennung und -lösung unseres Replikationsansatzes verwendet. Zunächst wurde die Datenstruktur definiert und erläutert, wie damit Konflikte erfasst werden können.

Anschließend wurde der Konfliktlösungsalgorithmus vorgestellt, der verwendet wird, um entstandene Konflikte konsistent aufzulösen. Er verwendet dazu, die ihm zu diesem Zeitpunkt lokal zur Verfügung stehenden Informationen und wartet nicht eine globale Entscheidung ab. Aus diesem Grund kann vom Replikationssystem eine hohe Datenverfügbarkeit gewährleistet werden, da über die lokale Ausführung einer Transaktionen sofort entschieden wird.

Die grundlegende Idee des Konfliktlösungsalgorithmus besteht darin, für die in Konflikt stehenden (kausal parallelen) Transaktionen eine global konsistente Ordnung (totale Ordnung) zu verwenden und die Transaktion mit dem kleinsten Ordnungsattribut gewinnen zu lassen. Die anderen am Konflikt beteiligten Transaktionen werden abgebrochen. Aufgrund der konsistenten Ordnung kommen alle Teilnehmer zum gleichen Ergebnis, vorausgesetzt diese sehen die gleichen Transaktionen. Als Ordnungskriterium können die in Kapitel 6 beschriebenen Echtzeit-Zeitstempel oder auch Prioritäts-

werte verwendet werden. Die Korrektheit des Konfliktlösungsalgorithmus wurde ebenfalls gezeigt.

Abschließend wurde das Commitverfahren unseres Replikationsansatzes beschrieben. Es basiert auf Commitpunkten, die Echtzeit-Zeitstempel verwenden. Diese Punkte bilden Commitintervalle, die bestimmen, welche Transaktionen festgeschrieben werden. Festgeschriebene bzw. endgültig abgebrochene Transaktionen brauchen nicht mehr im Vorgängergraphen gehalten zu werden und werden von einer Trimming Methode entfernt.

Kapitel 8

SYMORE - ein Datenbankreplikationssystem für MANETs

In diesem Kapitel stellen wir das Datenbankreplikationssystem SYMORE vor. Dieses Replikationssystem ist eine prototypische Implementierung[1] der von uns in den vorangegangenen Kapiteln vorgestellten Replikationsmethoden. SYMORE ist in Java (J2ME) *Connected Device Configuration* (CDC) für kleine und mobile Geräte implementiert und somit sowohl auf Laptops als auch PDAs oder Smartphones mit WLAN einsetzbar. Die Implementierung dient einerseits als Proof-of-Concept unserer Replikationsmethoden und andererseits als Basis für eine Evaluierung der Gesamtheit unserer entwickelten Ansätze (siehe Kapitel 10).

Um einen Überblick über die Implementierung zu bekommen, wird in Abschnitt 8.1 zunächst die Gesamtarchitektur erläutert. Anschließend werden die einzelnen Schritte der Transaktionsverarbeitung aus Implementierungssicht dargestellt (Abschnitt 8.2). In Abschnitt 8.3 werden zwei Beispielapplikationen vorgestellt, die SYMORE für die Verwaltung ihrer verteilten Daten verwenden.

8.1 Architektur

Die Architektur des Datenbankreplikationssystems SYMORE ist in Abbildung 8.1 dargestellt. Da SYMORE für kleine und mobile Geräte konzipiert ist, muss es nicht als externes System verwendet werden, sondern kann ähnlich wie andere Java Datenbanken wie Derby [der10] oder HSQLDB [hdb10] in die Virtual Machine der Applikation integriert werden. Die Applikationsschnittstelle von SYMORE und die vier verschiedenen Teilkomponenten werden im Folgenden kurz erläutert.

Applikationsschnittstelle Die Applikation greift auf SYMORE über eine JDBC-Schnittstelle zu, wie sie im Java Community Process JSR 169 von J2ME für kleine und mobile Geräte definiert ist. Die API wurde dafür speziell an die Anforderungen von mobilen Geräten mit begrenzten Ressourcen angepasst. SYMORE erscheint der Applikation somit wie ein geschlossenes repliziertes Datenbanksystem. Die interne Verwendung eines Datenbanksystems ist für die Applikation transparent. Die Applikation kann über die Schnittstelle mit DDL und DML Befehlen

[1] Eine ausführliche Darstellung der Implementierung wird in [Bre06] gegeben.

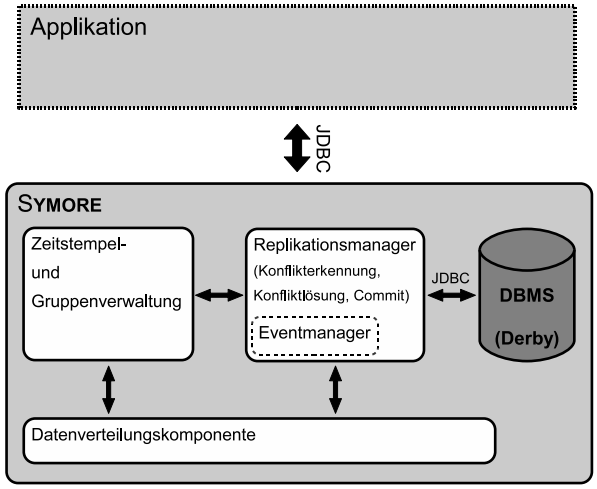


Abbildung 8.1: Architektur des Replikationssystems SYMORE

auf die replizierte Datenbank zugreifen. Replikationsspezifische Befehle sind als SQL Erweiterungen implementiert. Eine genaue Übersicht über alle SQL Befehle ist in der Arbeit von Frank Bregulla [Bre06] angegeben.

Rückmeldungen vom Replikationssystem werden mit Hilfe von Events (Ereignisse) realisiert. Die Applikation kann Beobachter für verschiedene Teile (Tabelle, Spalte, gesamte DB) beim Replikationssystem anmelden. Diese werden informiert, sobald auf dem entsprechenden Teil eine Änderung durchgeführt wird. Eine genaue Spezifikation dieser Schnittstelle befindet sich ebenfalls in der Arbeit von Frank Bregulla [Bre06].

Replikationsmanager Der Replikationsmanager (RM) ist die zentrale Verwaltungskomponente des Systems und koordiniert den gesamten Ablauf des Replikationsvorgangs. Der RM beinhaltet die Konflikterkennung, die Konfliktlösung und verwaltet ebenfalls das Commit-Verfahren. Dazu greift er auf das Datenbanksystem, die Datenverteilungskomponente und auf die Gruppen- und Zeitstempelverwaltung zu.

Gruppen- und Zeitstempelverwaltung Die Verwaltung der Zeitstempel und der Replikationsgruppe ist nicht im Replikationsmanager integriert, sondern in einer separaten Komponente gekapselt. Dieses dient der Erweiterbarkeit des Systems für zukünftige Untersuchungen. Neue Methoden zur Uhrensynchronisation oder zur Gruppenverwaltung können dadurch einfacher in das System integriert werden.

Datenbanksystem Innerhalb von SYMORE wird ein eigenes Datenbankmanagementsystem verwendet, auf das über eine JDBC Schnittstelle zugegriffen wird. Dadurch besteht zwischen

den Systemen nur eine lose Kopplung und somit kann bei Bedarf das DBMS problemlos ausgetauscht werden. Bei der im Rahmen dieser Arbeit behandelten Implementierung wird das Datenbanksystem Apache Derby verwendet, da dieses JDBC (JSR 169) unterstützt.

Datenverteilungskomponente Die Datenverteilungskomponente liegt eine Schicht unterhalb des Replikationsmanagers und der Gruppen-/Zeitstempelverwaltung. Sie ist für die Kommunikation mit den anderen Teilnehmern der Replikationsgruppe zuständig. Die zu verteilenden Objekte werden der Komponente übergeben und diese verwaltet deren Verteilung eigenständig. Der Replikationsmanager und die Gruppen-/Zeitstempelverwaltung können die Verteilung dann nicht weiter beeinflussen.

Sowohl die in Kapitel 5 vorgestellten Flutungsprotokolle, als auch die beiden Synchronisationsprotokolle sind in der Datenverteilungskomponente implementiert. Es kann jeweils ein Flutungs- und ein Synchronisationsprotokoll parallel verwendet werden. Die Einstellungen der Verteilungskomponente werden bei der Initialisierung vorgenommen und sind danach dauerhaft wirksam. Eine detaillierte Beschreibung der Implementierung wird in [Sch06] angegeben.

8.2 Transaktionsverarbeitung

Um einen Einblick in die Funktionsweise der Implementierung des Replikationsmanagers zu bekommen, werden die einzelnen Schritte der Transaktionsverarbeitung erläutert. Die wichtigsten Klassen, die an diesem Prozess beteiligt sind, werden in Abbildung 8.2 gezeigt.

Wie bereits erwähnt, greift die Applikation per JDBC auf das System zu. Mit Hilfe eines JDBC `Connection` Objekts kann sie `Statement` Objekte erzeugen, welche die eigentlichen SQL Anfragen beinhalten. Jedes `Statement` Objekt ist einem `Transaction` Objekt zugeordnet. Ein `Transaction` Objekt enthält ein oder mehrere `Statement` Objekte. Wird für das erste Statement einer `Transaction` die `execute` Methode aufgerufen, so wird die Transaktion implizit abgeschlossen und lokal festgeschrieben. Im Falle eines Aborts wird für die einzelnen Statements die `undo` Methode aufgerufen.

Die Abbildung 8.2 gibt ebenfalls eine Übersicht über die Hierarchie der Statement Klassen[2]. Alle Statements erben von dem übergeordneten Interface `Statement`. Die Unterklassen (`SelectStatement`, `DDLStatement`, `DMLStatement`) repräsentieren die unterschiedlichen Klassen von SQL Statements. Darüberhinaus wurden replikationsspezifische Statements wie z. B. `ForcedCommitStatement` hinzugefügt.

Wenn eine Transaktion verarbeitet wird, werden zunächst die enthaltenen Statements einzelnen *geparst* und *ausgeführt*. Anschließend wird die Transaktion *lokal festgeschrieben* und danach an die anderen Teilnehmer *weiterverteilt*. Diese Schritte und die damit verbundenen Problemstellungen

[2] Aus Gründen der Übersichtlichkeit sind nur die wichtigsten Klassen verkürzt dargestellt.

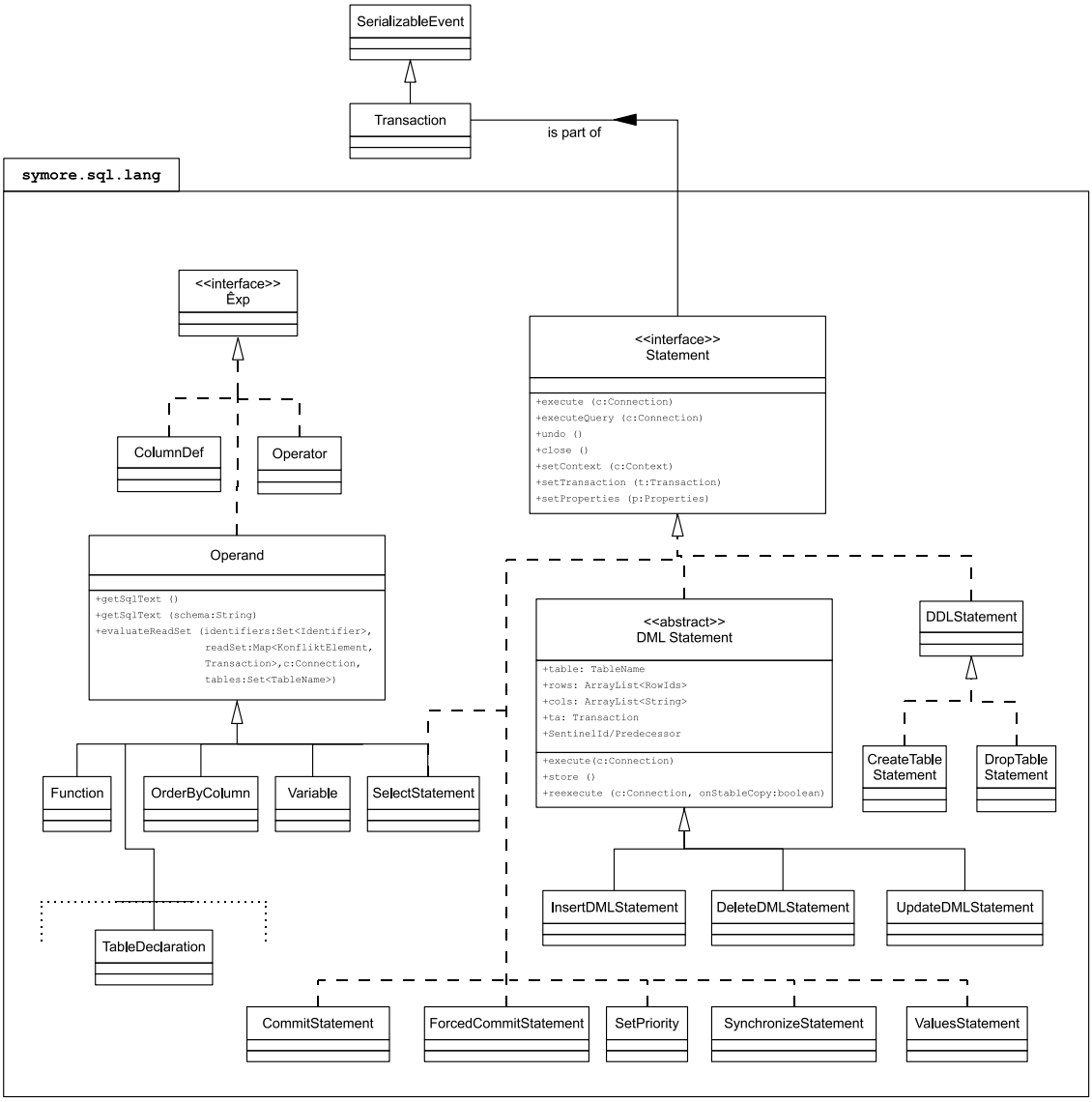


Abbildung 8.2: Klassendiagramm des Pakets `symore.sql.lang`

(Nebenläufigkeit, Aktualisierung der Datenbank) werden in den folgenden Abschnitten genauer erläutert. Wir behandeln an dieser Stelle nur SQL Anweisungen, da andere Anweisungen (SetPriority, Synchronize) Spezialfälle darstellen, die vom System individuell behandelt werden. Die verwendete SQL Grammatik ist eine vereinfachte Version der Grammatik der Open Source Datenbank *mckoi* [mck10] und wurde speziell an die Bedürfnisse von SYMORE angepasst.

8.2.1 Ausführung

Bei der erfolgreichen Ausführung einer Transaktion werden die einzelnen enthaltenen Statements ausgeführt, und für die Transaktion wird ein Knoten in den Vorgängergraph eingefügt. Um die verwendeten Konfliktelemente zu bestimmen, muss zunächst die Lese- und Schreibmenge der einzelnen Statements erfasst werden.

Die Erfassung der verwendeten Tabellen und Spalten kann direkt durch die entsprechenden Bezeichner in der SQL-Anweisung durchgeführt werden. Die Erfassung der gelesenen bzw. geschrieben Zeilen ist aufwändiger und benötigt die jeweiligen Primärschlüssel. Um diesen Vorgang zu vereinfachen, wird in SYMORE für jede Tabelle die Spalte `rowId` hinzugefügt, die einen künstlichen Schlüssel enthält. Dieser Schlüssel wird vom System automatisch vorgegeben und ist global eindeutig.[3]

Mit Hilfe der verwendeten Zeilen, Tabellen und Spalten können nun die entsprechenden Konfliktelemente bestimmt werden. Die Vereinigung aller jeweils geschriebenen und gelesenen Konfliktelemente der einzelnen Statements bildet dann die Lese- und Schreibmenge der Transaktion.

Zuletzt werden die einzelnen Statements auf der lokalen Datenbank ausgeführt. Ist die Ausführung fehlerfrei möglich, wird auf der lokalen Datenbank ein pre-Commit ausgeführt und für die Transaktion ein entsprechender Knoten in den Vorgängergraph eingefügt. Ist die Ausführung nicht möglich, so wird von der lokalen Datenbank ein Fehler gemeldet und die Transaktion wird abgebrochen. Dadurch wird auch kein entsprechender Knoten in den Vorgängergraphen eingefügt.

8.2.2 Verteilung

Ist die Transaktion lokal ausgeführt und in den Vorgängergraphen eingefügt worden, wird sie der Datenverteilungskomponente übergeben. Es wird sowohl die Transaktionen inklusive ihrer Lese- und Schreibmenge als auch der entsprechende Knoten des Vorgängergraphen inklusive seiner ausgehenden Kanten weitergereicht. Die SQL-Anweisungen der Transaktion werden dazu so verändert, dass sie auf den Datenbanken der anderen Replikationsteilnehmer auf die gleichen Zeilen zugreifen, damit keine Inkonsistenzen entstehen können. So wird z. B. verhindert, dass Bereichsanfragen bei unterschiedlichen Teilnehmern unterschiedliche Zeilenmengen liefern. Um die übertragene Datenmenge klein zu halten, werden die Syntaxbäume einer SQL-Anweisung nicht übertragen.

[3] Eine genaue Beschreibung findet sich in der Arbeit von Frank Bregulla [Bre06].

8.2.3 Behandlung der Nebenläufigkeit

Für die Verwaltung von nebenläufigen Ereignissen anderer Teilnehmer enthält jeder RM einen Eventmanager (siehe Abbildung 8.1). Dieser verwaltet zwei Ereignis-Warteschlangen, eine für die *Transaktionen* und eine für die *Commits*. Alle Ereignisse werden nach ihren Echtzeit-Zeitstempeln geordnet in die Warteschlangen eingefügt. Der Eventmanager verwaltet die lokalen und die entfernten Transaktionen in separaten Prozessen.

Wenn der Eventmanager ein entferntes Ereignis (Transaktion, Statusnachricht oder Commit) abarbeitet, muss er dafür sorgen, dass nebenläufig keine lokale Transaktion ausgeführt wird, da es sonst zu Dateninkonsistenzen kommen kann. Der Prozess zur Behandlung der entfernten Ereignisse fordert dazu beim Eventmanager eine Sperre an, so dass parallel keine lokalen Transaktionen ausgeführt werden. Je nach Ereignis wird bei dessen Abarbeitung die Konflikterkennung/-lösung durchgeführt und die Datenbanksicht aktualisiert.

8.2.4 Aktualisierung des festgeschriebenen Datenbankzustands

Wie bereits in Abschnitt 2.4 erwähnt, gibt es zwei unterschiedliche Sichten auf die lokale Datenbank eines Replikationsteilnehmers. Die eine Sicht entspricht dem Datenbankzustand nach der Ausführung aller aktuellen Transaktionen, die bereits lokal ausgeführt (pre-Commit) wurden. Dieser Zustand ist nur vorläufig und kann sich häufig ändern, da aufgrund von neuen entfernten Transaktionen bereits lokal ausgeführte Transaktionen abgebrochen werden können.

Die zweite Sicht entspricht dem Datenbankzustand nach der Ausführung aller Transaktionen, für die bereits ein globales Commit durchgeführt wurde. Dieser Datenbankzustand ist stabil und ändert sich erst, wenn ein neues globales Commit durchgeführt wird. Beide Sichten sind in SYMORE als materialisierte Datenbankzustände realisiert. Dazu wird bei der Initialisierung des Systems ein weiteres Schema angelegt. Ein Schema enthält dann den aktuellen Zustand und das andere den festgeschriebenen (stabilen) Zustand. Wird ein globales Commit durchgeführt, werden alle Transaktionen des aktuellen Commitintervalls auf dem festgeschriebenen Zustand ausgeführt.

Muss aufgrund von Konfliktlösungen der aktuelle Zustand geändert werden, wird dieser zunächst mit dem festgeschriebenen Zustand überschrieben. Anschließend werden alle aktivierten Transaktionen entsprechend der Ausführungsreihenfolge des Vorgängergraphen auf diesem Zustand ausgeführt, um den aktuellen Zustand zu erhalten.

Dieses Verfahren hat den Vorteil, dass der aktuelle Zustand sofort verfügbar ist und nicht bei jedem Zugriff erneut aufgebaut werden muss. Nur im Falle eines Konflikts muss mindestens eine Transaktion abgebrochen werden und somit der aktuelle Zustand neu erstellt werden. Durch dieses Verfahren verdoppelt sich jedoch auch der Speicherplatzbedarf der Datenbank. Diese aufgrund der einfachen Implementierung des Verfahrens entstehenden Probleme können jedoch durch entsprechende Methoden abgeschwächt werden. So kann z. B. mit Hilfe eines Copy-on-Write Verfahrens der Speicherplatzbedarf verringert werden, indem nur Kopien der veränderten Datenelemente ange-

legt werden. Auch der Neuaufbau des aktuellen Zustands kann durch eine feingranularere Aktualisierung effizienter gestaltet werden. Diese Problematik ist analog zur Durchführung eines „Redo“ auf einer materialisierten Datenbasis und wird in [KE06] weitergehend diskutiert.

8.3 Beispielapplikationen

In diesem Abschnitt wird das Replikationssystem SYMORE aus Sicht der Applikationsentwicklung und -anwendung betrachtet. Es wird untersucht, *wie transparent das Replikationssystem aus Sicht der Applikation ist*.

Dazu wurden zwei Beispielapplikationen implementiert, die SYMORE als Datenbankreplikationssystem verwenden. Die eine Applikation ist ein mobiles Wiki, in dem die Daten von den Nutzern parallel geändert werden können. Die andere Applikation ist ein mobiles Puzzlespiel, das ein gleichzeitiges „Wettpuzzeln“ mehrerer Spieler erlaubt.

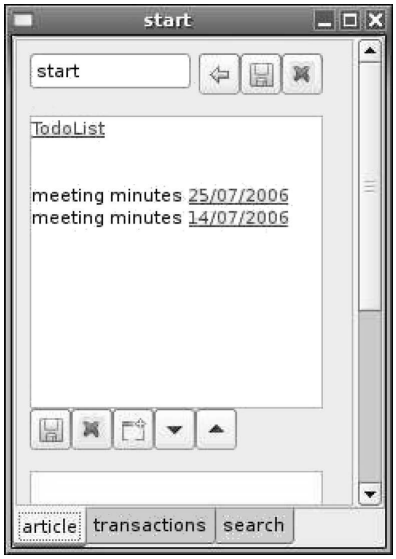
8.3.1 Mobiles Wiki

Das MOBILE-WIKI ist eine einfache Variante eines Wikis, mit dessen Hilfe Textartikel von mehreren Personen gleichzeitig bearbeitet werden können. Einsatzmöglichkeiten für diese Applikation sind z. B. das Campus- oder das Touristenszenario. Das MOBILE-WIKI dient dabei als Informationssystem, das von allen Teilnehmern ergänzt werden kann, aber in erster Linie lesend genutzt wird. Längere Unterbrechungszeiten stellen daher kein grundlegendes Problem dar.

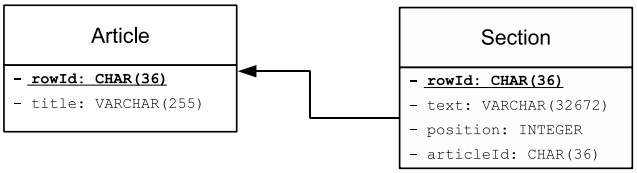
Das vorliegende Wiki orientiert sich stark an dem Desktop Notizbuch Tomboy [Gra10] und weist von der Verfahrensweise her Ähnlichkeiten zum Texteditor Google Docs [gdc10] auf, der ebenfalls das kollaborative Erstellen von Dokumenten ermöglicht. Der Inhalt des Wikis ist in Artikel aufgeteilt, die wiederum in mehrere Abschnitte unterteilt sind. Innerhalb des Textes können Verweise auf weitere Artikel erstellt werden, welche per Mausklick aufgerufen werden. Die Nutzer des Wikis können alle Artikel lesen, aber auch neue Artikel und Abschnitte erstellen oder bereits bestehende Abschnitte verändern. Werden gleiche Abschnitte kausal parallel geändert, so darf nur eine dieser Änderungen global gültig bleiben. Die anderen Änderungen müssen zurückgesetzt werden.

Die für mobile Geräte geeignete Applikation ist in Java (J2ME CDC) implementiert und wurde auf Laptops und PDAs (grafisch eingeschränkte Version) getestet. In Abbildung 8.3(a) ist das Hauptfenster der MOBILE-WIKI Applikation dargestellt. Zentral ist das Ansichts- und Bearbeitungsfenster. Im Gegensatz zu anderen Wiki Applikationen kann der Text ohne Aufruf einer speziellen Änderungssicht verändert werden. Im Fenster wird immer der aktuelle Artikel mit seinem Titel (links oben) angezeigt. Durch Aufrufen von Verweisen wird eine Historie angelegt, in der mit Hilfe des „Vor-“ oder „Zurück“ Buttons navigiert werden kann.

Das Schema der MOBILE-WIKI Applikation ist in Abbildung 8.3(b) dargestellt. Es besteht nur aus den zwei Tabellen `Article` und `Section`, welche die Artikel und die Abschnitte eines Arti-



(a) Screenshot der MOBILE-WIKI Applikation



(b) Datenbankschema der MOBILE-WIKI Applikation

Abbildung 8.3: MOBILE-WIKI Applikation

kels repräsentieren. Jeder Artikel besitzt eine durch SYMORE vergebene global eindeutige `rowId` und einen Titel. Ein Abschnitt (`Section`) besitzt ebenfalls eine `rowId`, einen Verweis auf seinen Artikel und eine Position, welche zur Bestimmung der Reihenfolge der Abschnitte verwendet wird. Der eigentliche Text des Abschnitts ist in der Spalte `text` gespeichert.

Um ein weiteres Verständnis für die Anwendung zu bekommen, werden die wichtigsten Anwendungsfälle (engl. use cases) im Folgenden aufgelistet:

- Artikel anlegen, löschen, aufrufen, speichern
- Titel eines Artikels ändern
- Abschnitt erstellen, löschen, ändern, speichern
- Reihenfolge von Abschnitten ändern
- Status der Transaktionen verfolgen

Die zentralen Anwendungsfälle sind das Aufrufen eines Artikels und das Ändern bzw. Erstellen und Speichern eines Abschnitts. Der Nutzer kann ebenfalls den Status seiner Transaktionen einsehen.

Entwurfsentscheidungen

An dieser Stelle werden Entwurfsentscheidungen diskutiert, die sich im Besonderen mit dem Zusammenspiel zwischen Applikation und Replikationssystem befassen.

Modellierung der Transaktionen Bei der Abbildung der Anwendungsfälle auf Transaktionen bestehen grundsätzlich zwei Möglichkeiten der Modellierung: Ein Lesen-Ändern-Schreiben Vorgang wird entweder als eine Transaktion aufgefasst oder die Anwendungsfälle werden jeweils als einzelne Transaktionen modelliert. Die erste Möglichkeit erzeugt aufgrund der enthaltenen Interaktion mit dem Benutzer länger andauernde Transaktionen, die in der momentanen Implementierung die gesamte lokale Datenbank für nicht lokale Transaktionen sperren würden. Aus diesem Grund wird die zweite Variante verwendet und der einzelne Anwendungsfall jeweils als eine Transaktion modelliert.

Wenn zwischen der lesenden und der schreibenden Transaktion eines Abschnitts eine von außen eingehende Transaktion auf demselben Abschnitt ausgeführt wurde, ergeben sich jedoch Probleme. Aus Sicht des Replikationssystems wird dann zwar die lokale Transaktion kausal hinter der von außen eingegangenen Transaktion eingefügt und es entsteht kein Konflikt. Aus der Sicht des Nutzers ist jedoch ein Konflikt aufgetreten, da der Abschnitt während der Bearbeitung von mindestens einem anderen Teilnehmer geändert wurde. Aus diesem Grund wird beim Speichern von der Applikation geprüft, ob sich der Datenbestand seit dem letzten Lesezugriff verändert hat. Ist dies der Fall, so wird der Nutzer informiert und darf eine der Änderungen (lokal, entfernt) auswählen. Der ausgewählte Inhalt wird dann in die Datenbank geschrieben. Für den Fall, dass mehr als zwei Teilnehmer den gleichen Abschnitt nebenläufig ändern, kann es passieren, dass mehrere Nutzer eine Änderung auswählen dürfen. Die zeitlich letzte Auswahl ist dann die gültige und somit können keine Inkonsistenzen in der Datenbank entstehen.

Wenn der gesamte Lesen-Ändern-Schreiben Vorgang als Transaktion behandelt würde, könnte die Prüfung durch die Applikation vermieden werden. Die Sicht des Nutzers würde dann mit der Sicht des Systems übereinstimmen.

Reihenfolge der Abschnitte Eine weitere Besonderheit der Applikation ist das Auftreten von logischen Inkonsistenzen, die jedoch keine Dateninkonsistenzen sind und dadurch nicht vom Replikationssystem erkannt werden können. Diese können bei der Reihenfolge der einzelnen Abschnitte eines Artikels entstehen. Würden z. B. zwei Teilnehmer kausal parallel hinter einem bestimmten Abschnitt einen neuen hinzufügen und dabei als Position des neuen Abschnitts jeweils den um eins inkrementierten `position` Wert des vorangegangenen Abschnitts verwenden, so hätten die zwei neuen Abschnitte dieselbe Position. Für das Replikationssystem stellt dies kein Problem dar, da nicht dieselben Datenelemente geändert wurden. Für die Applikation stellt sich jedoch das Problem, in welcher Reihenfolge die Abschnitte dargestellt werden sollen. Wie dieses Problem von der Applikation gelöst wird, hängt von den konkreten Anforderungen ab und kann deswegen nicht allgemein

gültig gelöst werden.

Ein sehr einfacher Lösungsansatz ist z. B., kausal parallel erstellte Artikel in beliebiger Reihenfolge darzustellen. Für die Implementierung ist jedoch eine striktere Variante verwendet worden. Diese setzt eine global eindeutige Reihenfolge der Abschnitte um, indem durch das kausal parallele Einfügen an der gleichen Stelle ein Konflikt verursacht wird. Damit der logische Konflikt einen Konflikt im Replikationssystem erzeugt, führt die Applikation innerhalb der Transaktion eine zusätzliche Pseudo-Schreiboperation auf dem vorangegangenen Abschnitt durch. Diese Operation verändert nichts, sondern dient nur dazu, beim parallelen Einfügen einen Konflikt zu verursachen. Fügen nun zwei Teilnehmer einen Abschnitt an der gleichen Stelle kausal parallel in einen Artikel ein, wird auf dem vorangehenden Artikel jeweils die Pseudo-Schreiboperation ausgeführt und somit vom Replikationssystem ein Konflikt erkannt.

Nachteil dieser Lösung ist jedoch, dass ein Abschnitt gelöscht wird, obwohl lediglich die Bestimmung seiner Position nicht eindeutig ist. Aus diesem Grund wäre ein sinnvoller Schritt bei dieser Art von Abbrüchen, die abgebrochene Transaktion erneut auf dem aktuellen Datenbankzustand auszuführen und ein eindeutiges Einfügen zu ermöglichen.

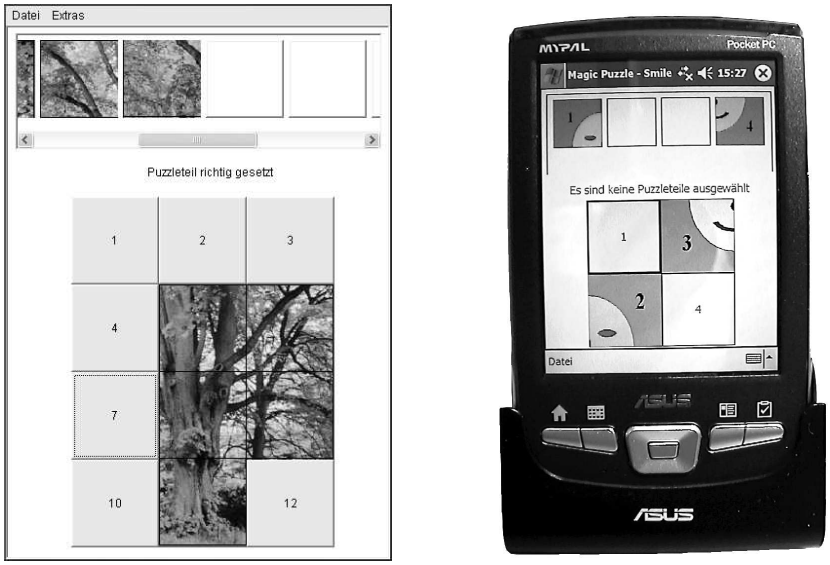
Fazit

Die Implementierung der MOBILE-WIKI Applikation hat gezeigt, dass das Datenbankreplikationssystem SYMORE für Anwendungen mit verteilten Daten einsetzbar ist. Das Replikationssystem nimmt dem Entwickler dabei den Großteil der Verwaltung der verteilten Daten ab. Es regelt die Datenverteilung und sorgt dafür, dass die Datenbank in einem konsistenten Zustand bleibt.

Trotzdem ist die Replikation für den Entwickler nicht vollständig transparent und so müssen teilweise Probleme berücksichtigt werden, die sich aufgrund der verteilten Datenänderungen ergeben. Ein Beispiel ist das beschriebene Problem der Artikelreihenfolge, das sich aufgrund des dezentralen optimistischen Replikationsansatzes ergibt. Hierbei muss der Entwickler Methoden erstellen, um Inkonsistenzen in der Applikationslogik zu verhindern. Die Problematik, dass sich ein Artikel während der Bearbeitung in der lokalen Datenbank verändern kann, existiert genauso bei zentralisierten Datenbanksystemen.

8.3.2 Mobiles Puzzlespiel

Das mobile Puzzlespiel MOBILE-PUZZLE ist eine einfache Spielapplikation, die es mehreren Teilnehmern erlaubt, gleichzeitig an einem Puzzle mitzuarbeiten. Im Gegensatz zur Wiki Applikation gibt es bei diesem Spiel eine hohe Anzahl an schnell aufeinanderfolgenden Schreiboperationen. Aus diesem Grund wird ein Szenario angenommen, in dem es nur sehr kurze Unterbrechungszeiten gibt. Im Idealfall befinden sich alle Teilnehmer in gegenseitiger Funkreichweite. Die Applikation und das Szenario wurden entsprechend gewählt, um das Replikationssystem auch unter zeitkritischen Bedingungen zu testen.



(a) Screenshot (b) PDA Version

Abbildung 8.4: MOBILE-PUZZLE Applikation

Ziel des Spiels ist es, schneller als die anderen Spieler die Puzzleteile an die richtige Stelle auf dem Spielfeld zu setzen. Ist ein Puzzleteil auf die korrekte Stelle gelegt worden, kann es nicht mehr bewegt werden. Ist das Puzzle beendet, gewinnt der Spieler, der die meisten Puzzleteile als Schnellster an die richtige Stelle gelegt hat. Konflikte können entstehen, wenn mehrere Teilnehmer ein Puzzleteil gleichzeitig setzen, bzw. wenn mehrere Puzzleteile gleichzeitig auf dieselbe Stelle des Spielfelds gesetzt werden. In diesem Fall darf nur einer der am Konflikt beteiligten Spielzüge durchgeführt werden, die anderen werden abgebrochen.

Die Applikation ist in Java (J2ME CDC) implementiert und für mobile Geräte geeignet und wurde auf Laptops und PDAs getestet. Das Hauptfenster der Puzzleapplikation und ein Foto der PDA Version ist in Abbildung 8.4 dargestellt. Die obere Leiste zeigt die Puzzleteile, die sich noch außerhalb des Puzzles befinden. Um diese auf ein bestimmtes Puzzlefeld zu setzen, klickt man das Puzzleteil und anschließend das Zielfeld an. Im Hauptbereich sieht man das Spielfeld mit dem aktuellen Zustand des Puzzles.

Datenbankschema und Anwendungsfälle

Das Schema der Puzzleapplikation ist in Abbildung 8.5 dargestellt. Es besteht aus den zwei Tabellen `Puzzlepiece` und `Position`. Die Tabelle `Puzzlepiece` beinhaltet die Puzzleteile, deren aktuelle Position und die Spielernummer des Spielers, der das Puzzleteil zuletzt bewegt hat. Die Spielfelder sind in der Tabelle `Position` gespeichert und werden durch ihre Koordinaten be-

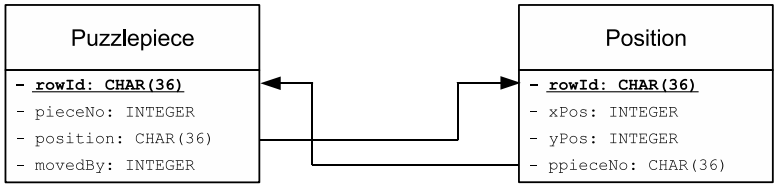


Abbildung 8.5: Datenbankschema der MOBILE-PUZZLE Applikation

stimmt. Zusätzlich verweist jedes Spielfeld auf ein ihm zugewiesenes Puzzleteil. Der Grund für diese redundante Modellierung wird weiter unten in den Entwurfsentscheidungen angegeben.

Für das mobile Puzzle gibt es im Spiel folgende relevante Anwendungsfälle:

- Spielfeld aktualisieren
- Puzzleteil auf die richtige/falsche Stelle setzen
- Puzzleteile tauschen

Das Spielfeld wird nach jeder Änderung aktualisiert. Dazu hat sich die Applikation als Beobachter angemeldet, der bei jeder Änderung (lokal oder von außen) der Datenbank informiert wird und dann den Lesevorgang anstößt. Da ein richtig gesetztes Puzzleteil nicht mehr bewegt werden darf, wird das Setzen eines Puzzleteils in zwei Anwendungsfälle aufgeteilt. Ebenso ist es möglich, zwei Puzzleteile zu vertauschen und dadurch bis zu zwei Puzzleteile auf die korrekte Position zu setzen.

Entwurfsentscheidungen

An dieser Stelle werden Entwurfsentscheidungen diskutiert, die sich auf die Replikationsproblematik beziehen.

Modellierung der Transaktionen Wie bei der MOBILE-WIKI Applikation sind die einzelnen Anwendungsfälle als separate Transaktionen modelliert. Bei der Aktualisierung des Spielfelds liest die zugehörige Transaktion immer alle Puzzleteile auf einmal. Das Auswählen des Puzzleteils stellt keine Transaktion auf der Datenbank dar, sondern wird nur von der Applikation behandelt. Durch diese Vorgehensweise kann es vorkommen, dass ein vom Spieler ausgewähltes Teil bereits von einem anderen Spieler gesetzt wurde. Wird diese entfernte Transaktion vor dem Setzen des Puzzleteils ausgeführt, so wird dem Spieler das Puzzleteil praktisch „aus der Hand genommen“ und auf das entsprechende Feld gesetzt. Im Gegensatz zur Wiki Applikation stellt dieses Verhalten aus Sicht des Nutzers kein Problem dar, sondern ist Teil des Spielprinzips. Das Setzen des Puzzleteils hingegen resultiert in einer Schreibtransaktion, die aus zwei Schreiboperationen besteht: Einerseits wird die Position des Puzzleteils verändert und andererseits wird dem Spielfeld das Puzzleteil zugeordnet.

Diese beiden Operationen müssen innerhalb einer Transaktion ausgeführt werden, da im Spiel sonst logische Inkonsistenzen auftreten können.

Die Spiellogik gibt im Falle eines Konflikts vor, dass der Spieler, der ein Puzzleteil zuerst gesetzt hat, den Konflikt gewinnt. Ebenso zählt aber für die Punktwertung, welcher Spieler ein Puzzleteil zuerst *korrekt* gesetzt hat. Aus diesem Grund muss bei einem Konflikt, bei dem nur ein Puzzleteil korrekt gesetzt wurde, die Transaktion des korrekt gesetzten Puzzleteils gewinnen.

Um dieses Verhalten mit Hilfe des Replikationssystems nachzubilden, wird für die Konfliktlösung die Priorität der Transaktion verwendet. Jeder Transaktion, die ein Puzzleteil auf eine falsche Position setzt, wird von der Applikation die Priorität 2 zugewiesen. Wird ein Puzzleteil auf das korrekte Feld gesetzt, so erhält es die Priorität 1 und gewinnt somit jeden Konflikt mit einer Transaktion, die den Wert 2 besitzt. Bei gleicher Priorität wird zusätzlich der Echtzeit-Zeitstempel bei der Konfliktlösung verwendet und die ältere Transaktion gewinnt, was ebenfalls die Spiellogik widerspiegelt.

Modellierung des Schemas Im nicht verteilten Fall wäre es bei der Definition des Schemas ausreichend, nur eine Referenz von `Puzzlepiece` auf `Position` zu setzen. Im verteilten Fall wird jedoch in diesem Schema vom Replikationssystem kein Konflikt erkannt, wenn zwei Puzzleteile auf dieselbe Stelle gesetzt werden, da zwei unterschiedliche Zeilen verändert werden. Um dieses Problem zu beheben, muss zusätzlich eine Referenz von `Position` auf `Puzzlepiece` gesetzt werden. Nun wird für diesen Fall parallel das gleiche Datenelement (`Position.ppieceNo`) verändert und dadurch ein Konflikt verursacht.

Fazit

Wie bei der Entwicklung der mobilen Wiki Applikation zeigt sich auch hier, dass sich der Entwicklungsaufwand durch die Verwendung des Replikationssystems reduziert. Zum Einen muss die Verteilung der Daten nicht berücksichtigt werden. Zum Anderen muss der Einfluss der Nebenläufigkeit auf die Konsistenz der Datenbank nicht berücksichtigt werden, weil das Replikationssystem dafür sorgt, dass die Datenbank in einem konsistenten Zustand bleibt. Da die Nebenläufigkeit ein Teil des Spielprinzips ist, kann ein Großteil der Spiellogik mit Hilfe des Replikationssystems umgesetzt werden. Im Gegensatz zu einer gleichartigen Anwendung, die nur mit lokalen Daten arbeitet, muss jedoch das Schema leicht angepasst werden, damit alle Konflikte in der Spiellogik auch vom Replikationssystem erkannt werden.

Das Puzzlespiel hat gezeigt, dass das Replikationssystem SYMORE auch für Applikationen mit zeitkritischen Anforderungen geeignet ist. Praktische Erfahrungen haben gezeigt, dass die Puzzleapplikation auch mit mehreren Teilnehmern (bis zu fünf) auf Laptops problemlos lauffähig ist. Auch bei der Verwendung von PDAs zeigt sich, dass das Replikationssystem trotz der geringeren Leistung der Geräte einsetzbar ist. Die grafische Ausgabe der Puzzleapplikation stellt jedoch ein Problem dar. Mit Hilfe eines Profilers konnte festgestellt werden, dass die grafische Ausgabe den

Großteil der Prozessorleistung des PDAs beansprucht. Aus diesem Grund führt dies bei mehr als zwei Teilnehmern dazu, dass die PDAs durch die häufigen Änderungen der grafischen Ausgabe schnell überlastet sind. Es kommt zu merklichen Unterbrechungen im Spielablauf. Eine zusätzliche Überprüfung des Problems mit einer textbasierten Version der Puzzleapplikation wies bei einer Anzahl von fünf Teilnehmern keine Einschränkungen auf.

8.4 Zusammenfassung

In diesem Kapitel haben wir das Datenbankreplikationssystem SYMORE vorgestellt. SYMORE ist ein in Java (J2ME CDC) implementierter Prototyp, der als Proof-of-Concept unserer Replikationsmethoden dient. Das System stellt eine JDBC konforme Schnittstelle zur Verfügung, mit der Applikationen auf die replizierte Datenbank zugreifen können. Auch replikationsspezifische Befehle werden über diese Schnittstelle zur Verfügung gestellt. Dazu wurde die SQL Grammatik um entsprechende Befehle erweitert.

Intern verwendet SYMORE für die Verwaltung der Daten eine Apache Derby Datenbank, auf die ebenfalls über eine JDBC Schnittstelle zugegriffen wird. Die Implementierung des Systems ermöglicht dadurch auch einen Austausch der Datenbank. Damit eine Applikation über Änderungen informiert wird, kann sich diese als Beobachter anmelden und wird durch ein Event über die Änderungen in Kenntnis gesetzt.

Um zu testen, inwieweit das Replikationssystem die Entwicklungsarbeit von Anwendungen erleichtert, die auf verteilten Daten arbeiten, wurden zwei Beispielapplikationen entwickelt. Für beide Applikationen (MOBILE-WIKI, MOBILE-PUZZLE) ergab sich eine Vereinfachung der Entwicklungsarbeit, da die Datenverteilung und Konsistenzhaltung durch SYMORE größtenteils transparent gehalten wurde. Trotzdem mussten bestimmte Aspekte an die Applikationslogik angepasst werden. Es zeigt sich bei den mobilen Anwendungen ebenfalls, dass eine transaktionale Replikation, wie sie von unserem System angeboten wird, notwendig ist, um die Anwendungsfälle korrekt abzubilden.

Teil III

Bewertung

Kapitel 9

Konfliktanalyse

In diesem Kapitel stellen wir eine analytische Betrachtung der von uns verwendeten optimistischen Replikationsmethoden vor[1]. Ziel dieser Analyse ist es, abzuschätzen, inwieweit sich der Einsatz der bisher vorgestellten Replikationsmethoden für bestimmte Szenarien eignet. Als Indikator für diese Abschätzung wird die Konfliktwahrscheinlichkeit verwendet, d.h. die Wahrscheinlichkeit dafür, dass eine bestimmte Transaktion mit mindestens einer anderen Transaktion in Konflikt steht.

Zu Beginn dieses Kapitels besprechen wir die relevanten Einfluss- und Zielgrößen (Abschnitt 9.1). Danach wird in Abschnitt 9.2 die Herleitung der Gesamtformel erläutert und anschließend werden die Ergebnisse diskutiert (Abschnitt 9.3). In Abschnitt 9.4 wird unser Ansatz mit einer von Gray et al. [GHOS96] durchgeführten Analyse verglichen.

9.1 Einfluss- und Zielgrößen

In diesem Abschnitt werden zunächst die relevanten Einfluss- und Zielgrößen besprochen. Eine Übersicht ist in Abbildung 9.1 dargestellt.

Konfliktwahrscheinlichkeit Die Konfliktwahrscheinlichkeit ist die von der Konfliktanalyse betrachtete Zielgröße. Sie gibt für eine Transaktion die Wahrscheinlichkeit an, dass diese mit mindestens einer anderen Transaktion in Konflikt steht. Die Konfliktwahrscheinlichkeit dient als Indikator für die Eignung der vorgestellten optimistischen Replikationsmethoden in gegebenen Szenarien. Eine große Anzahl von Konflikten führt dazu, dass die Replikation aus Sicht des Anwenders praktisch unbrauchbar wird. Hohe Konfliktwahrscheinlichkeiten zeigen somit an, dass optimistische Replikationsmethoden für das betreffende Szenario nicht geeignet sind.

Datenverteilungsintervall Das Datenverteilungsintervall gibt den durchschnittlich benötigten Zeitraum an, bis die Daten aller Teilnehmer im Netz so verteilt sind, dass alle Teilnehmer synchronisiert sind (siehe Abschnitt 2.1.4). Bei der Konfliktanalyse ist dieses Intervall eine Zufallsvariable.

[1] Die Konfliktanalyse wurde in Zusammenarbeit mit Frank Bregulla im Rahmen seiner Diplomarbeit entwickelt [Bre06].

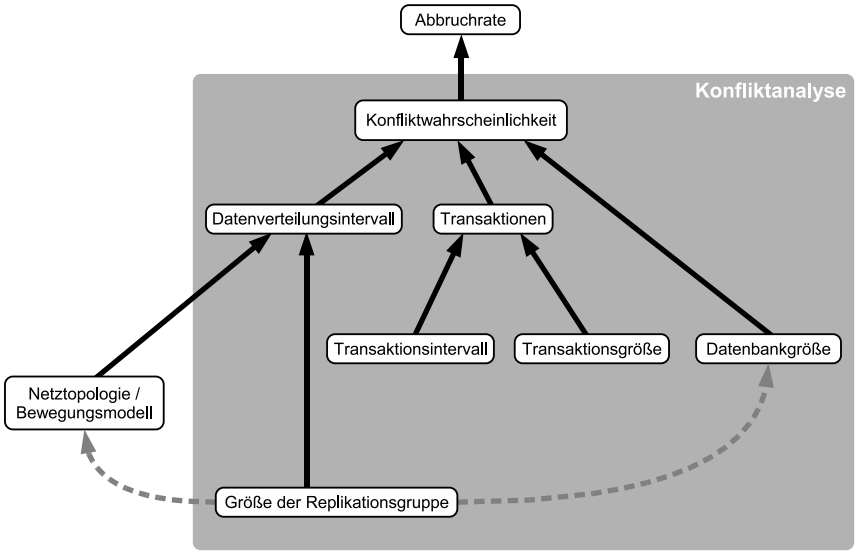


Abbildung 9.1: Einfluss- und Zielgrößen der Konfliktanalyse. Die Pfeile stellen die Abhängigkeiten dar.

Transaktionen Aus Sicht der Konfliktanalyse werden Transaktionen durch ihre Größe und ihre Häufigkeit (Transaktionsintervall) bestimmt. Zur Vereinfachung werden bei der Analyse nur nichtkommutative Schreiboperationen betrachtet.

Transaktionsintervall Das Transaktionsintervall ist das Intervall zwischen der Ausführung zweier aufeinanderfolgender Transaktionen eines Teilnehmers. Bei der Konfliktanalyse ist dieses Intervall eine Zufallsvariable. Dieses Intervall hängt stark von dem zugrunde liegenden Szenario ab. In einem Spielszenario, wie es für die beschriebene Puzzleapplikation (siehe Abschnitt 8.3.2) gegeben ist, liegen die Werte im Sekundenbereich. In einem Katastrophenszenario, in dem Helfer in einem Katastrophengebiet die Daten über Versorgungsdepots, Verletzte, etc. aktualisieren, sind diese Intervalle hingegen wesentlich länger.

Transaktionsgröße Dieses Parameter gibt an, auf wie viele Datenelemente eine Transaktion zugreift. Größere Transaktionen haben eine höhere Konfliktwahrscheinlichkeit zur Folge. Zwei kausal parallel durchgeführte Transaktionen stehen miteinander in Konflikt, wenn sie mindestens auf ein Datenelement gemeinsam schreibend zugreifen. Im Rahmen dieser Arbeit werden kleine Transaktionen mit einer Maximalgröße von fünf betrachtet.

Datenbankgröße Die Datenbankgröße gibt die Anzahl der enthaltenen Datenelemente an und beeinflusst die Konfliktwahrscheinlichkeit direkt. Je größer die Datenbank, desto geringer die Wahr-

scheinlichkeit, dass zwei Transaktionen nebenläufig auf das gleiche Datenelement zugreifen. Die Datenelemente werden in der Konfliktanalyse zufällig und gleichverteilt ausgewählt. Im Extremfall ergibt sich bei einer Datenbank mit nur einem Element für zwei nebenläufige Schreibtransaktionen immer ein Konflikt.

Die in dieser Arbeit betrachteten Datenbanken sind als integrierter Bestandteil einer mobilen Anwendung vorgesehen. Aus diesem Grund wird für die mobilen Datenbanken eine Maximalgröße von 2000 Datenelementen angenommen.

Netztopologie/Bewegungsmodell Die Netztopologie ist eine Einflussgröße, die das Datenverteilungsintervall direkt beeinflusst. Während Teilnehmer in einem Festnetz (LAN, WAN) häufig über eine gute Konnektivität verfügen und Daten somit schnell verteilt werden können, dauert die Datenverteilung in MANETs aufgrund längerer Unterbrechungszeiträume häufig länger. Diese Unterbrechungen sind in erster Linie auf die Mobilität der Knoten zurückzuführen. Um die Mobilität zu modellieren, werden Bewegungsmodelle verwendet, wie wir sie bereits in Kapitel 4 beschrieben haben.

Größe der Replikationsgruppe Die Größe der Replikationsgruppe ist eine weitere Einflussgröße, die sich auf die Konfliktwahrscheinlichkeit eines Szenarios auswirkt. Befinden sich in einer Replikationsgruppe mehr Teilnehmer, werden auch mehr Transaktionen ausgeführt, und die Konfliktwahrscheinlichkeit erhöht sich. Die Größe der Replikationsgruppe hat, im Gegensatz zu Festnetzen, ebenfalls Einfluss auf die Netztopologie eines MANETs. Eine größere Replikationsgruppe führt häufig zu einer höheren Knotendichte, wodurch die Konnektivität des Netzes erhöht wird. Dadurch wird die Datenverteilung beschleunigt und die Konfliktwahrscheinlichkeit verringert sich.

Der Einfluss der Replikationsgruppe auf die Netztopologie ist zu komplex, um analytisch betrachtet zu werden, da hierfür die Wahrscheinlichkeiten aller Aufenthaltsorte für sämtliche mobilen Knoten zu einem gegebenen Zeitpunkt bestimmt werden müssten (siehe [Tsc06]).

Aus diesem Grund betrachten wir diesen Aspekt in Kapitel 10, in dem experimentelle Untersuchungen mit dem Replikationssystem durchgeführt werden. Die zuvor beschriebenen Replikationsmethoden sind für kleine bis mittelgroße Replikationsgruppen konzipiert. Deshalb werden im Rahmen dieser Arbeit in erster Linie Replikationsgruppen mit maximal 100 Teilnehmern betrachtet.

Abbruchrate Die Abbruchrate ist eine Zielgröße und gibt den Anteil der Transaktionen an, die abgebrochen werden müssen. Nicht alle der in Konflikt stehenden Transaktionen müssen abgebrochen werden, da es für jeden Konflikt eine gewinnende Transaktion gibt. Je nach Anzahl der am Konflikt beteiligten Transaktionen n und den Abhängigkeiten innerhalb dieser Konfliktgruppe (siehe Abschnitt 7.2.2), reicht die Anzahl der Transaktionsabbrüche bei nicht-kommutativen Konflikten von 1 bis $n - 1$.

Die Bestimmung der Abbruchrate hängt stark von dem verwendeten Konfliktlösungsverfahren ab und ist bei größeren Konfliktgruppen sehr komplex: Zunächst müssen dafür alle Konfliktgruppen ermittelt werden, welche die gleiche Anzahl an Abbrüchen zur Folge haben. Im Vorgängergraphen sind diese Konfliktgruppen Teile des gesamten Graphen. Somit muss für die Bestimmung der Abbrüche festgestellt werden, ob die Graphen der Konfliktgruppe isomorph zueinander sind, d. h., ob sie den gleichen Aufbau besitzen. Zusätzlich zum Aufbau der Konfliktgruppen müssen bestimmte Teilreihenfolgen der Transaktionen berücksichtigt werden, da diese Teilreihenfolgen die Anzahl der Abbrüche beeinflussen. Um abschließend die Abbruchrate zu berechnen, müssen die Wahrscheinlichkeiten für das Auftreten der jeweiligen Konfliktgruppen berechnet werden.

Aufgrund der Komplexität dieses Problems wird die Abbruchrate in der Konfliktanalyse nicht betrachtet. In Szenarien mit einer geringen Konfliktwahrscheinlichkeit ist jedoch anzunehmen, dass die Abbruchrate bei ca. 50% der in Konflikt stehenden Transaktionen liegt. Da nur wenige Konflikte entstehen, ist es sehr wahrscheinlich, dass die Mehrheit dieser Konflikte jeweils nur zwischen zwei Transaktionen besteht. In diesem Fall wird immer genau eine Transaktion abgebrochen, wodurch sich die oben genannte Abbruchrate ergibt.

Anwendungsbereich Bis auf die Netztopologie und die Abbruchrate werden alle in Abbildung 9.1 dargestellten Größen in der Konfliktanalyse direkt berücksichtigt. Wie bereits gesagt kann die Abbruchrate nicht genau bestimmt werden. Trotzdem muss bei der Diskussion der resultierenden Konfliktwahrscheinlichkeiten berücksichtigt werden, dass die Anzahl der tatsächlich abgebrochenen Transaktionen kleiner ist, als die Anzahl der in Konflikt stehenden Transaktionen.

Die Netztopologie eines MANETs ist ebenfalls zu komplex (siehe oben), um analytisch betrachtet zu werden. Somit wird die Netztopologie durch das Datenverteilungsintervall repräsentiert. Es stellt sich dabei die Frage, inwieweit diese Vereinfachungen die Genauigkeit der Konfliktanalyse beeinflusst. In Kapitel 10 werden deshalb die Ergebnisse der experimentellen Untersuchung mit denen der Konfliktanalyse verglichen.

9.2 Herleitung

In diesem Abschnitt wird die Gesamtformel für die Konfliktanalyse schrittweise hergeleitet. Eine Orientierung über alle relevanten Parameter und deren Beschreibung gibt die folgende Tabelle 9.1.

9.2.1 Basisformel

Um die grundlegende Idee der Konfliktanalyse zu veranschaulichen, beginnen wir mit einem einfachen Szenario mit zwei Teilnehmern und einer Datenbankgröße von $e = 1$. Die Transaktionen bestehen jeweils aus nur einer Schreiboperation. Somit resultiert jede nebenläufige Ausführung zweier Transaktionen in einem Konflikt. Wenn also zwei Transaktionen von unterschiedlichen Teilnehmern

Parameter	Beschreibung
$\frac{1}{\lambda}$	Erwartungswert der Exponentialverteilung $f(x;\lambda) = \lambda e^{-\lambda x}$
$D \sim \text{Exp}(\lambda_D)$	Zufallsvariable, welche die Intervalllänge des Datenverteilungsintervalls repräsentiert; definiert durch eine exponentielle Verteilungsfunktion mit Parameter λ_D
$T \sim \text{Exp}(\lambda_T)$	Zufallsvariable, welche die Intervalllänge zwischen zwei aufeinanderfolgenden Transaktionen eines Teilnehmers repräsentiert; definiert durch eine exponentielle Verteilungsfunktion mit Parameter λ_T
$T' \sim \text{Exp}(\lambda_{T'})$	Zufallsvariable, welche die Intervalllänge zwischen zwei aufeinanderfolgenden und potentiell in Konflikt stehenden Transaktionen unterschiedlicher Teilnehmer repräsentiert; definiert durch eine exponentielle Verteilungsfunktion mit Parameter $\lambda_{T'}$
$E(D) = \frac{1}{\lambda_D}$	Erwartungswert für die Intervalllänge zwischen zwei Synchronisationspunkten
$E(T) = \frac{1}{\lambda_T}$	Erwartungswert für die Länge des Intervalls T
$E(T') = \frac{1}{\lambda_{T'}}$	Erwartungswert für die Länge des Intervalls T'
o	Anzahl der Schreiboperationen jeder Transaktion (Transaktionsgröße)
e	Anzahl der Datenelemente der Datenbank (Datenbankgröße)
s	Anzahl der Replikationsteilnehmer (Größe der Replikationsgruppe)

Tabelle 9.1: Überblick der relevanten Parameter der Konfliktanalyse

lokal ausgeführt wurden, ohne dass dazwischen eine Datenverteilung zwischen den Teilnehmern stattgefunden hat, dann stehen die Transaktionen miteinander in Konflikt.

In dem für die Konfliktanalyse verwendeten Modell wird im Gegensatz zum System angenommen, dass nach dem Ablauf des Datenverteilungsintervalls alle Teilnehmer miteinander die notwendigen Transaktionen ausgetauscht haben und synchronisiert sind. Die Zeitpunkte, die die Datenverteilungsintervalle begrenzen, werden deshalb als **Synchronisationspunkte** bezeichnet. Die Zwischenschritte der Datenverteilung zwischen einzelnen Teilnehmern werden im Gegensatz zum System nicht betrachtet. Zu einem Zeitpunkt innerhalb des Datenverteilungsintervalls wird angenommen, dass die Teilnehmer bis zum letzten Synchronisationspunkt synchronisiert sind. Eine Veranschaulichung eines Konfliktfalls ist in Abbildung 9.2 dargestellt.

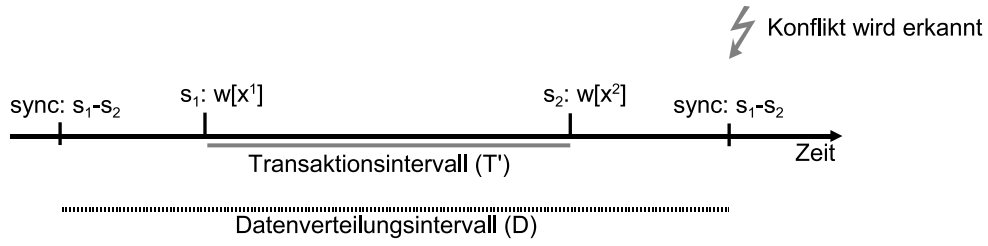


Abbildung 9.2: Einfaches Konfliktbeispiel

Zunächst wurden die Datenänderungen der beiden Teilnehmer verteilt, so dass die beiden Teilnehmer synchronisiert sind (bezeichnet mit *sync:* $s_1 - s_2$). Nach einer gewissen Zeit führt s_1 lokal

eine Transaktion mit einer Schreiboperation auf dem Datenelement x aus. Der Teilnehmer s_2 führt etwas später ebenfalls eine lokale Transaktion auf dem Element x durch.[2] Danach kommt es zwischen den beiden Teilnehmern erneut zu einer Verbindung. Nach dem Abgleich wird ein Konflikt festgestellt. Hätten sich die beiden Teilnehmer per Datenverteilung zwischen der Ausführung der beiden Transaktionen synchronisiert, wäre der Konflikt vermieden worden.

In der Abbildung sind ebenfalls die Längen des Transaktionsintervalls und des Datenverteilungsintervalls dargestellt. Das Transaktionsintervall ist deutlich kleiner als das Datenverteilungsintervall und somit konnte ein Konflikt entstehen. Bei dem Vergleich der Längen muss auch der Versatz der beiden Intervalle (Abstand der Intervallanfänge) berücksichtigt werden. Trotzdem lässt sich erkennen, dass das Größenverhältnis dieser beiden Intervalle zueinander einen grundlegenden Faktor zur Bestimmung der Konfliktwahrscheinlichkeit darstellt: Szenarien mit kurzen Transaktionsintervallen und langen Datenverteilungsintervallen resultieren in einer höheren Konfliktwahrscheinlichkeit; Szenarien mit umgekehrten Längenverhältnissen resultieren in einer niedrigeren Konfliktwahrscheinlichkeit.

Für die Verteilung der Transaktionsinitiierung über die Zeit wird eine Exponentialverteilung angenommen, da über die tatsächliche Verteilung nichts bekannt ist, außer, dass beliebig große Werte möglich sind. Ebenso wird für die Datenverteilungsdauer eine Exponentialverteilung angenommen. Somit wird die Länge des Transaktions- und des Datenverteilungsintervalls jeweils durch eine Exponentialverteilung bestimmt. Aufgrund der Gedächtnislosigkeit der Exponentialverteilung muss der Versatz der Intervalle nicht berücksichtigt werden. Dies führt zum ersten Schritt der Herleitung der Basisformel, in der die Längen der beiden Intervalle (Transaktion, Datenverteilung) verglichen werden.

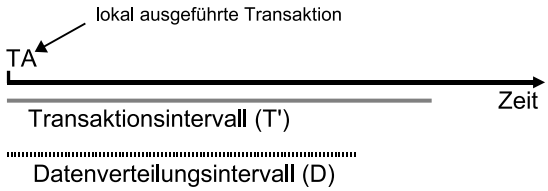


Abbildung 9.3: Vergleich des Transaktions- mit dem Datenverteilungsintervall

In Abbildung 9.3 ist ein Beispiel gegeben. Die Länge des Intervalls zwischen zwei aufeinanderfolgenden Transaktionen verschiedener Teilnehmer, deren Write-Sets sich schneiden, ist die Zufallsvariable T'. Diese Variable dient der Vereinfachung und fasst mehrere Einflussgrößen zusammen. In den nächsten Schritten wird dies differenziert, und die Variable wird durch die entsprechenden Einflussgrößen ersetzt.

Die Länge des Datenverteilungsintervalls ist die Zufallsvariable D. Wie bereits erläutert, kann

[2] Mit x^1 und x^2 werden jeweils die lokalen Versionen des Datenelements bezeichnet (siehe Abschnitt 2.4.1)

der Versatz der Intervalle vernachlässigt werden, so dass wir nun die Wahrscheinlichkeit bestimmen können, dass die Zufallsvariable T' größer als D ist: $P(T' > D)$.

Um diesen Wert zu berechnen, verwenden wir die Dichtefunktion $f(x) = \lambda_{T'} e^{-\lambda_{T'} x}$ von T' und die Dichtefunktion $g(x) = \lambda_D e^{-\lambda_D x}$ von D. Mit Hilfe dieser Funktionen bestimmen wir für alle möglichen Intervalllängen die Wahrscheinlichkeit, dass das Transaktionsintervall länger als das Datenverteilungsintervall ist. Dazu müssen wir die Integrale der Dichtefunktionen miteinander multiplizieren:

$$\begin{aligned}
P(T' > D) &= \int_0^\infty g(y) \left(\int_y^\infty f(x) dx \right) dy \\
&= \int_0^\infty \lambda_D \, e^{-\lambda_D \, y} \left(\int_y^\infty \lambda_{T'} \, e^{-\lambda_{T'} \, x} \, dx \right) dy \\
&= \int_0^\infty \lambda_D \, e^{-\lambda_D \, y} (1 - F(y)) \, dy \\
&= \int_0^\infty \lambda_D \, e^{-\lambda_D \, y} (1 - (1 - e^{-\lambda_{T'} \, y})) \, dy \\
&= \int_0^\infty \lambda_D \, e^{-\lambda_D \, y} \, e^{-\lambda_{T'} \, y} \, dy \\
&= \int_0^\infty \lambda_D \, e^{y(-\lambda_D - \lambda_{T'})} \, dy \\
&= \left[-\frac{\lambda_D \, e^{y(-\lambda_D - \lambda_{T'})}}{\lambda_D + \lambda_{T'}} \right]_0^\infty \\
&= 0 - \left(-\frac{\lambda_D}{\lambda_D + \lambda_{T'}} \right) = \frac{\lambda_D}{\lambda_D + \lambda_{T'}}
\end{aligned}$$

Nun setzen wir in die Gleichung die Erwartungswerte $E(D) = \frac{1}{\lambda_D}$ und $E(T') = \frac{1}{\lambda_{T'}}$ ein, welche die Länge der beiden Intervalle angeben. Die resultierende Gleichung liefert nun die Wahrscheinlichkeit, dass für die gegebenen Erwartungswerte das Transaktionsintervall länger als das Datenverteilungsintervall ist:

$$P(T' > D) = \frac{E(T')}{E(T') + E(D)} \tag{9.1}$$

Diese Gleichung liefert noch nicht die Wahrscheinlichkeit für eine konfliktfreie Transaktion, da bisher nur die Intervalle (zeitlich) *folgender* Transaktionen betrachtet wurden (siehe Abbildung 9.3). Um die Wahrscheinlichkeit für eine konfliktfreie Transaktion zu bekommen, müssen auch die Intervalle *vorangehender* Transaktionen betrachtet werden (siehe Abbildung 9.4).

Da die beiden Ereignisse (Konflikt mit folgender/vorangehender Transaktion) stochastisch unabhängig sind, führt dies zu folgender Gleichung:

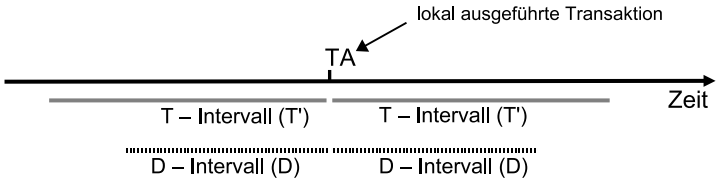


Abbildung 9.4: Berücksichtigung der Transaktions- und Datenverteilungsintervalle vor und nach einer Transaktion

$$P(\overline{Konflikt}) = P(T' > D)^2 = \left(\frac{E(T')}{E(T') + E(D)}\right)^2$$

Wir können nun die Konfliktwahrscheinlichkeit für ein einfaches Szenario mit zwei Teilnehmern und den Erwartungswerten für die Länge des Transaktions- und des Datenverteilungsintervalls berechnen:

$$P(Konflikt) = 1 - \left(\frac{E(T')}{E(T') + E(D)}\right)^2 \tag{9.2}$$

In den nächsten Abschnitten werden wir diese Berechnung verfeinern, indem wir den Wert von $E(T')$ mit Hilfe weiterer Einflussgrößen detaillierter bestimmen.

9.2.2 Berücksichtigung der Teilnehmerzahl

Im Gegensatz zu der zuvor beschriebenen Basisformel verwenden wir nun den Erwartungswert für die Länge des Transaktionsintervalls zwischen zwei aufeinanderfolgenden Transaktionen *eines* Teilnehmers. Dieser wird mit $E(T)$ bezeichnet und ist der Erwartungswert einer Exponentialverteilung.

Bei einer Replikationsgruppe mit s Teilnehmern werden s mal so viele Transaktionen ausgeführt wie bei einem Teilnehmer. Somit ist $\frac{E(T)}{s}$ der Erwartungswert für die Intervalllänge zwischen zwei aufeinanderfolgenden Transaktionen von beliebigen Teilnehmern. Dies beinhaltet auch den Fall zwei aufeinanderfolgender Transaktionen *eines* Teilnehmers.

Um die Konfliktwahrscheinlichkeit zu bestimmen, wird jedoch der Erwartungswert der Länge des Intervalls zwischen zwei aufeinanderfolgenden Transaktionen benötigt, die potentiell miteinander in Konflikt stehen, die also von *unterschiedlichen* Teilnehmern stammen. Diesen Erwartungswert erhalten wir, indem wir die zu erwartende Anzahl der Transaktionen zwischen zwei aufeinanderfolgenden Transaktionen unterschiedlicher Teilnehmer bestimmen und diese mit der Intervalllänge $\frac{E(T)}{s}$ multiplizieren.

Zunächst berechnen wir die Wahrscheinlichkeit, dass die n-te Transaktion TA_n, die auf TA_0 folgt, die erste ist, die von einem anderen Teilnehmer stammt als von dem TA_0 initiierenden Teilnehmer. Die Zufallsvariable I gibt die Stelle dieser Transaktion an. Die Wahrscheinlichkeit $P(I = n)$ wird

berechnet, indem der Anteil aller positiven Fälle (TA_n stammt von einem anderen Teilnehmer) an der Gesamtzahl aller möglichen Permutationen bestimmt wird. Die Berechnung von $P(I = n)$ wird anhand von Abbildung 9.5 erläutert.

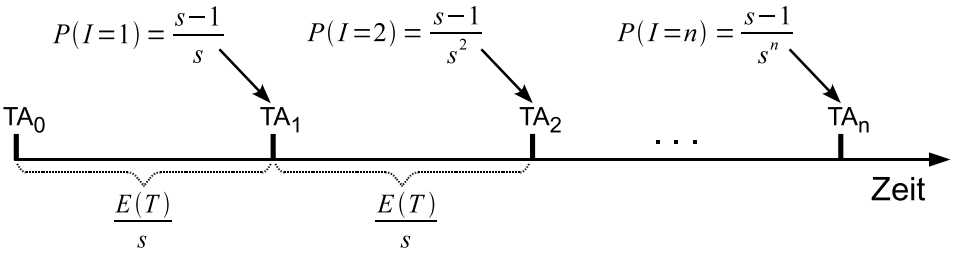


Abbildung 9.5: Bestimmung der Wahrscheinlichkeit, dass die n-te Transaktion von einem anderen Teilnehmer als von dem TA_0 initiierenden Teilnehmer stammt

Im ersten Schritt betrachten wir die erste auf TA_0 folgende Transaktion TA_1. Mit s Teilnehmern gibt es $s - 1$ Möglichkeiten, dass TA_1 von einem anderen Teilnehmer stammt. Insgesamt gibt es s unterschiedliche Möglichkeiten für TA_1, da ebenfalls die Möglichkeit berücksichtigt werden muss, dass die Transaktion von demselben Teilnehmer stammt.

Im nächsten Schritt betrachten wir nur die Wahrscheinlichkeit für die zweite auf TA_0 folgende Transaktion TA_2. Erneut gibt es $s - 1$ Möglichkeiten, dass die Transaktion TA_2 von einem anderen Teilnehmer stammt, als von dem Teilnehmer, der TA_0 initiiert hat. Das liegt daran, dass die Transaktion TA_1 vom gleichen Teilnehmer wie TA_0 stammen muss. Ansonsten wäre TA_1 bereits die nächste nachfolgende Transaktion, die von einem anderen Teilnehmer initiiert wurde. Die Gesamtzahl der Möglichkeiten liegt bei s^2, da beide folgenden Transaktionen (TA_1 und TA_2) berücksichtigt werden müssen.

Allgemein liefert dies für die n-te Transaktion eine Wahrscheinlichkeit von $\frac{s-1}{s^n}$. Um nun den Erwartungswert der Position der zweiten Transaktion zu bestimmen, müssen alle Möglichkeiten entsprechend ihrer Wahrscheinlichkeit berücksichtigt und mit ihrer Position multipliziert werden. Diesen Erwartungswert liefert die folgende Gleichung:

$$E(I) = \sum_{n=1}^{\infty} n \cdot P(I_n) = \sum_{n=1}^{\infty} n \cdot \frac{s-1}{s^n} = \frac{s}{s-1} \tag{9.3}$$

Dadurch, dass T ebenfalls exponentialverteilt ist (siehe oben), kann der endgültige Wert $E(T')$ einfach bestimmt werden, indem $E(I)$ (erwartete Position) mit der Intervalllänge zwischen zwei beliebigen Transaktionen $\frac{E(T)}{s}$ multipliziert wird:

$$E(T') = E(I) \cdot \frac{E(T)}{s} = \frac{E(T)}{s-1} \tag{9.4}$$

9.2.3 Berücksichtigung der Datenbank- und Transaktionsgröße

Bisher wurde nur der Fall betrachtet, dass die Datenbank aus einem Element besteht und jede Transaktion genau eine Operation besitzt, die auf dieses Element schreibend zugreift. In diesem Abschnitt wollen wir nun die Analyse erweitern und auch unterschiedliche Datenbank- und Transaktionsgrößen berücksichtigen. Während eine größere Datenbank die Konfliktwahrscheinlichkeit verringert, sorgen größere Transaktionen für einen Anstieg der Konfliktwahrscheinlichkeit. Der Einfluss der Transaktionsgröße auf die Konfliktwahrscheinlichkeit ist wesentlich stärker, da sich die Write-Sets zweier Transaktionen nur schneiden müssen, um einen Konflikt zu verursachen. Um die Lesbarkeit zu erleichtern, sprechen wir bei sich schneidenden Write-Sets davon, dass sich die zwei entsprechenden Transaktionen *überlappen*.

Um den Erwartungswert der Position einer zweiten Transaktion zu berechnen, die sich mit einer anderen ausgewählten Transaktion überlappt, verfahren wir analog zum vorigen Abschnitt. Wir betrachten wiederum eine ausgewählte Transaktion TA_0. Die Anzahl der Datenelemente der Datenbank wird mit e bezeichnet. Es wird angenommen, dass alle lokalen Datenbanken der Teilnehmer dieselbe Anzahl besitzen. Die Anzahl der Schreiboperationen einer Transaktion auf unterschiedlichen Elementen wird mit o bezeichnet. Wir nehmen an, dass alle Transaktionen dieselbe Anzahl an Operationen besitzen. Somit ergeben sich für eine einzelne Transaktion $\binom{e}{o}$ Kombinationsmöglichkeiten für die Auswahl der Datenelemente.

Wir bestimmen nun die Wahrscheinlichkeit, dass die n-te auf Transaktion TA_0 folgende Transaktion TA_n, die erste Transaktion ist, die sich mit TA_0 überlappt. Dieses Ereignis wird durch die Zufallsvariable I'_n repräsentiert. Da es für eine Transaktion $\binom{e}{o}$ Kombinationen gibt, existieren für TA_n insgesamt $\binom{e-o}{o}$ Kombinationen sich *nicht* mit TA_0 zu überlappen. Somit sind $\binom{e}{o} - \binom{e-o}{o}$ die möglichen Kombinationen für eine Überlappung der beiden Transaktionen TA_0 und TA_n. Damit TA_n die erste Transaktion ist, die sich mit TA_0 überlappt, dürfen sich alle vorangegangenen Transaktionen nicht mit TA_0 überlappen. Die Anzahl der Kombinationen für dieses Ereignis ist $\binom{e-o}{o}^{n-1}$. Somit ergibt sich für das Ereignis, dass TA_n die erste Transaktion ist, die sich mit TA_0 überlappt, folgende Anzahl von Kombinationen:

$$\left(\binom{e}{o} - \binom{e-o}{o}\right) \cdot \binom{e-o}{o}^{n-1}$$

Um nun die Wahrscheinlichkeit für das Ereignis I'_n zu bekommen, müssen wir den obigen Term durch die Anzahl der gesamten Kombinationsmöglichkeiten dividieren. Dies führt zu folgender Gleichung:

$$P(I'_n) = \frac{\left(\binom{e}{o} - \binom{e-o}{o}\right) \cdot \binom{e-o}{o}^{n-1}}{\binom{e}{o}^{n}} \tag{9.5}$$

Daraus kann der Erwartungswert für die Position bestimmt werden, indem die einzelnen Wahr-

scheinlichkeitswerte aufsummiert und mit ihrer entsprechenden Position multipliziert werden:

$$E(I') = \sum_{n=1}^{\infty} n \cdot P(I'_n) = \sum_{n=1}^{\infty} n \cdot \frac{\left(\binom{e}{o} - \binom{e-o}{o}\right) \cdot \binom{e-o}{o}^{n-1}}{\binom{e}{o}^n} = \frac{\binom{e}{o}}{\binom{e}{o} - \binom{e-o}{o}} \tag{9.6}$$

Um den Erwartungswert für die Länge des Intervalls zu erhalten, muss der Erwartungswert $E(I')$ wiederum mit der Intervalllänge $\frac{E(T)}{s}$ multipliziert werden. Dieser Schritt wird im nächsten Abschnitt bei der Herleitung der Gesamtformel durchgeführt.

9.2.4 Gesamtformel

Im ersten Schritt haben wir eine Basisformel für die Konfliktanalyse präsentiert, die weder die Anzahl der Teilnehmer, die Datenbankgröße noch die Transaktionsgröße berücksichtigt hat. Die Einflussgrößen wurden alle in dem Erwartungswert $E(T')$ zusammengefasst. Danach haben wir schrittweise gezeigt, wie die obigen Einflussgrößen direkt berücksichtigt werden können. $E(T')$ kann somit durch die entsprechenden Terme ersetzt werden (siehe unten).

Die Gesamtformel basiert auf dem Erwartungswert $E(T)$, der die Länge des Intervalls zwischen zwei aufeinanderfolgenden Transaktionen *eines* Teilnehmers repräsentiert. Die einzelnen Umformungsschritte werden in der folgenden Gleichung gezeigt:

$$\begin{aligned}
P(Konflikt) &= 1 - \left(\frac{E(T')}{E(T') + E(D)}\right)^2 = 1 - \left(1 - \frac{E(D)}{E(D) + E(T')}\right)^2 \\
&= 1 - \left(1 - \frac{E(D)}{E(D) + \frac{E(T)}{s} \cdot \frac{s}{s-1} \cdot \frac{\binom{e}{o}}{\binom{e}{o} - \binom{e-o}{o}}}\right)^2 \\
&= 1 - \left(1 - \frac{E(D)}{E(D) + \frac{E(T)}{s-1} \cdot \frac{\binom{e}{o}}{\binom{e}{o} - \binom{e-o}{o}}}\right)^2
\end{aligned}$$

9.3 Diskussion

Die in den vorangegangenen Abschnitten beschriebene Konfliktanalyse stellt die Eigenschaften realer Szenarien zum Teil vereinfacht dar. Dies liegt darin begründet, dass die Topologie des MANETs und das Benutzerverhalten durch Verteilungsfunktionen modelliert werden. Da die tatsächlichen Verteilungen unbekannt sind, werden für das Datenverteilungs- und das Transaktionsintervall Ex-

ponentialverteilungen verwendet, da sie am ehesten der zu erwartenden Verteilung entsprechen und darüberhinaus einfach zu handhaben sind.

Die Verteilung der Operationen auf den Datenelementen ist ebenfalls nicht bekannt, da sie sehr von der Applikation und dem verwendeten Szenario abhängig ist. In der Konfliktanalyse wird eine Gleichverteilung angenommen. In der Realität kann es jedoch vorkommen, dass die Nutzer den Datenzugriff zum Teil außerhalb des Systems regeln, was in einer geringeren Konfliktwahrscheinlichkeit resultiert. Ebenso sind jedoch auch Hotspot Datenelemente denkbar, auf die wesentlich häufiger zugegriffen wird, wodurch sich die Konfliktwahrscheinlichkeit wiederum erhöht.

Die hier vorgestellte Konfliktanalyse stellt einen allgemeinen Ansatz dar, weshalb für jedes Szenario die gleichen Verteilungen verwendet werden. Inwieweit die beschriebenen Annahmen die Genauigkeit der Analyse beeinflussen wird in Kapitel 10 betrachtet. Dort werden die Ergebnisse der Analyse mit den Ergebnissen der experimentellen Untersuchung verglichen.

Trotz der Vernachlässigung der zuvor genannten Aspekte stellt die Konfliktanalyse einen guten Indikator für die Eignung unserer Replikationsmethoden für bestimmte Szenarien dar. Die Analyse bezieht sich nicht nur auf die von uns verwendeten Replikationsmethoden, sondern auf die Verwendung von optimistischer Replikation im Allgemeinen. Es wird dabei nur die kausal parallele Ausführung von Transaktionen auf denselben Daten betrachtet und keine speziellen Methoden zur Konfliktlösung und -erkennung. Um ein besseres Verständnis für die Konfliktanalyse zu bekommen, werden im Folgenden die Auswirkungen der einzelnen Parameter auf die Konfliktwahrscheinlichkeit anhand von Beispielwerten betrachtet.

Basisformel

Zunächst wird erneut die Basisformel der Konfliktanalyse betrachtet, da diese nur die grundlegenden Einflussgrößen verwendet: den Erwartungswert für die Länge des Intervalls zwischen zwei potentiell in Konflikt stehenden Transaktionen $E(T')$ und den Erwartungswert für die Länge des Datenverteilungsintervalls $E(D)$.

In Abbildung 9.6 ist auf der x-Achse das Größenverhältnis von $E(D)$ und $E(T')$ und auf der y-Achse die Konfliktwahrscheinlichkeit aufgetragen. Die Werte der x-Achse reichen von $\frac{E(D)}{E(T')} = \frac{1}{512}$ bis 64 und sind logarithmisch skaliert. Wenn beide Werte identisch sind, ergibt sich eine Konfliktwahrscheinlichkeit von 75%, was bereits sehr hoch ist. Wahrscheinlichkeiten von weniger als 10% ergeben sich erst ab einem Verhältnis von $\frac{E(D)}{E(T')} = \frac{1}{16}$. Im mittleren Bereich des Graphen ist die Steigung höher als am Rand. Daran ist zu erkennen, dass sich bei geringen Unterschieden der Intervalllängen Änderungen des Größenverhältnisses stärker auswirken.

In der Gesamtformel werden zusätzlich die folgenden Einflussgrößen berücksichtigt: Datenbankgröße, Anzahl der Teilnehmer (Größe der Replikationsgruppe) und die Größe der Transaktionen. Wir betrachten im Folgenden, wie sich diese Einflussgrößen auf die Konfliktwahrscheinlichkeit auswirken.

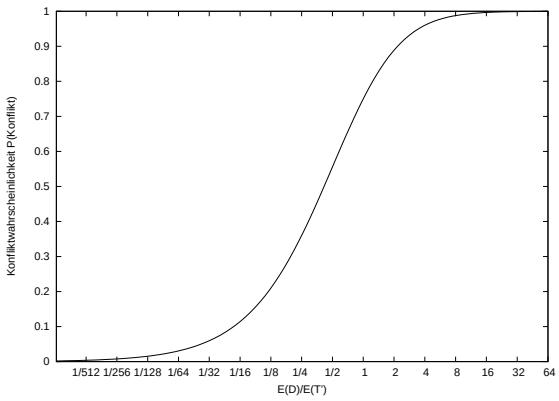


Abbildung 9.6: Einfluss des Größenverhältnisses zwischen $E(D)$ und $E(T')$

Transaktionsgröße

Wie bereits erwähnt, betrachten wir nur Szenarien, in denen kleine Transaktionen zu erwarten sind, die aus maximal fünf Operationen bestehen. Anhand von Abbildung 9.7 soll der Einfluss der Transaktionsgröße auf die Konfliktwahrscheinlichkeit verdeutlicht werden.

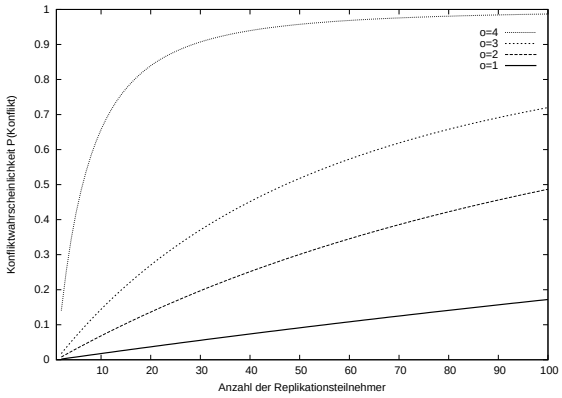


Abbildung 9.7: Einfluss der Transaktionsgröße

Auf der x-Achse ist die Anzahl der Teilnehmer und auf der y-Achse wiederum die Konfliktwahrscheinlichkeit aufgetragen. Es gilt $E(D) = E(T)$, und die lokalen Datenbanken bestehen jeweils aus 1000 Datenelementen.

Wie erwartet, wirkt sich die Größe der Transaktionen stark auf die Konfliktwahrscheinlichkeit aus. Während Transaktionsgrößen von 1 bis 3 bei kleinen Replikationsgruppen noch geringe Werte liefern, zeigen sich bei der Verwendung von Transaktionen mit 4 Schreiboperationen selbst bei kleinen Replikationsgruppen für die Konfliktwahrscheinlichkeit hohe Werte. Geringere Konfliktwahrschein-

lichkeiten ergeben sich nur in Szenarien mit größeren Datenbanken oder einem kleineren $\frac{E(D)}{E(T)}$ Wert.

Größe der Datenbank und der Replikationsgruppe

Anhand der Abbildungen 9.8(a), 9.8(b) und 9.8(c) soll der Einfluss der Datenbank- und der Replikationsgruppengröße auf die Konfliktwahrscheinlichkeit verdeutlicht werden. Die gezeigten Szenarien unterscheiden sich in ihren $\frac{E(D)}{E(T)}$ Werten ($\frac{E(D)}{E(T)} = 1$, bzw. $\frac{1}{3}$) und in ihrer Transaktionsgröße (1 bzw. 2). Die unterschiedliche Anzahl der Datenelemente und die Größe der Replikationsgruppe (Anzahl der Teilnehmer) sind auf der x- und y-Achse aufgetragen. Die z-Achse zeigt die Konfliktwahrscheinlichkeit an. Alle Diagramme enthalten ebenfalls zwei Grenzlinien, welche die Konfliktwahrscheinlichkeiten von 10% und 20% markieren und als Orientierung dienen. Diese Werte markieren die Grenze für Szenarien, in denen der Einsatz von optimistischer Replikation aus unserer Sicht noch sinnvoll erscheint. Konfliktwahrscheinlichkeiten über 20% resultieren in zu vielen Transaktionsabbrüchen und machen somit die Replikation praktisch unbrauchbar. Werte unter 20% resultieren in Abbruchraten von etwa 10% oder weniger (siehe „Abbruchrate“ S. 141), was für ein mobiles Szenario akzeptabel ist.

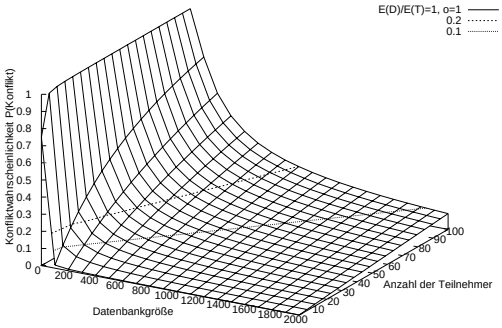
Im ersten Diagramm sind die Ergebnisse verschiedener Szenarien mit einer Transaktionsgröße von $o = 1$ und $\frac{E(D)}{E(T)} = 1$ abgebildet. Mit einer Datenbankgröße von $e = 2000$ Elementen liegt die Konfliktwahrscheinlichkeit für alle betrachteten Gruppengrößen unterhalb von 10%. Bis zu einer Datenbankgröße von $e = 800$ Elementen liegt die Konfliktwahrscheinlichkeit bei 100 Teilnehmern noch unterhalb von 20%.

Im zweiten Diagramm sind unterschiedliche Szenarien mit einem längeren Transaktionsintervall abgebildet ($\frac{E(D)}{E(T)} = \frac{1}{3}$). Somit ergeben sich auch insgesamt geringere Werte für die Konfliktwahrscheinlichkeit. Sogar Szenarien mit Datenbankgrößen von weniger als $e = 300$ Datenelementen zeigen akzeptable Ergebnisse für alle Gruppengrößen.

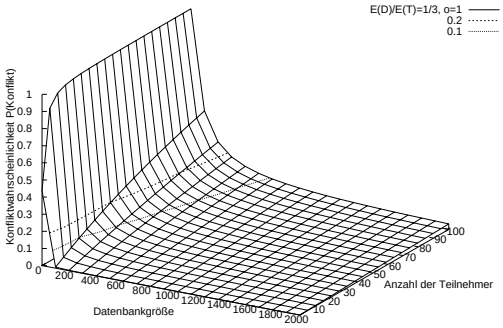
Im letzten Diagramm wurde die Transaktionsgröße auf $o = 2$ erhöht und $\frac{E(D)}{E(T)} = \frac{1}{3}$ belassen. Wie schon vorher beobachtet, zeigt sich erneut der starke Einfluss der Transaktionsgröße auf die Konfliktwahrscheinlichkeit. Obwohl $\frac{E(D)}{E(T)} = \frac{1}{3}$ gilt, sind die Konfliktwahrscheinlichkeiten wesentlich höher als die des Diagramms aus Abbildung 9.8(a) (mit $\frac{E(D)}{E(T)} = 1$). Im Hinblick auf die Gruppengröße zeigt sich, dass Konfliktwahrscheinlichkeiten unter 10% nur für Gruppengrößen mit maximal 80 Teilnehmern erreicht werden.

Betrachtet man alle drei Diagramme, kann man feststellen, dass kleine Datenbankgrößen die Konfliktrate drastisch erhöhen. Eine Datenbankgröße von 100 erlaubt nur kleine Replikationsgruppen mit maximal 30 Teilnehmern, um akzeptable Konfliktwahrscheinlichkeiten zu erreichen.

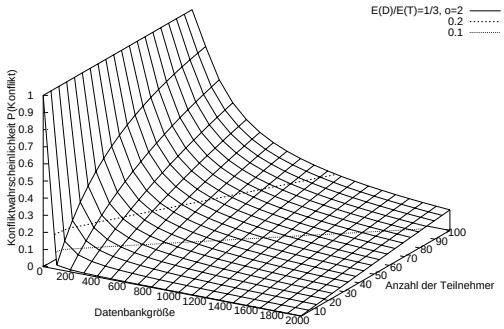
Partielle Replikation Bei der partiellen Replikation ist der Datenbestand im Gegensatz zur vollständigen Replikation unterteilt und es arbeitet jeweils nur ein Teil der Replikationsgruppe an einer Partition. Will man die Konfliktwahrscheinlichkeit für ein Szenario mit partieller Replikation be-



(a) $o = 1$ (Transaktionsgröße) und $\frac{E(D)}{E(T)} = 1$



(b) $\frac{E(D)}{E(T)} = \frac{1}{3}$



(c) $o = 2$ (Transaktionsgröße) und $\frac{E(D)}{E(T)} = \frac{1}{3}$

Abbildung 9.8: Einfluss der Datenbank- und Gruppengröße

rechnen, bei denen die einzelnen Partitionen disjunkt zueinander sind, so müssen zunächst die Konfliktwahrscheinlichkeiten für die einzelnen Subszenarien mit dem jeweiligen Datenbestand und der jeweiligen Teilnehmeranzahl berechnet werden. Um die Konfliktwahrscheinlichkeit für das gesamte Szenario zu erhalten, muss dann der Mittelwert aus den einzelnen Subszenarien gebildet werden.

Ist die Partitionierung des Datenbestandes proportional zur der jeweiligen Partitionierung der Teilnehmer, so ergibt sich für die partielle Replikation dieselbe Konfliktwahrscheinlichkeit wie bei einer vollständigen Replikation des gesamten Datenbestandes mit allen Teilnehmern der Subszenarien. Diese Eigenschaft ist deutlich an dem linearen Verlauf der zwei Grenzlinien in Abbildung 9.8 zu erkennen: Eine proportionale Veränderung der Gruppengröße und der Datenbankgröße bewirkt keine Veränderung der Konfliktwahrscheinlichkeit. So ergibt sich in Abbildung 9.8(a) für 70 Teilnehmer bei einer Datenbankgröße von 1300 Elementen dieselbe Konfliktwahrscheinlichkeit für 35 Teilnehmer bei einer Datenbankgröße von 650 Elementen. Bei einer proportionalen Partitionierung ergibt

sich somit als Mittelwert der Subszenarien dieselbe Konfliktwahrscheinlichkeit wie bei einer vollständigen Replikation.

In der vorliegenden Diskussion der Konfliktanalyse haben wir nur allgemeine Beispiele gezeigt, ohne auf spezielle Szenarien (Touristen-, Campus-, Katastrophenszenario, etc.) einzugehen, die am Anfang der Arbeit erwähnt wurden. Diese Betrachtung der Konfliktanalyse wird im Rahmen von Kapitel 10 durchgeführt, wo die genannten Szenarien experimentell untersucht werden und die Ergebnisse mit denen der Konfliktanalyse verglichen werden.

9.4 Vergleich mit der Analyse von Gray et al.

In diesem Abschnitt vergleichen wir unsere Konfliktanalyse mit der Analyse, die von Gray et al. in [GHOS96] durchgeführt wurde, da sie einen ähnlichen Ansatz verfolgt. Das Hauptziel dieser Analyse ist es, das Skalierbarkeitsproblem von optimistischer Multimaster Replikation aufzuzeigen. Für diesen Zweck ist eine einfache Abschätzung ausreichend. Aus diesem Grund wurden verschiedene Werte bei der Herleitung der Berechnungsformel zum Teil nur abgeschätzt (siehe [GHOS96]). Alle relevanten Parameter der Analyse von Gray et al. sind in Tabelle 9.2 beschrieben.

Parameter	**Beschreibung**
TPS	Pro Sekunde initiierte Transaktionen (pro Teilnehmer)
$t = \frac{1}{TPS}$	Länge des Transaktionsintervalls (zwischen zwei aufeinanderfolgenden Transaktionen eines Teilnehmers)
d	Länge des Datenverteilungsintervalls
s	Anzahl der Teilnehmer (Größe der Replikationsgruppe)
e	Anzahl der Datenlemente der Datenbank (Datenbankgröße)
o	Anzahl der Update-/Schreiboperationen einer Transaktion (Transaktionsgröße)

Tabelle 9.2: Parameter des Modells von Gray et al.

Im Gegensatz zu unserer Analyse werden keine Verteilungsfunktionen für die Länge des Transaktions- und des Datenverteilungsintervalls angenommen, sondern feste Werte verwendet. Ebenso wie bei dem zugrunde liegenden Modell unserer Analyse synchronisiert sich ein Teilnehmer zu einem bestimmten Zeitpunkt mit allen anderen Teilnehmern. Die Länge des Transaktionsintervalls ist indirekt durch die Anzahl der Transaktionen pro Sekunde (TPS) gegeben und kann durch $t = \frac{1}{TPS}$ bestimmt werden.

Im ersten Schritt der Herleitung wird die Anzahl aller ausgehenden Update-/Schreiboperationen u_{out} eines einzelnen Teilnehmers abgeschätzt. Der Wert von u_{out} gibt die Anzahl aller Update-/Schreiboperationen an, die von einem Teilnehmer initiiert werden, während er nicht mit den anderen

Teilnehmern verbunden ist:

$$u_{out} \approx d \cdot TPS \cdot o = \frac{d}{t} \cdot o$$

Analog können alle von den anderen Teilnehmern eingehenden Update-/Schreiboperationen für einen Teilnehmer abgeschätzt werden:

$$u_{in} \approx (s-1) \cdot d \cdot TPS \cdot o = (s-1) \cdot \frac{d}{t} \cdot o$$

Somit ergibt sich die Wahrscheinlichkeit, dass eine der eingehenden Operationen mit einer der ausgehenden Transaktionen in Konflikt[3] steht aus folgender Abschätzung:

$$P(Konflikt_{Teiln}) \approx \frac{u_{in} \cdot u_{out}}{e} \approx \frac{s \cdot (\frac{d}{t} \cdot o)^2}{e} \tag{9.7}$$

Diese Formel gibt die Wahrscheinlichkeit an, dass eine Operation eines Teilnehmers mit der eines anderen Teilnehmers in Konflikt steht. Die *Konfliktwahrscheinlichkeit für die gesamte Replikationsgruppe* liefert die folgende Abschätzung:

$$P(Konflikt_{Gruppe}) \approx P(Konflikt_{Teiln}) \cdot \frac{s}{d} \approx \frac{d \cdot (\frac{1}{t} \cdot o \cdot s)^2}{e} \tag{9.8}$$

Im Gegensatz zu unserer Konfliktanalyse wird hierbei die Konfliktwahrscheinlichkeit für die gesamte Replikationsgruppe (9.8) bzw. für einen Teilnehmer (9.7) anstatt für eine Transaktion berechnet. Weiterhin ist die Analyse nur approximiert und entspricht keiner Wahrscheinlichkeitsverteilung, da der Wertebereich über das Intervall $[0, 1]$ hinausgeht. Aufgrund anderer Annahmen ist dies in der Analyse von Gray et al. nicht problematisch.

Direkter Vergleich Um zu sehen, wie stark sich die beiden Analysen in einem bestimmten Szenario unterscheiden, führen wir einen direkten Vergleich durch. Beide Analysen können nur in bestimmten Fällen direkt miteinander verglichen werden. Dies liegt an dem oben beschriebenen Unterschied, dass mit der Gleichung von Gray et al. nur die Konfliktwahrscheinlichkeit pro Teilnehmer und nicht pro Transaktion bestimmt werden kann. Aus diesem Grund nehmen wir für das Transaktions- und das Datenverteilungsintervall eine Länge von 1 an. In diesem Fall initiiert ein Teilnehmer während seiner Unverbundenheit mit den anderen Teilnehmern genau eine Transaktion. Somit stimmt die Konfliktwahrscheinlichkeit pro Teilnehmer mit der Konfliktwahrscheinlichkeit pro Transaktion überein. Um diesen Wert zu bestimmen, verwenden wir die Gleichung 9.7. Die Ergebnisse der beiden Analysen sind in Abbildung 9.9 dargestellt.

[3] In [GHOS96] werden diese als Kollisionen bezeichnet.

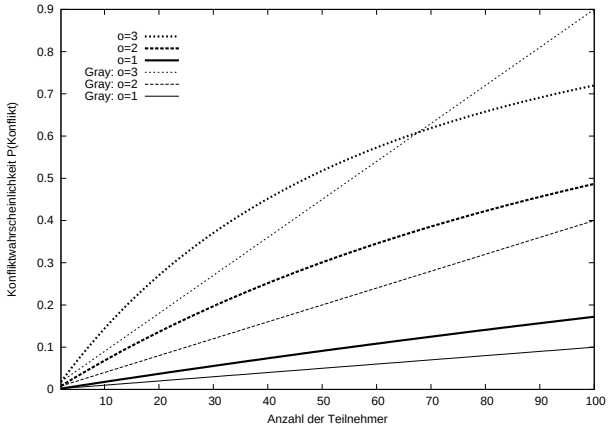


Abbildung 9.9: Vergleich mit der Konfliktanalyse von Gray et al.

In der Abbildung werden die Ergebnisse der beiden Analysen für drei unterschiedliche Transaktionsgrößen (1, 2 und 3) gezeigt. Wie bereits gesagt, gilt $t = 1$ und $d = 1$. Die Datenbankgröße ist $e = 1000$. Betrachtet man die Graphen, stellt man bei dem Graphen der Funktion von Gray et al. einen linearen Verlauf und bei den Graphen unserer Funktion ein abnehmendes Wachstum fest. In diesem Beispiel liefert unsere Analyse in den meisten Fällen höhere Werte. In dem Bereich mit höheren Wahrscheinlichkeitswerten ergeben sich jedoch für die Analyse von Gray et al. höhere Werte. Insgesamt ergibt sich in den meisten untersuchten Fällen zwischen den einzelnen Werten unserer Analyse und den Werten der Analyse von Gray et al. ein Unterschied von ungefähr $5 - 10$ Prozentpunkten.

Dieser Unterschied kann vernachlässigt werden, wenn nur die Skalierbarkeit abgeschätzt werden soll. Für eine genauere Bestimmung der Konfliktwahrscheinlichkeit für bestimmte Szenarien, wie sie in unserem Fall benötigt wird, ist die Analyse von Gray et al. jedoch nicht ausreichend.

9.5 Zusammenfassung

In diesem Kapitel haben wir eine Konfliktanalyse vorgestellt, welche die zu erwartende Konfliktwahrscheinlichkeit der Transaktionen für ein gegebenes Szenario bestimmt. Diese Analyse dient dazu, die Eignung der von uns verwendeten optimistischen Replikationsmethoden für bestimmte Szenarien zu prüfen.

Die grundlegende Idee der Konfliktanalyse ist der Vergleich der Datenverteilungsdauer mit der Zeit, die zwischen der Ausführung der Transaktionen vergeht. Diese Einflussgrößen werden in der Analyse durch zwei Intervalle (Transaktionsintervall, Datenverteilungsintervall) repräsentiert. Die Analyse berücksichtigt zusätzlich die Größe der Datenbank, die Transaktionsgröße und die Anzahl der Teilnehmer der Replikationsgruppe.

Die Ergebnisse der berechneten Beispielszenarien zeigen, dass sich die Transaktionsgröße am stärksten auf die Konfliktwahrscheinlichkeit auswirkt. Betrachtet man die Datenbankgröße und die Anzahl der Teilnehmer, stellt sich heraus, dass die Mehrheit der Szenarien von bis zu 100 Teilnehmern, Datenbankgrößen bis 2000 Elementen und mit gleich großem Transaktions- und Datenverteilungsintervall in niedrigen Konfliktraten resultiert (< 10%).

Wir haben unsere Analyse mit einem ähnlichen Ansatz von Gray et al. verglichen. Im Gegensatz zu unserer Analyse ist es dabei ausreichend, die Berechnungen teilweise abzuschätzen. Dadurch ergeben sich in einem direkten Vergleich anhand eines Beispielszenarios in den berechneten Konfliktwahrscheinlichkeiten Unterschiede von bis zu 10 Prozentpunkten. Ebenso wird die Konfliktwahrscheinlichkeit in der Analyse von Gray et al. pro Replikationsgruppe bzw. Teilnehmer bestimmt und nicht, wie für unsere Bewertung benötigt, pro Transaktion.

Kapitel 10

Evaluation

Nachdem in den vorherigen Kapiteln die von uns vorgestellten Replikationsmethoden einzeln betrachtet und bewertet wurden, wird in diesem Kapitel eine abschließende Gesamtbewertung durchgeführt. Anhand einer experimentellen Untersuchung wird das Verhalten unseres Replikationssystems SYMORE in verschiedenen MANET-Szenarien getestet.

Die Ziele dieser Untersuchung werden im ersten Abschnitt des Kapitels dargelegt. In Abschnitt 10.2 werden die Einfluss- und Messgrößen der Untersuchung erläutert und anschließend wird in Abschnitt 10.3 der Aufbau der Experimente beschrieben. Die Ergebnisse der unterschiedlichen Untersuchungsreihen werden in den drei Abschnitten 10.4, 10.5 und 10.6 dargestellt und diskutiert.

10.1 Ziele der Evaluation

Die in dieser Arbeit vorgestellten Replikationsmethoden sind nicht in beliebigen Szenarien einsetzbar. *Im Rahmen der Evaluation wird untersucht, für welche MANET-Szenarien sich die Replikationsmethoden eignen.* Die Untersuchungen werden anhand des Replikationssystems SYMORE durchgeführt. Das Hauptziel der Evaluation ist die Analyse der Konfliktwahrscheinlichkeit, da die Nutzbarkeit des Replikationssystems von der Konfliktwahrscheinlichkeit abhängig ist (siehe Abschnitt 9.1). Weil die Konfliktwahrscheinlichkeit eines Szenarios mit Hilfe experimenteller Untersuchungen nicht beliebig genau bestimmt werden kann, wird für die beobachteten Resultate der Begriff **Konfliktrate** verwendet. Die Konfliktrate gibt die relative Anzahl der in Konflikt stehenden Transaktionen für ein untersuchtes Szenario an. Im Idealfall stimmt die gemessene Konfliktrate mit der Konfliktwahrscheinlichkeit überein.

Im Einzelnen werden im Rahmen der Evaluation folgende Aspekte untersucht:

Skalierbarkeit Anhand der resultierenden Konfliktrate wird untersucht, wie sich das Replikationssystem SYMORE verhält, wenn bestimmte Einflussgrößen eines Szenarios, wie z. B. die Anzahl der Teilnehmer erhöht wird. Die Untersuchungsergebnisse lassen ebenfalls Schlüsse auf syntaktische Konflikterkennung im Allgemeinen zu, da die syntaktische Konflikterkennung von SYMORE

wie bei anderen gleichartigen Replikationssystemen nur auf der Erkennung der kausalen Nebenläufigkeit von Operationen basiert (siehe Kapitel 3).

Konfliktanalyse In Kapitel 9 wurde ein analytischer Ansatz zur Bestimmung der resultierenden Konfliktrate eines Szenarios vorgestellt. Wie bereits erwähnt, vernachlässigt diese Konfliktanalyse Aspekte der Datenverteilung und der Netztopologie, so dass für die Berechnungsergebnisse Ungenauigkeiten zu erwarten sind. Im Rahmen der experimentellen Untersuchung werden deshalb die Untersuchungsergebnisse mit den Ergebnissen der Konfliktanalyse verglichen, um festzustellen, wie stark diese voneinander abweichen.

Bewegungsmodelle In Kapitel 4 wurde das von uns entworfene Area Graph-based Modell vorgestellt und gezeigt, dass dieses Einfluss auf die Broadcastprotokolle hat. In diesem Kapitel werden die Ergebnisse von Szenarien mit unterschiedlichen Bewegungsmodellen (Random Waypoint und Area Graph-based Modell) miteinander verglichen, um zu untersuchen, wie stark sich das Bewegungsmodell auf die resultierende Konfliktrate auswirkt.

Kommutative Transaktionen Durch die Berücksichtigung von kommutativen Transaktionen soll festgestellt werden, wie stark sich die Verwendung der Transaktionssemantik auf die Anzahl der Konflikte auswirkt. Dieser Ansatz wurde gewählt, da er leichtgewichtig ist und für MANET-Szenarien anbietet.

10.2 Einfluss- und Messgrößen

In diesem Abschnitt werden sowohl die in die Untersuchung eingehenden Einflussgrößen als auch die aus den Untersuchungen resultierenden Messgrößen erläutert. Da ein Großteil dieser Messgrößen bereits in Kapitel 9 beschrieben wurde, wird an dieser Stelle jeweils nur eine kurze Erläuterung gegeben.

10.2.1 Einflussgrößen

Die experimentellen Untersuchungen werden durch die folgenden Einflussgrößen (Parameter) bestimmt:

- Die **Anzahl der Teilnehmer** der Replikationsgruppe beeinflusst sowohl die Gesamtanzahl der durchgeführten Transaktionen als auch die Netztopologie eines Szenarios. Da der Einfluss auf die Netztopologie analytisch nur bedingt zu bestimmen ist, wird diese Einflussgröße bei den experimentellen Untersuchungen besonders fokussiert.

- Die **Datenbankgröße** gibt die Anzahl der enthaltenen Datenelemente an und beeinflusst die Konfliktbildung. Die Datenelemente werden zufällig und gleichverteilt aus der Datenbank ausgewählt. Es werden Datenbankgrößen von bis zu 1000 Elementen betrachtet. Datenbanken mit mehr als 1000 Elementen würden die Konfliktwahrscheinlichkeit weiter verringern. Eine Berechnung der entsprechenden Wahrscheinlichkeit ist mit Hilfe der Konfliktanalyse möglich (Kapitel 9).

- Die **Größe der Transaktion** gibt die Anzahl ihrer Operationen an. In den experimentellen Untersuchungen werden Transaktion betrachtet, die entweder aus einer oder zwei Operationen bestehen.

- Der Erwartungswert der **Länge des Transaktionsintervalls** $E(T)$ gibt an, wie häufig ein Teilnehmer Transaktionen initiiert. Das konkrete Intervall wird (wie in der Konfliktanalyse) mit Hilfe einer Exponentialverteilung bestimmt.

- Das **Bewegungsmodell** bestimmt die Topologie des MANETs durch die Modellierung der Teilnehmerbewegungen. Ein Teilziel der Untersuchungen ist es, den Einfluss unterschiedlicher Bewegungsmodelle auf die Replikation zu betrachten.

Datenverteilung Auch die verwendeten Broadcastprotokolle können die Konfliktrate beeinflussen. Für diese Untersuchung wurden als fest gewählte Broadcastprotokolle die von uns entworfenen adaptiven Protokolle verwendet. Das adaptive Flutungsprotokoll verwendet den Schwellwert 10 (siehe Abschnitt 5.2.2), das adaptive Synchronisationsprotokoll den Schwellwert 0 (siehe Abschnitt 5.3.3).

Wenn es in den Szenarien nicht anders angegeben ist, wird ein Statusnachrichtenintervall von 180s verwendet. Das emulierte Netz verwendet für die Kommunikation ein IEEE 802.11 [Com99] konformes MAC Protokoll, bei dem die Bandbreite 11 Mbit/s und die Funkreichweite 100m beträgt[1].

10.2.2 Messgrößen

- **Anzahl der Transaktionen**, die während der Dauer des Szenarios initiiert wurden.

- **Anzahl der Transaktionen, die an einem Konflikt beteiligt waren**. Dieser Wert wird mit Hilfe des Vorgängergraphen bestimmt.

- **Anzahl der abgebrochenen Transaktionen**. Dieser Wert wird ebenfalls mit Hilfe des Vorgängergraphen bestimmt.

[1] Mit einer Paketverlustrate von 25%, bei größeren Entfernungen nimmt die Verlustrate stark zu.

- **Durchschnittliche Datenverteilungsdauer**. Dieser Wert gibt bei dem Versenden einer Nachricht an alle Knoten die Zeit an, die im Mittel benötigt wird um einen Knoten zu erreichen (siehe Abschnitt 2.1.4). Der Wert wird mit Hilfe der Zeitpunkte berechnet, an denen die Teilnehmer Transaktionen erstellen, bzw. Transaktionen von anderen Teilnehmer erhalten. Der Durchschnittswert wird über die Einzelwerte aller Knotenpaare und Nachrichten gebildet.

10.2.3 Zielgrößen

- **Konfliktrate** des Szenarios. Diese gibt den Anteil der Transaktionen an, die mit anderen Transaktionen in Konflikt stehen (bezogen auf die Gesamtanzahl). Bei den Ergebnissen der Konfliktanalyse entspricht die Konfliktrate der berechneten Konfliktwahrscheinlichkeit.
- **Abbruchrate** des Szenarios. Diese gibt den Anteil der abgebrochenen Transaktionen an (bezogen auf die Gesamtanzahl).
- **Durchschnittliche Datenverteilungsdauer** $E(D^*)$. Die durchschnittliche Datenverteilungsdauer ist ebenfalls eine Zielgröße und dient als Eingabeparameter $E(D)$ für die Konfliktanalyse. In der Konfliktanalyse wurde angenommen, dass sich die Teilnehmer bei der Datenverteilung in beide Richtungen synchronisieren. Im Gegensatz dazu gibt die Datenverteilung bei der Emulation eine gerichtete Verteilung eines Teilnehmers zu allen anderen Teilnehmern an, d. h., es besteht nur eine Synchronisation in eine Richtung. Der daraus resultierende Unterschied beträgt in Szenarien mit geringer Konfliktrate wie sie hier betrachtet werden im Ergebnis jedoch weniger als einen Prozentpunkt.

10.3 Aufbau der Experimente

Die experimentelle Untersuchung unserer Replikationsmethoden wird mit Hilfe eines Emulators durchgeführt. Dies hat den Vorteil, dass die bereits als Proof-of-Concept realisierte Implementierung des Replikationssystems SYMORE verwendet und für größere Szenarien getestet werden kann. Im Gegensatz zu einem Simulator ist die Verwendung eines Emulators aufwändiger und langwieriger, da die Emulationen im Normalfall nur in Echtzeit durchgeführt werden können. Das liegt daran, dass Teile der Emulation reale Komponenten enthalten, die nur unter Echtzeitbedingungen getestet werden können. In unserem Fall sind das die Netzwerkkommunikation und das Replikationssystem.

10.3.1 MarNET Emulator

Der MarNET Emulator [ESHF04] ist ein auf Linux basierender Netzemulator. Im Gegensatz zu anderen Emulatoren (z. B. [MRBV05]) umgeht der MarNET Emulator das Problem des hohen Bedarfs an echten Netzknoten, indem er mit Hilfe eines Mikrokernels erlaubt, mehrere mobile Knoten auf einem Linux Rechner zu emulieren. Der Emulator ist dabei nicht auf einen realen Rechner beschränkt,

sondern erlaubt auch die Verwendung mehrerer Rechner, auf denen jeweils mehrere mobile Knoten emuliert werden. Der Netzverkehr wird durch eine weitere Komponente, den sogenannten *Multiplexer*, realisiert. Dieser verbindet das reale Netz mit dem virtuellen Netz der emulierten Knoten. Die Steuerung der Emulation wird durch eine zentrale Steuerungskomponente vorgegeben.

10.3.2 Messumgebung

Die Architektur der von uns verwendeten Messumgebung ist in Abbildung 10.1 dargestellt. Der

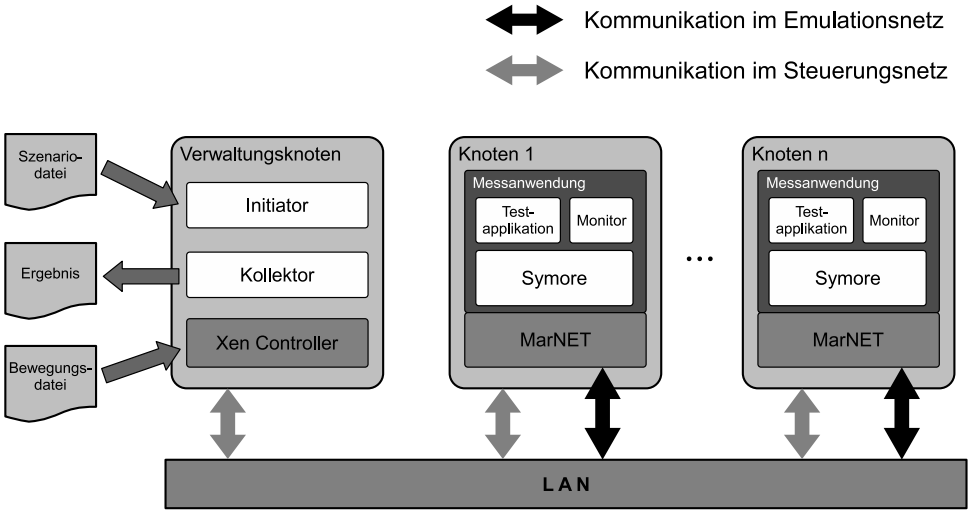


Abbildung 10.1: Übersicht der Messumgebung

zentrale *Verwaltungsknoten* des Emulators benötigt eine Szenariodatei und eine Bewegungsdatei, in der die Bewegungsdaten aller Knoten für die Dauer des Szenarios enthalten sind. Die Knoten werden über einen separaten Kanal gesteuert, damit keine Störungen bei den Messungen auftreten.

Alle emulierten Knoten enthalten eine lokale *Messanwendung*, welche die Testapplikation, den Monitor und das Replikationssystem SYMORE enthält. Die *Testapplikation* ist eine kleine Anwendung, die das Benutzerverhalten simuliert und Transaktionen initiiert. SYMORE ist das zu untersuchende System und entspricht der in Kapitel 8 beschriebenen Java Implementierung. Der *Monitor* zeichnet die Übertragungszeitpunkte und die Resultate der Konfliktlösung der von SYMORE verarbeiteten Transaktionen auf. Dazu meldet er sich beim Replikationssystem an (siehe Abschnitt 8.1).

Die Kommunikation der Knoten im emulierten Netz wird durch den MarNET Emulator gesteuert. Die Paketverlustrate wird auf Basis des Funkabstands mit Hilfe des Standardmodells des MarNET Emulators berechnet [ESHF04]. Die sogenannte Net-Shaper Komponente des MarNET Emulators simuliert dies, indem sie die versendeten Pakete entsprechend der gegebenen Verlustrate verwirft.

Messungenauigkeiten und -schwankungen

Die experimentellen Untersuchungen dieses Kapitels weisen für die Konfliktrate insgesamt einen durchschnittlichen Standardfehler von 1,9 Prozentpunkten auf. Dieser Wert ist höher als der für vergleichbare Untersuchungen, die mit einem Simulator durchgeführt werden können, da durch den Einsatz eines Emulators alle Szenarien nahezu in Echtzeit ablaufen müssen. Es wurden pro Teilszenario mindestens drei Messreihen durchgeführt, in denen je nach Teilnehmerzahl ca. 100 - 800 Transaktionen und ca. 500 - 32.000 Nachrichten verarbeitet wurden. Aus Gründen der Lesbarkeit werden die Standardfehler nicht pauschal mit angegeben, sondern in den Diskussionen der Ergebnisse nur erwähnt, wenn sie relevant sind.

Die Messungenauigkeiten, die sich aufgrund der verzögerten Benachrichtigung des lokalen Monitors eines Knotens ergeben, liegen im Millisekundenbereich (< 50ms) und können bei allen Szenarien vernachlässigt werden (Ausnahme: Spielszenario). Die durch die Synchronisation der lokalen Uhren der Knoten entstehende Ungenauigkeit ist ebenfalls sehr klein (< 80ms) und kann vernachlässigt werden.

Technische Parameter

Die Emulationen werden mit Hilfe von Pentium 4 (3 GHz) Desktop PCs (Debian Linux) durchgeführt, denen für die Emulation jeweils ein Gigabyte Hauptspeicher zur Verfügung steht. Ein PC wird separat als Verwaltungsknoten verwendet. Alle weiteren PCs beinhalten drei bis fünf emulierte Knoten, denen jeweils 300 Megabyte Hauptspeicher zur Verfügung stehen.

Da für die Emulation die Verbindungsinformationen des gesamten Szenarios auf jedem einzelnen emulierten Knoten gespeichert werden müssen, kann aufgrund des Speicherbedarfs maximal eine Gruppengröße von 40 emulierten Knoten betrachtet werden. Ebenso ist dadurch die Emulationsdauer auf zwei Stunden beschränkt. Aus diesem Grund wurde mit Hilfe von Tests untersucht, ob eine Beschleunigung der Emulation möglich ist. Bis zu einer Verdopplung der Emulationsgeschwindigkeit haben sich bei der Untersuchung keine messbaren Unterschiede ergeben. Aus diesem Grund werden alle länger laufenden Emulationen beschleunigt ausgeführt. Somit können Szenarien bis zu vier Stunden Länge emuliert werden.

10.4 Untersuchung unterschiedlicher MANET-Szenarien

In diesem Abschnitt stellen wir die Ergebnisse der Untersuchungen des Replikationssystems SYMORE vor. Hauptziel dieser Untersuchungsreihe ist die Analyse der Konfliktrate beim Einsatz von SYMORE in MANET-Szenarien mit kleinen bis mittelgroßen Replikationsgruppen. Zusätzlich wird die Auswirkung verschiedener Einflussgrößen auf die Konfliktrate betrachtet. Darüberhinaus werden die Ergebnisse der Untersuchung mit den Ergebnissen der Konfliktanalyse verglichen, um festzustellen, wie exakt die Konfliktrate eines Szenarios analytisch bestimmt werden kann.

10.4.1 Hypothesen

Für die vorliegende Untersuchungsreihe stellen wir folgende Hypothesen auf:

Hypothese 1 (Konfliktrate/Abbruchrate) Da in den betrachteten Szenarien die Replikationsgruppen klein sind und Schreibzugriffe eher selten stattfinden, nehmen wir an, dass sich in den experimentellen Untersuchungen für die Konfliktrate und die Abbruchrate akzeptable Werte ergeben.

Hypothese 2 (Einfluss der Teilnehmeranzahl/Skalierbarkeit) In den untersuchten Szenarien wird jeweils die resultierende Konfliktrate für unterschiedlich große Replikationsgruppen (Anzahl der Teilnehmer) betrachtet. Weil das Netz nur aus den Teilnehmern der Replikationsgruppe besteht, wirkt sich dieser Parameter nicht nur auf die Anzahl der Transaktionen, sondern auch auf die Netztopologie aus. Somit ist bei einer größeren Replikationsgruppe einerseits ein negativer Einfluss auf die Konfliktrate zu erwarten, da mehr Transaktionen initiiert werden. Andererseits erhöht sich dabei auch die Knotendichte des Netzes, wodurch die Verteilung der Transaktionen beschleunigt und die Konfliktrate reduziert wird.

Wir erwarten, dass sich diese beiden gegensätzlich wirkenden Einflussgrößen in den einzelnen Szenarien unterschiedlich stark auswirken. Es kann somit keine allgemeine Aussage getroffen werden, ob die Konfliktrate bei steigender Teilnehmeranzahl ebenfalls steigt oder aber sinkt. Es ist zu vermuten, dass bei einem Szenario mit geringer Fläche die positiven Auswirkungen überwiegen, da die Anzahl der Teilnehmer hier einen starken Einfluss auf die Netztopologie hat und somit die Datenverteilung beschleunigt.

Hypothese 3 (Genauigkeit der Konfliktanalyse) Die Konfliktanalyse vernachlässigt Aspekte der Datenverteilung und der Netztopologie. Aus diesem Grund ist zu erwarten, dass die Ergebnisse der Konfliktanalyse von denen der experimentellen Untersuchung abweichen. Um die Ergebnisse miteinander vergleichen zu können, wird die mittlere Datenverteilungsdauer eines Szenarios als Eingabeparameter für die Konfliktanalyse verwendet (siehe Abschnitt 10.2.3).

Obwohl derselbe Mittelwert verwendet wird, unterscheiden sich die Werte der Konfliktanalyse und die der experimentellen Untersuchungen in ihrer *Verteilung*. Diese gibt jeweils an, wie häufig bestimmte Werte vorkommen. Für die Konfliktanalyse wird eine Exponentialverteilung für die Datenverteilungsdauer angenommen, d. h., dass kleinere Werte häufiger vorkommen als größere. Weicht die tatsächliche Verteilung der Werte eines untersuchten Szenarios stärker von der Exponentialverteilung ab, ist auch zu erwarten, dass sich die von der Konfliktanalyse berechnete Konfliktrate stärker vom Ergebnis der experimentellen Untersuchung unterscheidet.

Wie bereits am Anfang dieses Kapitels erwähnt, ergibt sich bei der Konfliktanalyse auch aus der Annahme, dass sich bei einer Verbindung alle Teilnehmer synchronisieren, eine geringe Abwei-

chung von weniger als einem Prozentpunkt.

Wir vermuten für die Konfliktanalyse nur eine geringe Gesamtabweichung, die hinreichend klein ist, so dass mit Hilfe der Konfliktanalyse für gegebene Parameter die Größenordnung der Konfliktrate eines Szenarios bestimmt werden kann.

10.4.2 Katastrophenszenario

Das Katastrophenszenario wird mit Hilfe des Random Waypoint Bewegungsmodells umgesetzt. Die Fläche ist 800m $\times$ 800m groß, was zum Beispiel der Größe einer kleinen Ortschaft entspräche, die nach einer Katastrophe von Einsatzkräften verwaltet wird. Es wird angenommen, dass die Einsatzkräfte zu Fuß unterwegs sind und sich mit einer Geschwindigkeit von 1,3 $\frac{\text{m}}{\text{s}}$ durch das Gebiet bewegen. Dies entspricht der Geschwindigkeit zielgerichteten Gehens eines erwachsenen Menschen [Mol95]. Die Einsatzkräfte nehmen mit Hilfe der mobilen Geräte Daten über vermisste Personen, Vorräte und die Infrastruktur des Gebietes auf. Haben die Einsatzkräfte einen Zielort erreicht, so verweilen sie dort 10s - 300s. Das emulierte Zeitfenster des Szenarios beträgt vier Stunden. Alle weiteren Parameter sind in Tabelle 10.1 angegeben.

Anzahl der Teilnehmer	5,10,20,30,40
Datenbankgröße	1000 Elemente
Transaktionsgröße	1-2 Operationen
Transaktionsintervall[2]	720s

Tabelle 10.1: Parameter des Katastrophenszenarios

Die Ergebnisse des Katastrophenszenarios sind in Abbildung 10.2 dargestellt. Auf der x-Achse ist die Anzahl der Teilnehmer der Replikationsgruppe aufgetragen. Diese reicht von 5 bis 40 Teilnehmer. Auf der y-Achse ist links der Prozentwert (0% bis 80%) für die Konflikt- bzw. Abbruchrate angegeben. Auf der rechten Seite ist die durchschnittliche Verteilungsdauer $E(D^*)$ angegeben (gestrichelte graue Kurve).

Beobachtung Die durchschnittliche Verteilungsdauer fällt von fast 2900s mit steigender Teilnehmerzahl bis auf etwa 700s. Dies entspricht den Erwartungen für MANET-Szenarien dieser Größenordnung. Die Konfliktrate reicht von 3,8% bei 5 Teilnehmern bis 20,7% bei 40 Teilnehmern. Bei einer Gruppengröße von 20 Teilnehmern wird ein lokales Maximum von 11,8% erreicht.

Die Abbruchrate liegt immer ungefähr bei der Hälfte der Konfliktrate und erreicht ein Maximum von 10,7% bei 40 Teilnehmern. Die Konfliktrate der Konfliktanalyse ist für eine Teilnehmerzahl

[2] Die Länge des Transaktionsintervalls wird durch eine Exponentialverteilung bestimmt (siehe Abschnitt 10.2). In den Parametertabellen wird jeweils der Erwartungswert angegeben.

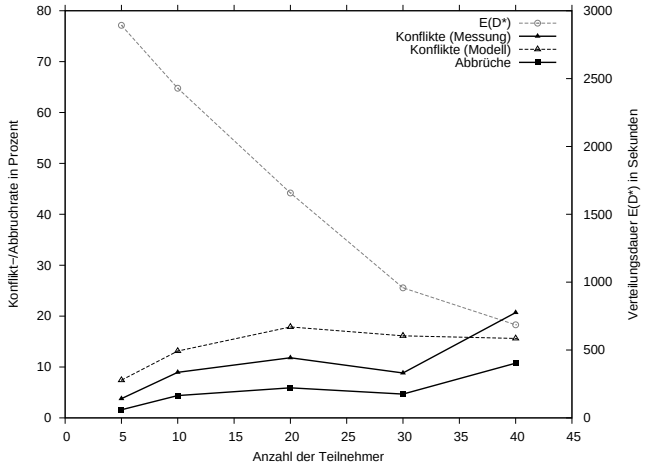


Abbildung 10.2: Ergebnis des Katastrophenszenarios

von 5 bis 30 etwas höher als die gemessenen Werte und bei 40 Teilnehmern etwas geringer. Die durchschnittliche Abweichung zwischen den gemessenen und den berechneten Werten beträgt 5,2 Prozentpunkte.

Diskussion Mit steigender Teilnehmerzahl erhöht sich die Knotendichte im Netz. Somit wird die Verteilungsdauer ebenfalls reduziert, was sich positiv auf die Konfliktrate auswirkt. Andererseits wird die Konfliktrate bei steigender Teilnehmeranzahl durch eine größere Anzahl an Transaktionen negativ beeinflusst.

Während die Konfliktrate bis zu einer Teilnehmerzahl von 20 monoton steigt, wirkt sich der positive Effekt der höheren Netzdichte erst ab 30 Teilnehmern stärker auf die Konfliktrate aus und reduziert diese. Offenbar wird bei dieser Teilnehmerzahl ein Schwellwert für die Konnektivität des Netzes überschritten, der eine stärkere Reduzierung der Konfliktrate zur Folge hat. Diese Vermutung deckt sich ebenfalls mit den Werten der Konfliktanalyse, deren Kurve zwischen 20 und 30 Teilnehmern ebenfalls fallend ist. Bei einer Gruppengröße von 40 steigt die Konfliktrate wiederum an, was durch die geringe Veränderung der Verteilungsdauer zu erklären ist. Bei Gruppengrößen von mehr als 40 Teilnehmern ist deshalb zu erwarten, dass die Konfliktrate weiter ansteigt.

Die Tatsache, dass die Abbruchrate bei allen Werten ungefähr der Hälfte der Konfliktrate entspricht, deutet darauf hin, dass die meisten Konflikte klein sind und nur aus zwei Transaktionen bestehen, von denen dann eine abgebrochen wird. Eine Überprüfung der resultierenden Vorgängergraphen des Szenarios bestätigt diese Vermutung.

Die maximale Abbruchrate von 10,7% stellt einen noch akzeptablen Wert für mobile Szenarien dar. Ob höhere Abbruchraten akzeptabel sind, hängt stark von der jeweiligen Anwendung ab und kann nicht pauschal beantwortet werden.

Eine Untersuchung des Katastrophenszenarios mit gleichbleibender Netztopologie ist im Anhang beschrieben (siehe Abschnitt B.1). Die resultierenden Abbruchraten sind dabei wie erwartet geringer als in diesem Szenario da eine konstante Anzahl von 40 Netzteilnehmern verwendet wurde und sich somit eine geringere Verteilungsdauer ergibt.

10.4.3 Campusszenario

Das Campusszenario wird mit Hilfe des Area Graph-based Bewegungsmodells modelliert. Das Modell ist in Abbildung 10.3 gezeigt und stellt ein Campusgelände mit zwei Instituten, einer Mensa und einer Cafeteria dar. Die Studenten bewegen sich innerhalb der Gebäude überhaupt nicht oder nur wenig. Die Aufenthaltsdauer der Studenten in einem Gebäude ist unterschiedlich und beträgt 30 - 120 Minuten. Zwischen den Gebäuden bewegen sich die Studenten mit 1,3 $\frac{m}{s}$ zum nächsten Gebäude.

Die Studenten verwenden ein Informationssystem (z. B. Wiki), das dazu dient, aktuelle Informationen über den Lehrbetrieb auszutauschen. Das emulierte Zeitfenster des Szenarios beträgt vier Stunden. Alle weiteren Parameter sind in Tabelle 10.2 angegeben. Im Gegensatz zu den anderen Szenarien ist die Datenbankgröße variabel und richtet sich nach der Anzahl der Teilnehmer. Dadurch wird ein Informationssystem betrachtet, in dem durch die Nutzer nicht nur Informationen geändert, sondern auch neue Informationen hinzufügt werden. So kann z. B. ein neuer Abschnitt zu einem Wiki Artikel hinzugefügt werden.

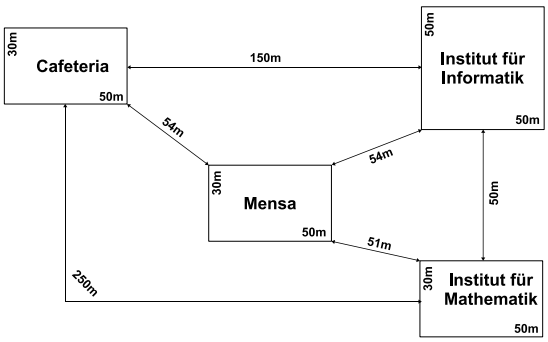


Abbildung 10.3: Bewegungsmodell des Campusszenarios

Die Ergebnisse des Campusszenarios sind in Abbildung 10.4 dargestellt.

Beobachtung Der Graph der Verteilungsdauer zeigt einen alternierenden Verlauf mit Werten zwischen 2519s und 2855s. Die Konfliktrate liegt zwischen 9,1% und 15,9%.

Die Abbruchrate ist für alle Gruppengrößen ungefähr halb so groß wie die Konfliktrate und erreicht bei 20 Teilnehmern ihren Maximalwert von 8,5%. Die durchschnittliche Abweichung der

Anzahl der Teilnehmer	10, 20, 30, 40
Datenbankgröße	Anz. Teiln. $\times$ 25 Elemente
Transaktionsgröße	1 Operation
Transaktionsintervall	1600s

Tabelle 10.2: Parameter des Campusszenarios

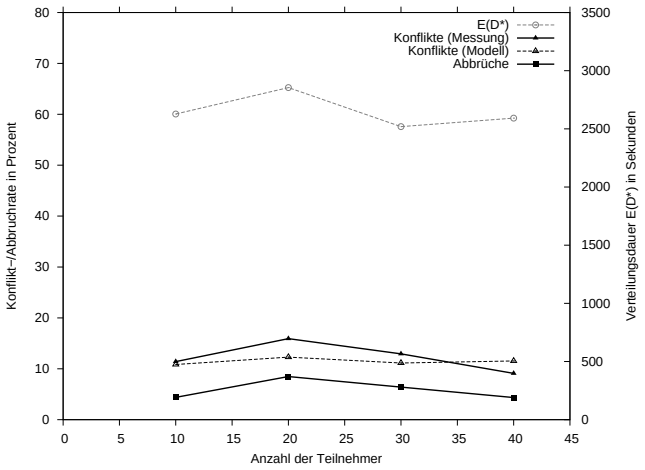


Abbildung 10.4: Ergebnisse des Campusszenarios

Werte der Konfliktanalyse von den Messergebnissen beträgt 2,1 Prozentpunkte.

Diskussion Im Gegensatz zu den Ergebnissen der vorigen Szenarien nimmt die Verteilungsdauer im Campusszenario mit steigender Teilnehmerzahl nicht monoton ab. Dies liegt an der Topologie des Bewegungsmodells des Campusszenarios. Aufgrund der langen Aufenthaltsdauer und der minimalen Bewegung der Teilnehmer in den Gebäuden ergeben sich bei größeren Gruppen vereinzelt isolierte Teilnehmer, die relativ lange Zeit nicht erreicht werden. Dieser Effekt wirkt dem positiven Einfluss der höheren Knotendichte entgegen, und somit ergibt sich der alternierende Verlauf der Verteilungsdauer.

Der Einfluss der zur Gruppengröße proportional zunehmenden Datenbankgröße ist sehr groß: Obwohl die Verteilungsdauer für alle Gruppengrößen länger als 2500s ist, nimmt die Konfliktrate bei 30 und 40 Teilnehmern ab. Es ergibt sich somit im Vergleich zu den Szenarien mit einer festen Datenbankgröße ein anderer Verlauf der Konfliktrate. Während zuvor eine Gruppengröße von mehr als 40 Teilnehmern problematisch erschien, sind in diesem Szenario größere Gruppengrößen denkbar.

Die geringen Abweichungen der Konfliktanalyse sind damit zu erklären, dass die Verteilungskurve der Verteilungsdauer für alle Gruppengrößen große Ähnlichkeit mit der angenommenen Vertei-

lungskurve der Exponentialverteilung aufweist.

Eine weiteres Szenario, das mit Hilfe des Area Graph-based Bewegungsmodells modelliert wurde, ist das Touristenszenario. Die Untersuchungsergebnisse dieses Szenarios ähneln den bisherigen Ergebnissen und werden deshalb in Abschnitt B.2 im Anhang beschrieben.

10.4.4 Spielszenario

Im Unterschied zu den vorherigen Szenarien findet das Spielszenario nur auf einer sehr kleinen Grundfläche ohne Bewegung statt. Somit befinden sich alle Teilnehmer in gegenseitiger Funkreichweite und es gibt keine Partitionierung des Netzes. Diese Modellierung soll eine gemeinsame Spielrunde mit mehreren Personen in einem Raum darstellen. Diese Spielrunde könnten z. B. mehrere in einem Zug reisende Personen sein, die in einem Abteil das in Abschnitt 8.3.2 vorgestellte Puzzlespiel gemeinsam spielen.

Ein weiterer Unterschied zu den vorigen Szenarien ist eine wesentlich häufigere Initiierung von Transaktionen. Die Länge der Transaktionsintervalle für dieses Szenario liegt nicht im Minuten-, sondern im Sekundenbereich. Aus diesem Grund ist auch das Statusnachrichtenintervall auf 30s reduziert. Das emulierte Zeitfenster des Szenarios beträgt 10 Minuten. Alle weiteren Parameter sind in der Tabelle 10.3 angegeben.

Ziel dieser Untersuchung ist herauszufinden, inwieweit sich das Replikationssystem SYMORE auch für zeitkritische Szenarien eignet, die das System einer größeren Last aussetzen. Da die Verteilungsdauer des Spielszenarios sehr gering ist, wirken sich die in Abschnitt 10.3.2 erwähnten Messungenauigkeiten stärker auf die Ergebnisse aus. Somit ist eine Betrachtung der Verteilungsdauer und der Ergebnisse der Konfliktanalyse nicht sinnvoll und es wird nur die resultierende Konfliktrate besprochen. Aufgrund der Beschränkungen des Emulators und der Messumgebung wurden nur Gruppengrößen bis zu 16 Teilnehmern betrachtet.

Anzahl der Teilnehmer	2, 4, 8, 16
Datenbankgröße	20 Elemente
Transaktionsgröße	1 Operation
Transaktionsintervall	5s, 10s, 15s

Tabelle 10.3: Parameter des Spielszenarios

Die Ergebnisse des Spielszenarios sind in Abbildung 10.4 dargestellt.

Beobachtung Die gemessenen Konfliktraten sind sehr unterschiedlich und reichen von 0% für 2 und 4 Teilnehmer bis hin zu 74% für 16 Teilnehmer (Transaktionsintervall 5s). Die Kurven der

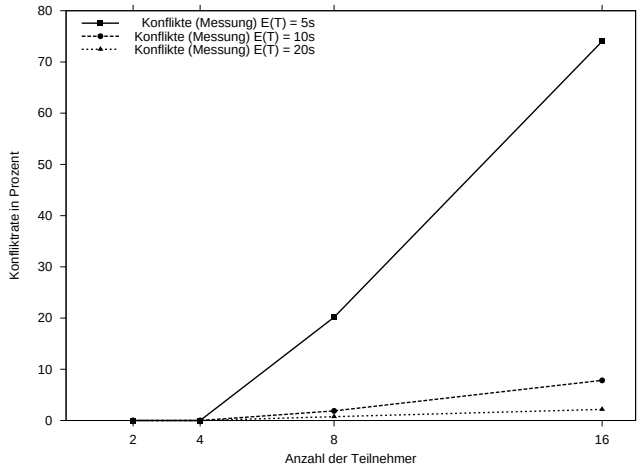


Abbildung 10.5: Ergebnisse des Spielszenarios

Untersuchungsreihen mit einem Transaktionsintervall von 10s und 20s zeigen selbst für 8 Teilnehmer und 16 Teilnehmer nur geringe Werte (0,7% bis 7,8%). Die Kurve der Untersuchungsreihe mit einem Transaktionsintervall von 5s hebt sich davon deutlich ab.

Diskussion Die drei Untersuchungsreihen des Spielszenarios lassen deutlich die Belastungsgrenze des Replikationssystems erkennen. Während ein Transaktionsintervall von 10s und 20s die Konfliktrate bei größeren Gruppen leicht ansteigen lässt, zeigt sich bei einem Intervall von 5s bereits bei einer Anzahl von 8 Teilnehmern eine Überlastung des Systems. Dies führt zu einer Erhöhung der Verteilungsdauer. Das liegt daran, dass die Verteilungsdauer nicht nur die Zeit beinhaltet, die benötigt wird, um eine Transaktion an einen anderen Teilnehmer zu versenden, sondern auch den Zeitbedarf, den das System benötigt, um die Transaktion zu verarbeiten und in den Vorgängergraphen einzufügen. Im Normalfall stellt der letztere Vorgang nur einen Bruchteil der gesamten Verteilungsdauer dar. Durch die Überlastung des Systems kehrt sich dieses Verhältnis jedoch um. Während die Dauer der eigentlichen Übertragung im Bereich von Zehntelsekunden liegt, ergibt sich für die Verarbeitung zum Teil eine Dauer von mehreren Sekunden, da die Transaktionen so lange im Eingangspuffer warten müssen. Aus diesem Grund entspricht die Verteilungsdauer in diesen Fällen einem Vielfachen des üblichen Wertes.

Abschließend ist festzuhalten, dass sich die vorliegende Implementierung für belastungsintensive Anwendungsszenarien mit zeitlich kritischen Anforderungen nur für kleine Replikationsgruppen bis maximal 8 Teilnehmer eignet. Dabei ist jedoch zu berücksichtigen, dass es sich um eine prototypische Implementierung handelt, deren Leistung noch nicht vollständig optimiert wurde.

10.4.5 Fazit und Überprüfung der Hypothesen

Betrachtet man die Ergebnisse der untersuchten Szenarien, sieht man, dass die Konfliktraten für Replikationsgruppen von bis zu 40 Teilnehmern nicht wesentlich über 20% steigen und in den meisten Fällen deutlich darunter liegen. Dadurch wird Hypothese 1 (Konfliktrate/Abbruchrate) bestätigt. Die Abbruchrate liegt in allen untersuchten Szenarien in etwa bei der Hälfte der Konfliktrate und erreicht einen Maximalwert von 11%. Somit zeigen sich die in dieser Arbeit vorgestellten Replikationsmethoden für die untersuchten MANET-Szenarien geeignet. Für Gruppengrößen von 50 - 100 Teilnehmern sind für die Katastrophenszenarien zu hohe Konfliktraten zu erwarten. Der Einsatz der vorgestellten Replikationsmethoden ist dann nicht mehr sinnvoll. In den anderen Szenarien wären zum Teil auch Gruppengrößen von mehr als 40 Teilnehmern denkbar.

In den Untersuchungen wurden Szenarien mit unterschiedlichen Bewegungsmodellen und verschiedenen Werten für die Einflussgrößen betrachtet. Aus diesem Grund lassen sich die gewonnenen Erkenntnisse auf MANET-Szenarien im Allgemeinen ausdehnen. Die untersuchten Replikationsmethoden sind somit für MANET-Szenarien mit kleinen bis mittelgroßen Replikationsgruppen und ähnlichen Werten für das Transaktionsintervall und der Verteilungsdauer $(E(T') \sim E(D^*))$ geeignet. Die einzige Ausnahme stellt das Spielszenario mit zum Teil sehr hohen Konfliktraten dar. Hierbei sollte jedoch die Belastungsgrenze des Systems getestet werden und deshalb wurden die Einflussgrößen entsprechend gewählt. Die Verteilungsdauer hingegen wird weniger von den Eigenschaften des Netzes, sondern hauptsächlich von der Verarbeitungsdauer des Replikationssystems bestimmt. Das Replikationssystem [Bre06] und die enthaltene Verteilungskomponente [Sch06] wurden prototypisch als Proof-of-Concept in Java realisiert. Dadurch ergeben sich im Hinblick auf die Kopplung der einzelnen Komponenten, die verwendete Programmiersprache und die Datenverwaltung (siehe Abschnitt 8.2.4) Möglichkeiten, die Verarbeitungsdauer des Systems zu reduzieren.

Wie in Hypothese 2 (Einfluss der Teilnehmeranzahl/Skalierbarkeit) angenommen, wird die Skalierbarkeit nicht nur von der Teilnehmerzahl beeinflusst, sondern hängt auch von der Topologie des Netzes ab: Die Konfliktrate wird durch eine größere Replikationsgruppe einerseits negativ beeinflusst, da mehr Transaktionen initiiert werden. Andererseits erhöht sich dadurch auch die Knotendichte des Netzes und die Verteilungsdauer verringert sich, wodurch die Konfliktrate reduziert wird. In den einzelnen Szenarien besteht für bestimmte Gruppengrößen ein besonders ungünstiges Verhältnis zwischen diesen beiden Einflussgrößen und es ergibt sich eine hohe Konfliktrate. Bildlich gesprochen ergeben sich „lokale Maxima“, wenn man die Konfliktrate als Funktionsergebnis und die Gruppengröße als Eingabewert betrachtet.

Die Vermutung, dass die Erhöhung der Teilnehmerzahl in kleineren Szenarien die Konfliktrate stärker reduziert, lässt sich nicht bestätigen. Es zeigt sich jedoch, dass beim Random Waypoint Modell der positive Effekt der Gruppengröße stärker zum Tragen kommt (siehe 10.4.2). Dies ist auf die Gleichverteilung der Teilnehmer auf der Grundfläche zurückzuführen.

Der Vergleich der Ergebnisse der Konfliktanalyse mit den Ergebnissen der Messungen zeigt in den

untersuchten Szenarien einen Unterschied zwischen 2,1 Prozentpunkten (Campusszenario) und 5,2 Prozentpunkten (Katastrophenszenario). Die durchschnittliche Abweichung der Werte für alle bisherigen Szenarien liegt bei 3,7 Prozentpunkten und ist unter Berücksichtigung des Standardfehlers der Messwerte (1,9 Prozentpunkte) verhältnismäßig gering. Es zeigt sich, dass die Abweichung der Konfliktanalyse eng mit der Verteilung der Werte der Verteilungsdauer zusammenhängt: Je stärker die Verteilung dieser Werte mit der angenommenen Exponentialverteilung übereinstimmt, desto weniger weichen die Werte der Konfliktanalyse von den gemessenen Werten ab. Es bestätigt sich somit die Annahme der Hypothese 3 (Genauigkeit der Konfliktanalyse): Die Konfliktanalyse ermöglicht eine Abschätzung der Größenordnung der Konfliktrate für gegebene Szenarien. Voraussetzung dafür ist die Kenntnis der entsprechenden Werte der Einflussgrößen, insbesondere der angenommenen Verteilungsdauer.

Wie bereits erwähnt, zeigt sich für die Abbruchraten mit nur geringen Schwankungen in allen Fällen in etwa eine Halbierung der Konfliktrate. Bei geringen Konfliktraten liegt dieser Wert sehr nahe, da an den Konflikten häufig nur zwei Transaktionen beteiligt sind. Dies wurde ebenfalls durch die Betrachtung der beteiligten Vorgängergraphen bestätigt. Nutzt man diese statistischen Erfahrungswerte, so lässt sich mit Hilfe der Konfliktanalyse für Szenarien mit geringen Konfliktraten (< 20%) zusätzlich die Abbruchrate abschätzen.

10.5 Einfluss der Bewegungsmodelle

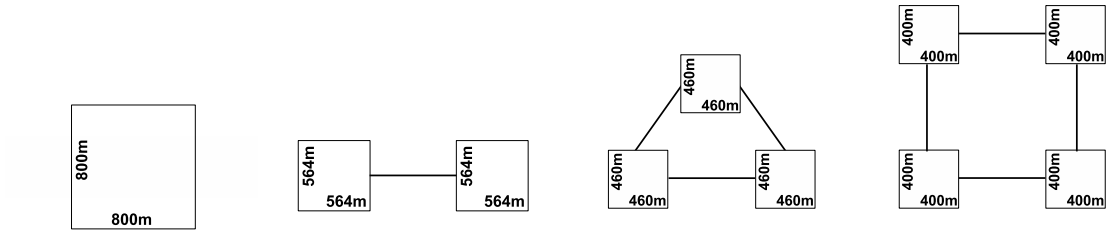
In dieser Untersuchungsreihe soll betrachtet werden, wie stark sich die Verwendung unterschiedlicher Bewegungsmodelle auf die gemessene und die berechnete Konfliktrate auswirkt. Es wird das Random Waypoint Modell (RWM) und das Area Graph-based Bewegungsmodell (AGM) verwendet. Da der Fokus auf dem Vergleich der unterschiedlichen Topologien beider Modelle liegen soll, betrachten wir Modelle mit der gleichen Knotendichte bzw. der gleichen Verteilungsdauer.

10.5.1 Hypothesen

Für die vorliegende Untersuchungsreihe stellen wir folgende Hypothesen auf:

Hypothese 1 (Einfluss bei gleicher Dichte) Wenn die Bewegungsmodelle eine gleiche Dichte besitzen, ist davon auszugehen, dass die Ergebnisse des AGMs höhere Konfliktraten liefern. Durch die Aufteilung der mobilen Knoten auf mehrere Cluster wird die Datenverteilung zwangsläufig verzögert und somit erhöht sich auch die Konfliktrate. Aus diesem Grund ist zu vermuten, dass die Konfliktrate bei steigender Clusteranzahl ebenfalls steigt.

Hypothese 2 (Einfluss bei gleicher Verteilungsdauer) Wenn die Bewegungsmodelle die gleiche durchschnittliche Verteilungsdauer besitzen, besteht der Unterschied zwischen den Modellen nur



(a) Random Waypoint (b) AGM - zwei Cl. (c) AGM - drei Cl. (d) AGM - vier Cl.

Abbildung 10.6: Topologie der Bewegungsmodelle mit gleicher Knotendichte

noch in der Verteilung der Einzelwerte der Datenverteilungsdauer. Aus diesem Grund ist zu vermuten, dass die resultierenden Konfliktraten nicht sehr stark voneinander abweichen.

10.5.2 Bewegungsmodelle mit gleicher Knotendichte

In dieser Untersuchung werden verschiedene Bewegungsmodelle mit gleicher Knotendichte miteinander verglichen. Die Knotendichte ist als Anzahl der mobilen Knoten pro Fläche definiert. Da für alle Untersuchungen eine gleiche Teilnehmeranzahl gegeben ist, besitzen alle Modelle die gleiche Flächengröße. Weil die Verbindungen zwischen den Clustern nicht in die Fläche mit einberechnet werden, muss gewährleistet werden, dass diese einen möglichst geringen Einfluss auf die Verteilung haben. Aus diesem Grund bewegen sich die Knoten auf den Verbindungen sehr schnell, womit erreicht wird, dass die Verteilung der Daten auf den Verbindungen praktisch nicht mehr ins Gewicht fällt.

Die unterschiedlichen Topologien der Modelle sind in Abbildung 10.6 dargestellt. Alle weiteren Parameter sind in Tabelle 10.4 angegeben.

Anzahl der Teilnehmer	40
Datenbankgröße	500 Elemente
Transaktionsgröße	1 Operation
Transaktionsintervall	2400s

Tabelle 10.4: Parameter der Untersuchungsreihe mit gleicher Knotendichte

Die Ergebnisse der Untersuchung für Bewegungsmodelle mit gleicher Knotendichte sind in Abbildung 10.7 dargestellt.

Beobachtung Die resultierende Verteilungsdauer der einzelnen Szenarien nimmt von links nach rechts zu und reicht von 594s bis 829s. Die Konfliktrate verhält sich dazu ähnlich und erreicht den Maximalwert für das AGM mit vier Clustern (4,9%). Der kleinste Wert wird beim RWM Szenario

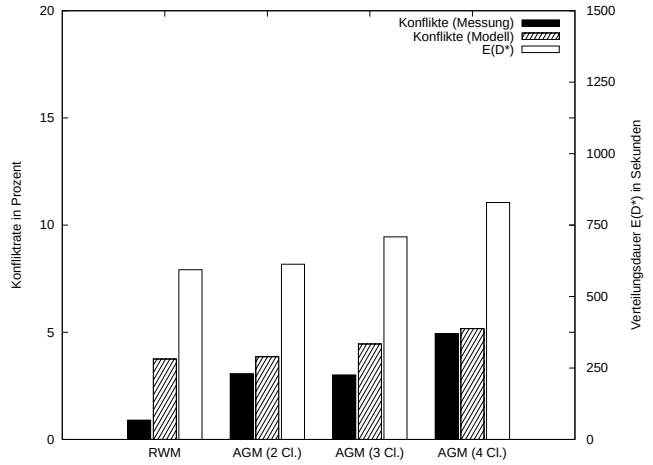


Abbildung 10.7: Ergebnisse für die unterschiedlichen Bewegungsmodelle mit gleicher Dichte

mit 0,9% erreicht. Die AGMs mit zwei und drei Clustern liefern ähnliche Werte (3,1% für zwei Cluster und 3,0% für drei Cluster).

Der Wert der Konfliktanalyse weicht mit 2,8 Prozentpunkten am stärksten beim ersten Szenario von den Messergebnissen ab. Für die AGMs ergeben sich geringere Abweichungen (0,2 Prozentpunkte bis 1,4 Prozentpunkte).

Diskussion Die Ergebnisse entsprechen den in den Hypothesen aufgestellten Erwartungen. Der Sonderfall, dass das AGM mit 2 Clustern in einer höheren Konfliktrate resultiert als das AGM mit 3 Clustern ist durch die Standardfehler zu erklären, die 0,5 Prozentpunkte (2 Cluster) und 0,1 Prozentpunkt (3 Cluster) betragen.

Die Vergleiche mit den Werten der Konfliktanalyse zeigen, dass die Berechnung für die AGMs genauer ist. Das liegt daran, dass die Verteilung der Verteilungsdauerwerte eher einer Exponentialverteilung entspricht als die resultierende Verteilung des Random Waypoint Modells.

10.5.3 Bewegungsmodelle mit gleicher Verteilungsdauer

In dieser Untersuchung werden verschiedene Bewegungsmodelle verglichen, welche dieselbe durchschnittliche Verteilungsdauer $E(D^*)$ besitzen. Die Untersuchungergebnisse sind im Detail in Abschnitt B.3 im Anhang beschrieben.

Die Ergebnisse zeigen geringere Abweichungen zwischen den Modellen als bei der vorigen Untersuchung. Der größte Unterschied ergibt sich in der gemessenen Konfliktrate zwischen dem RWM (6,9%) und einem AGM mit zwei Clustern (4,4%).

10.5.4 Fazit und Überprüfung der Hypothesen

Die Untersuchungen zeigen, dass sich das Random Waypoint Modell vom Area Graph-based Modell im Hinblick auf die resultierende Konfliktrate unterscheidet. Bei gleicher Knotendichte zeigen sich für die AGM Szenarien höhere Werte als für die RWM Szenarien. Dadurch wird Hypothese 1 (Einfluss bei gleicher Dichte) bestätigt.

Bei gleicher Verteilungsdauer unterscheiden sich die Werte des Random Waypoint Modells und des AGMs nur gering, wodurch sich Hypothese 2 (Einfluss bei gleicher Verteilungsdauer) als zutreffend erweist.

Desweiteren zeigt sich für die Ergebnisse der Konfliktanalyse eine bessere Übereinstimmung mit den Messwerten, die aus AGM Szenarien resultieren. Für RWM Szenarien sind die Abweichungen höher. Das liegt daran, dass die Verteilung der Werte der Verteilungsdauer bei AGM Szenarien näher an der angenommenen Exponentialverteilung der Konfliktanalyse liegt als die Verteilung bei RWM Szenarien.

10.6 Einfluss kommutativer Transaktionen

In dieser Untersuchungsreihe wird der Einfluss von kommutativen Transaktionen betrachtet. Wie bereits in Kapitel 7 beschrieben, müssen in Konflikt stehende Transaktionen, die kommutativ zueinander sind, nicht abgebrochen werden. Somit ergibt sich in diesen Fällen eine Reduktion der Abbruchrate. Wie sich der unterschiedliche Anteil von kommutativen Transaktionen auf die Abbruchrate auswirkt, soll anhand von experimentellen Untersuchungen zweier Szenarien betrachtet werden.

10.6.1 Hypothese

Bei der Verwendung von kommutativen Transaktionen ist grundsätzlich eine Reduktion der Abbrüche zu erwarten, da bei kommutativen Konflikten keine Transaktionen abgebrochen werden. Diese Abbruchrate müsste mit zunehmendem Anteil kommutativer Konflikte reduziert werden. Eine Verhinderung aller Abbrüche ist nicht anzunehmen, da selbst bei der ausschließlichen Verwendung von kommutativen Transaktionen Konflikte möglich sind, die zu Abbrüchen führen. Dies liegt daran, dass bei einem kommutativen Konflikt alle beteiligten Transaktionen dieselben Elternknoten besitzen müssen (siehe Abschnitt 7.3.1). Ist dies nicht der Fall, so muss ein Teil der Transaktionen abgebrochen werden. Aus diesem Grund müsste sich auch die Reduzierung der Abbruchrate mit zunehmendem Anteil kommutativer Transaktionen abschwächen.

10.6.2 Kommutative Transaktionen im Katastrophenszenario

In dieser Untersuchung wird erneut das Katastrophenszenario verwendet (siehe Abschnitt 10.4.2). Der einzige Unterschied besteht in der Größe der Transaktionen, da in dieser Untersuchung jede Transaktion zwei Schreiboperationen enthält.

Die Ergebnisse der Untersuchung sind in Abbildung 10.8 dargestellt. Im Diagramm werden nur die resultierenden Abbruchraten gezeigt. Dabei wird das Ergebnis ohne die Verwendung von kommutativen Transaktionen und die Ergebnisse für einen Anteil von 30%, 60% und 90% kommutativer Transaktionen gezeigt.

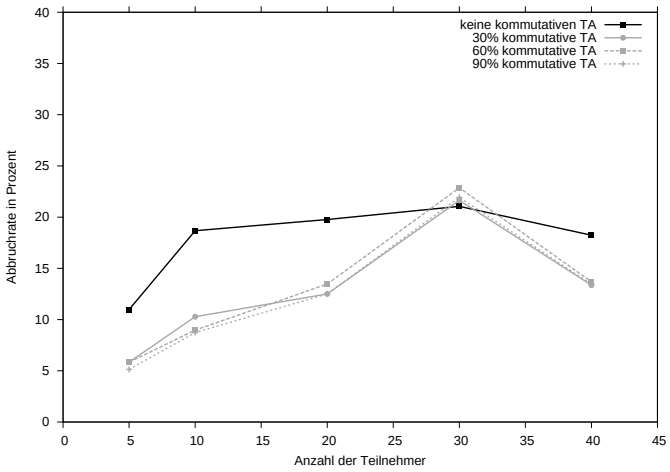


Abbildung 10.8: Ergebnisse für das Katastrophenszenario mit kommutativen Transaktionen

Beobachtung Die resultierenden Abbruchraten der Teilszenarien, in denen kommutative Transaktionen verwendet werden, unterscheiden sich nur geringfügig voneinander (maximal 1,5 Prozentpunkte). Für die Ergebnisse bei 5 und 10 Teilnehmern wird die ursprüngliche Abbruchrate (ohne kommutative Transaktionen) um etwa die Hälfte reduziert; bei 40 Teilnehmern nur um ein Viertel. Bei 30 Teilnehmern ist bei der Verwendung von kommutativen Transaktionen die Abbruchrate sogar leicht höher als die ursprüngliche.

Diskussion Wie vermutet, reduziert die Verwendung von kommutativen Transaktionen die Abbruchrate. Unerwartet ist, dass die Abbruchrate bei einem Anteil von 30% kommutativer Transaktionen zum Teil schon um die Hälfte reduziert wird. Die unterschiedlichen Anteile an kommutativen Transaktionen unterscheiden sich in ihren Auswirkungen jedoch kaum. Daraus lässt sich schließen, dass sich die bereits zuvor genannte Einschränkung bei kommutativen Konflikten (gleiche Eltern) stärker auswirkt als vermutet. Ebenfalls unerwartet ist die erhöhte Abbruchrate bei 30 Teilnehmern.

Nach einer detaillierten Untersuchung der Emulationen konnten Emulations- und Messfehler als Ursache ausgeschlossen werden. Die Betrachtung der resultierenden Vorgängergraphen zeigte jedoch, dass sich in diesem Fall die bereits beschriebene Einschränkung (gleiche Eltern) noch stärker auswirkt als bei den anderen Gruppengrößen.

Ebenso ist anhand der Ergebnisse zu erkennen, dass sich die Anzahl der Teilnehmer stark auf den Einfluss der kommutativen Transaktionen auswirkt. Aufgrund der unterschiedlichen Teilnehmerzahl sind auch die Abhängigkeiten zwischen den Transaktionen innerhalb der entstehenden Konfliktgruppen (siehe Abschnitt 7.2.2) sehr unterschiedlich und das wirkt sich besonders bei der Verwendung von kommutativen Transaktionen stark auf die resultierende Abbruchrate aus. Dies ist ebenfalls auf die Einschränkung der gleichen Eltern bei kommutativen Konflikten zurückzuführen. Trotzdem ergibt sich für die Mehrheit der Untersuchungen eine deutliche Reduktion der Abbruchrate.

10.6.3 Kommutative Transaktionen im Campusszenario

Die Auswirkungen von kommutativen Transaktionen werden ebenfalls für das Campuszenario untersucht. Das an dieser Stelle verwendete Szenario ist identisch mit dem aus Abschnitt 10.4.3.

Die Ergebnisse für das Campusszenario mit kommutativen Transaktionen sind in Abbildung 10.9 dargestellt.

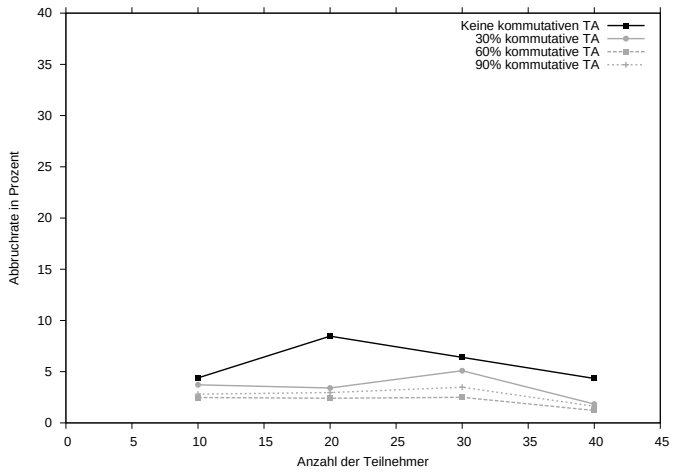


Abbildung 10.9: Ergebnisse für das Campusszenario und der Verwendung von kommutativen Transaktion

Beobachtung Die Ergebnisse der Teilszenarien mit kommutativen Transaktionen verlaufen ähnlich und es ergeben sich leichte Unterschiede von bis zu 2,6 Prozentpunkten. Die Ergebnisse der Untersuchungen mit einem Anteil von 30% kommutativer Transaktionen zeigen die geringste Re-

duktion der Abbruchrate. Bei einem Anteil von 60% wird die Abbruchrate am stärksten reduziert. Die Reduktion der Abbruchraten schwankt zwischen 15% und 71%.

Diskussion Wie bereits zuvor zeigt sich bei der Reduktion der Abbruchrate eine starke Abhängigkeit von der Anzahl der Teilnehmer. Im Gegensatz zum Katastrophenszenario werden die Abbruchraten bei der Verwendung von kommutativen Transaktionen immer reduziert. Diese Reduktion ist zumindest für die Anteile von 60% und 90% ebenfalls stärker als zuvor. Die stärkste Reduktion erfolgt bei einem Anteil von 60% kommutativer Transaktionen, bei 90% ist die Reduktion geringer. Dieses Verhalten ist wieder durch die Einschränkung der kommutativen Konflikte (gleiche Elternknoten) zu erklären: Bei einem hohen Anteil von kommutativen Transaktionen ergeben sich für viele Knoten im Vorgängergraphen auch mehr Elternknoten. Die Wahrscheinlichkeit, dass dabei zwei in einem kommutativen Konflikt stehende Knoten dieselben Eltern besitzen, verringert sich dadurch.

10.6.4 Fazit und Überprüfung der Hypothesen

Die Verwendung von kommutativen Transaktionen hat in den meisten Fällen eine deutliche Reduktion der Abbruchrate zur Folge. Dadurch bestätigt sich der erste Teil der aufgestellten Hypothese. Diese Reduktion hängt jedoch stark von der Anzahl der Teilnehmer ab, da diese den Aufbau der Konfliktgruppe beeinflusst, was sich erheblich auf den Einfluss kommutativer Transaktionen auswirkt.

Die zweite Vermutung der Hypothese (abnehmende Reduktion der Abbruchrate bei zunehmendem Anteil kommutativer Transaktionen) stimmt ebenfalls grundsätzlich. Der negative Einfluss der Einschränkung (gleiche Eltern) bei kommutativen Konflikten wirkt sich jedoch deutlich stärker aus als vermutet. Dies hat zur Folge, dass ein Anteil von 90% kommutativer Transaktionen im Campusszenario immer in einer schlechteren Reduktion der Abbruchrate resultiert als ein Anteil von 60%. Ebenso führt diese Einschränkung im Katastrophenszenario für eine Anzahl von 30 Teilnehmern bei der Verwendung von kommutativen Transaktionen sogar zu einer Erhöhung der Abbruchrate.

Insgesamt zeigt sich, dass eine Verwendung von kommutativen Transaktionen die Abbruchrate meistens reduziert. Die Einschränkung der kommutativen Transaktionen auf die gleiche Menge an Elternknoten wirkt diesem positiven Effekt jedoch zum Teil entgegen.

10.7 Zusammenfassung

In diesem Kapitel haben wir die Evaluation unserer Replikationsmethoden vorgestellt. Mit Hilfe der in Kapitel 8 vorgestellten Implementierung SYMORE konnten unsere einzelnen Ansätze im Verbund eines realen Gesamtsystems untersucht werden.

Damit die bereits existierende Implementierung des Systems ohne Veränderungen untersucht werden kann, wurde ein MANET Emulator verwendet. Mit Hilfe des Emulators wurde das System expe-

rimentell für mehrere unterschiedliche MANET-Szenarien untersucht. Die fokussierten Zielgrößen sind die resultierenden Konflikt- und Abbruchraten.

Die Ergebnisse der verschiedenen MANET-Szenarien zeigen, dass die resultierende Konfliktrate nur selten über 20% steigt und die Abbruchrate maximal 11% beträgt. Diese Ergebnisse belegen, dass sich der von uns gewählte Replikationsansatz grundsätzlich für MANET-Szenarien mit kleinen bis mittelgroßen Replikationsgruppen von bis zu 40 Teilnehmern eignet.

Ebenso wurde untersucht, wie stark die Resultate der Konfliktanalyse (siehe Kapitel 9) von den Ergebnissen der experimentellen Untersuchung abweichen. Die Ergebnisse haben gezeigt, dass die berechneten Werte der Konfliktanalyse für alle betrachteten Szenarien eine durchschnittliche Abweichung von weniger als 3 Prozentpunkten (3,7 Prozentpunkte in der ersten Untersuchungsreihe in Abschnitt 10.4) aufweisen. Im Hinblick auf die Zielsetzung (Bestimmung der Größenordnung der zu erwartenden Konfliktrate) und unter Berücksichtigung des durchschnittlichen Standardfehlers von 1,9 Prozentpunkten stellt dies eine hinreichende Genauigkeit dar. Die Abweichungen der Konfliktanalyse lassen sich hauptsächlich darauf zurückführen, dass für die Bestimmung der Verteilungsdauer der Transaktionen allgemein eine Exponentialverteilung angenommen wird. Besonders Szenarien, die das Random Waypoint Modell verwenden, zeigen größere Unterschiede in der Verteilung der Werte der Verteilungsdauer. Deshalb weichen diese in ihren Werten stärker von den Messergebnissen ab als Szenarien, die das Area Graph-based Modell verwenden.

Die Ergebnisse der Untersuchung bezüglich der Auswirkung unterschiedlicher Bewegungsmodelle haben gezeigt, dass das Random Waypoint und das Area Graph-based Modell auch bei gleicher Knotendichte unterschiedliche Ergebnisse bezüglich der Konfliktrate liefern. Bei gleicher Verteilungsdauer gab es ebenfalls erkennbare Unterschiede, jedoch waren diese nur gering.

Abschließend wurde ebenfalls der Einfluss der Verwendung von kommutativen Transaktionen auf die Abbruchrate untersucht. Die Ergebnisse haben gezeigt, dass die Abbruchrate in den meisten Fällen bereits bei einem relativ kleinen Anteil (30%) von kommutativen Transaktionen deutlich reduziert wird. Die Einschränkung, dass bei kommutativen Konflikten die beteiligten Knoten dieselben Elternknoten aufweisen müssen, wirkte sich jedoch unerwartet stark aus. Aus diesem Grund gab es für höhere Anteile an kommutativen Transaktionen eine geringere Reduktion der Abbruchrate als erwartet.

Teil IV

Abschluss

Kapitel 11

Zusammenfassung und Fazit

Im Rahmen der vorliegenden Arbeit wurde betrachtet, wie sich die Eigenschaften mobiler ad-hoc Netze auf die verwendeten Replikationsverfahren auswirken. Um die Eigenschaften solcher MANETs (Mobilität, Topologie) berücksichtigen zu können, wurden ebenfalls Bewegungsmodelle und Broadcastprotokolle betrachtet. Zur Bewertung der verwendeten Verfahren wurde ein Replikationssystem entwickelt und mit Hilfe eines Emulators experimentell untersucht. Im Folgenden werden die Beiträge und die in der Arbeit gewonnenen Erkenntnisse zusammengefasst.

Zunächst wurde das **Area Graph-based Bewegungsmodell** vorgestellt, das eine Erweiterung des Graph-based Modells ist. Das Modell verwendet eine übergeordnete Topologie, die durch einen gerichteten Graphen definiert wird. Darüberhinaus besitzen die Knoten des Graphen (Cluster) jeweils gesonderte Bewegungsmodelle, die die Bewegung der mobilen Knoten innerhalb der Fläche des Clusters bestimmen. Dadurch wird sowohl die makroskopische als auch die mikroskopische Bewegung der Knoten modelliert. Im Vergleich zu anderen Bewegungsmodellen konnte folgendes festgestellt werden:

- Die Datenverteilung innerhalb des Area Graph-based Modells unterscheidet sich von der Datenverteilung des Graph-based Modells. Wie groß diese Unterschiede sind, hängt stark von den Eigenschaften des zu modellierenden Szenarios ab. Entgegen unseren Erwartungen sind die Unterschiede in bestimmten Szenarien nur gering.

- Das Area Graph-based Bewegungsmodell ist realistischer als andere Bewegungsmodelle, die sich ausschließlich auf die Modellierung der Makro- oder der Mikrobewegung beschränken, wie z. B. das Random Waypoint, das Random Walk oder das Graph-based Bewegungsmodell.

Für eine effiziente Verteilung der Replikationsdaten wurden zwei **adaptive Broadcastprotokolle** entwickelt: das adaptive Flutungsprotokoll und das adaptive Synchronisationsprotokoll. Beide Protokolle wurden mit bisherigen Protokollen verglichen. Es ergaben sich folgende Ergebnisse:

- Das adaptive Verhalten der beiden entwickelten Protokolle reduziert die Datenlast, die für eine Verteilung der Daten im Netz benötigt wird. Das adaptive Flutungsprotokoll spart in den

untersuchten Szenarien im Vergleich zu den anderen Protokollen bis zu 50% der Netzlast ein. Das Synchronisationsprotokoll hingegen spart im Vergleich zu den anderen Protokollen in den untersuchten Szenarien bis zu 13% der Netzlast ein. Diese Einsparung ist in mobilen Netzen besonders wichtig, da die mobilen Geräte nur über begrenzte Ressourcen und Energie verfügen.

- In Szenarien, in denen das Area Graph-based Modell verwendet wurde, spart das adaptive Flutungsprotokoll im Vergleich zu anderen Protokollen die meiste Netzlast ein. Hierbei zeigt sich, dass nicht nur die mittlere Knotendichte eines Szenarios relevant ist, sondern auch die Topologie eines MANETs deutliche Auswirkung auf die Effizienz der Verteilungsprotokolle hat.

Es wurde ebenfalls ein **Uhrensynchronisationsverfahren** für MANETs entwickelt. Das verwendete Verfahren schätzt die Sendeverzögerung der Uhrensynchronisation ab. Dadurch ist es möglich, für zwei beliebige Echtzeit-Zeitstempel unterschiedlicher Teilnehmer zu bestimmen, ob deren Echtzeitreihenfolge garantiert werden kann. Die Echtzeit-Zeitstempel werden von der Konfliktlösung und den Commitverfahren verwendet.
Das **Verfahren zur Erkennung und Lösung von Konflikten** berücksichtigt die kausalen Abhängigkeiten der Transaktionen, um Konflikte zwischen diesen zu erkennen. Die Abhängigkeiten werden im Replikationssystem mit Hilfe einer Datenstruktur, dem sogenannten Vorgängergraphen, verwaltet. Diese Datenstruktur ist eine Erweiterung der bisher von anderen Systemen verwendeten Vorgängerliste und berücksichtigt im Gegensatz dazu nicht nur einzelne Operationen, sondern auch Transaktionen. Die Konflikterkennungs- und -lösungsalgorithmen arbeiten auf dem Vorgängergraphen. Im Hinblick auf die Komplexität des Verfahrens konnte folgendes festgestellt werden:

- Der Konflikterkennungs- und -lösungsalgorithmus besitzt im Falle von syntaktischer Konflikterkennung eine lineare Laufzeit. Aufgrund der eingeschränkten Leistung mobiler Geräte ist eine geringe Komplexität der Algorithmen notwendig.

- Wird ebenfalls die Kommutativität von Transaktionen berücksichtigt, ergibt sich eine quadratische Laufzeit in Bezug auf die Menge der an einem Konflikt beteiligten Transaktionen und deren Abhängigkeiten. In den untersuchten MANET-Szenarien sind häufig nur wenige Transaktionen an einem Konflikt beteiligt. Deshalb bietet sich dieser Konfliktlösungsansatz auch für mobile Geräte an.

Als Proof-of-Concept der gewählten Replikationsmethoden und als Basis für die experimentellen Untersuchungen wurde das **Replikationssystem SYMORE** implementiert. Es wurden ebenfalls zwei Beispielapplikationen (verteiltes Puzzlespiel, mobiles Wiki) entwickelt und gemeinsam mit dem Replikationssystem auf PDAs und Laptops getestet. Dabei ergaben sich folgende Erkenntnisse:

- Das Replikationssystem SYMORE lässt sich auf mobilen Geräten wie PDAs und Laptops einsetzen. Die in den MANET-Szenarien beschriebenen Tätigkeiten lassen sich somit bereits in der Realität mit Hilfe von SYMORE durchführen.

- Es zeigt sich, dass bereits bei einfachen mobilen Applikationen eine transaktionale Verarbeitung von Operationen benötigt wird, um die Logik der Applikation korrekt abzubilden. Die von uns oben beschriebene Berücksichtigung von Transaktionen im Vorgängergraphen ist somit auch in mobilen Szenarien eine notwendige Erweiterung der bisherigen Verfahren.

Um die vorgeschlagenen Replikationsmethoden analytisch bewerten zu können, wurde ein Verfahren zur **Konfliktanalyse** entwickelt. Mit Hilfe der Konfliktanalyse kann die Konfliktwahrscheinlichkeit für bestimmte Replikationsszenarien berechnet werden. Als Ergebnis der Konfliktanalyse konnten folgende Sachverhalte festgestellt werden:

- Das vorgestellte Verfahren zur Analyse der Konfliktwahrscheinlichkeit ist genauer als die vergleichbare Analyse von Gray et al. [GHOS96].

- Der Vergleich der berechneten Konfliktwahrscheinlichkeit mit den Ergebnissen der experimentellen Untersuchungen hat gezeigt, dass die Werte in den untersuchten Szenarien im Mittel weniger als 3 Prozentpunkte voneinander abweichen. Die Konfliktanalyse kann somit als Basis für zukünftige Untersuchungen dienen, um die Konfliktwahrscheinlichkeit für Replikationsszenarien analytisch zu bestimmen und Simulationsergebnisse zu ergänzen.

- Eine analytische Betrachtung der Auswirkungen der Transaktionsgröße hat gezeigt, dass ab einer Transaktionsgröße von vier Operationen selbst bei kleinen Replikationsgruppen zu viele Konflikte entstehen. Diese Beschränkung der Transaktionsgröße ist für mobile Szenarien jedoch in vielen Fällen unproblematisch. Im Gegensatz zu Replikationsszenarien in Festnetzen werden von den Applikationen in mobilen Szenarien eher kleinere und kurz laufende Transaktionen benötigt.

Abschließend wurde eine **experimentelle Untersuchung** der Replikationsmethoden anhand des Replikationssystems SYMORE durchgeführt, um den vorgeschlagenen Replikationsansatz zu evaluieren. Die Untersuchungen wurden mit Hilfe des MarNET Emulators für unterschiedliche MANET-Szenarien durchgeführt und lieferten folgende Ergebnisse:

- In den untersuchten Szenarien ergaben sich für Replikationsgruppen von bis zu 40 Teilnehmern Konfliktraten, die nur selten über 20% lagen. Die entsprechenden Abbruchraten hatten einen Maximalwert von 11%. In den meisten Festnetz-basierten Replikationsszenarien wären diese Werte nicht akzeptabel. Die Rahmenbedingungen mobiler Szenarien unterscheiden sich davon jedoch deutlich: Aufgrund der Mobilität sind die Verbindungen zwischen den Replikationsteilnehmern häufig unterbrochen. Ebenso entstehen dadurch Netzpartitionen, die zum

Teil über längere Zeiträume bestehen bleiben. Vor diesem Hintergrund ergeben sich abgeschwächte Anforderungen und die oben genannte Abbruchrate ist aus Sicht des Benutzers akzeptabel.

- Werden in einem Szenario auch kommutative Transaktionen verwendet, so wird die Abbruchrate dadurch weiter verringert. Dabei reicht bereits ein relativ geringer Anteil an kommutativen Transaktionen aus, um die Abbruchrate deutlich zu reduzieren.

Beantwortung der Forschungsfragen

Ist ein optimistischer Replikationsansatz für den Einsatz in MANETs trotz der erhöhten Datenverteilungsdauer praktikabel oder entstehen zu viele Konflikte?

Die Untersuchungsergebnisse dieser Arbeit zeigen, dass bei der Verwendung eines optimistischen Replikationsansatzes in kleinen bis mittelgroßen MANET-Szenarien aufgrund der entstandenen Konflikte in den meisten Fällen 5% bis 10% der Transaktionen abgebrochen werden müssen. Der vorgeschlagene Replikationsansatz eignet sich somit für die untersuchten Szenarien, da die ermittelten Abbruchraten für mobile Szenarien akzeptabel sind.

Um die Eignung des von uns vorgestellten Replikationsansatzes für weitere MANET-Szenarien abzuschätzen, wurde von uns eine analytische Bewertung durchgeführt. Mit Hilfe dieser Analyse ist es möglich, die Größenordnung der resultierenden Konfliktwahrscheinlichkeit für gegebene MANET-Szenarien zu bestimmen. Dadurch können zukünftige Simulationen von ähnlichen Szenarien sinnvoll ergänzt werden.

Wie lässt sich ein optimistischer Replikationsansatz in MANETs dezentral organisieren?

Die Konflikterkennung und -lösung, die Uhrensynchronisation und das Commitverfahren sind in unserem Replikationsansatz dezentral organisiert. Für eine dezentrale Konflikterkennung und -lösung ist es notwendig, dass jeder Teilnehmer die kausalen Abhängigkeiten der ihm bekannten Transaktionen kennt. Die kausalen Abhängigkeiten werden in unserem System mit Hilfe des vorgestellten Vorgängergraphen verwaltet. Dadurch ist die Konflikterkennung bei allen Teilnehmern konsistent. Die Konfliktlösung basiert auf den Echtzeit-Zeitstempeln der Transaktionen. Die Zeitstempel werden vom Uhrensynchronisationsverfahren verwaltet. Durch diese dezentrale Verwaltung können die einzelnen Teilnehmer jederzeit auf dem ihnen bekannten Datenstand arbeiten. Diese hohe Verfügbarkeit der Daten ist in MANETs besonders wichtig, da es vorkommen kann, dass ein Teilnehmer über mehrere Stunden keinen Kontakt zu anderen Teilnehmern der Replikationsgruppe hat.

Eine dezentrale Gruppenverwaltung und ein dezentrales Commitverfahren ist in MANETs ebenfalls denkbar. Die Nutzbarkeit hängt jedoch vom Szenario ab, in dem das System eingesetzt wird. In

einem Katastrophenszenario zum Beispiel finden sich die Einsatzkräfte am Anfang und in bestimmten Zeitabständen zusammen, wodurch eine Initialisierung und Änderung der Replikationsgruppe vereinfacht ist. Echte Ausfälle eines Teilnehmers gibt es dabei nur selten. Längere Verbindungsunterbrechungen von einigen Stunden sind der Regelfall und führen nicht zum Ausschluss eines Teilnehmers. In einem Touristenszenario, in dem die Teilnehmer beliebig hinzukommen oder wegfallen können, ist der Einsatz einer Gruppenverwaltung dagegen nicht sinnvoll.

Welche optimistischen Replikationsmethoden eignen sich aufgrund ihrer Komplexität für den Einsatz auf mobilen Geräten?

Eine syntaktische Konflikterkennung und Konfliktlösung mit Hilfe von Zeitstempeln ist in linearer Laufzeit möglich. Die Berücksichtigung von kommutativen Transaktionen hat eine quadratische Laufzeit in Bezug auf die Menge der an einem Konflikt beteiligten Transaktionen und deren Abhängigkeiten. Dies hat jedoch kaum eine Auswirkung, da sich in den untersuchten Szenarien gezeigt hat, dass häufig nur wenige Transaktionen an den Konflikten beteiligt sind. Deshalb eignen sich die verwendeten Methoden für den Einsatz auf mobilen Geräten mit eingeschränkter Leistung.

Die experimentellen Untersuchungen zeigen ebenfalls, dass die Implementierung des Replikationssystems für die Verwendung auf mobilen Geräten geeignet ist, da den emulierten Knoten nur eingeschränkte Ressourcen zur Verfügung standen und es im Rahmen der Untersuchung zu keiner unvorhergesehenen Überlastung des Systems kam. Auch die praktischen Erfahrungen mit realen Geräten (PDAs, Laptops) bestätigen diese Erkenntnis.

Fazit

Es zeigt sich, dass der Einsatz des von uns entwickelten Replikationssystems SYMORE und der dabei verwendeten optimistischen Replikationsmethoden in bestimmten MANET-Szenarien sinnvoll ist, um mobile Applikationen zu unterstützen, die mit verteilten Daten arbeiten. Dabei ist zu berücksichtigen, dass sich die Anforderungen von MANET-Szenarien zum Teil erheblich von den Anforderungen herkömmlicher Replikationsszenarien unterscheiden. Die Applikation und die Benutzer nehmen aufgrund der hohen Dynamik von MANETs einen gewissen Grad an Unzuverlässigkeit bzw. mangelnde Aktualität der Daten in Kauf. So ist zum Beispiel eine höhere Konfliktanzahl akzeptabel als in anderen Replikationsszenarien. Ebenso verhält es sich bei den Commitverfahren und der Verwaltung der Gruppenmitgliedschaft. Die Abwesenheit eines Replikationsteilnehmers von mehreren Stunden wird hier nicht als Fehler interpretiert und führt somit auch nicht zum Ausschluss eines Teilnehmers.

Nichtsdestotrotz existieren ebenfalls MANET-Szenarien, in denen aufgrund von Konflikten zu viele Transaktionen abgebrochen werden müssen oder in denen die Verwaltung der Gruppenmitgliedschaft nicht möglich ist. Im untersuchten Katastrophenszenario ist bei einer Gruppengröße von 40 Teilnehmern mit einer Abbruchrate von 11% in etwa die Grenze erreicht. Größere Replikations-

gruppen von 50 bis 100 Teilnehmern hätten zu viele Konflikte zur Folge. Das betrachtete Touristenszenario eignet sich aufgrund der hohen Gruppendynamik nicht. Die Anzahl der Konflikte ist im Touristenszenario jedoch akzeptabel. Da in diesem Szenario nur ein loser Informationsaustausch benötigt wird, ist der Einsatz von Replikation hierbei ohne eine Gruppenverwaltung und globalem Commit denkbar.

Insgesamt gesehen kann man schwer allgemein gültige Aussagen über die Eignung von optimistischer Replikation in mobilen Szenarien treffen, sondern muss dies von Fall zu Fall anhand der einzelnen Eigenschaften und Anforderungen eines Szenarios entscheiden. Die in dieser Arbeit gewonnenen Erkenntnisse sollen dabei als Basis für zukünftige Untersuchungen im Bereich Replikation in mobilen Netzen dienen.

11.1 Ausblick

An dieser Stelle werden kurz weiterführende Arbeiten erläutert, die auf den Erkenntnissen und Ergebnissen der Arbeit aufbauen.

Semantische Konflikterkennung In dieser Arbeit wurde in erster Linie optimistische Replikation mit syntaktischer Konflikterkennung betrachtet, da diese eine geringe Komplexität aufweist und somit für mobile Geräte mit geringerer Leistung geeignet ist. Ebenfalls wurde ein erster Ansatz zur semantischen Konflikterkennung basierend auf der Kommutativität von Transaktionen untersucht. Aufgrund der strikten Integritätsbedingung (gleiche Elternknoten) für kommutative Transaktionen sollte überprüft werden, wie sich eine Abschwächung dieser Bedingung auf die Abbruchrate auswirkt.

Darüberhinaus stellt sich die Frage, inwieweit die zusätzliche Berücksichtigung von Semantik die Konfliktrate weiter verringern kann, ohne dabei die mobilen Geräte zu überlasten. Für diese Betrachtung ist es sinnvoll zu überprüfen, ob die Ansätze des Replikationssystems IceCube [KRSD01] entsprechend verändert werden können.

Hybride Netztopologien Die in dieser Arbeit untersuchten Szenarien sind reine MANETs ohne Anbindung an ein Festnetz. Wie bereits in der Einleitung erwähnt, sind diese nur als temporäre Teilszenarien eines größeren Gesamtszenarios aufzufassen. Ein reines MANET besteht dabei nur für wenige Stunden oder Tage solange keine Verbindung zu einem Festnetz besteht. Es ist jedoch beim Katastrophen- oder Forscherszenario eine Basis denkbar, zu der die Teilnehmer nach einer bestimmten Zeit wieder zurückkehren.

Es bleibt zu untersuchen, wie die bisherigen Methoden an dieses neue Systemmodell angepasst werden müssen und welche Konsequenzen sich daraus ergeben. So ist für das gesamte Szenario durchaus eine zentrale Replikationsverwaltung möglich, während für das temporäre MANET-Szenario eine dezentrale Verwaltung beibehalten wird.

Literaturverzeichnis

[AGPM08] ASCHENBRUCK, N., E. GERHARDS-PADILLA und P. MARTINI: *A Survey on Mobility Models for Performance Analysis in Tactical Mobile Networks*. Journal of Telecommunications and Information Technology, Seiten 558–565, 2008.

[Bar03] BARRETO, J. P.: *Information Sharing in Mobile Networks: a Survey on Replication Strategies*. Technischer Bericht RT/015/03, Instituto Superior Técnico/Distributed Systems Group, Inesc-ID Lisbon, September 2003.

[BBHS05] BÖSE, J., F. BREGULLA, K. HAHN und M. SCHOLZ: *Adaptive Data Dissemination in Mobile ad-hoc Networks*. In: *Proceedings of the first Workshop of the GI-Fachgruppe Mobilität und Mobile Informationssysteme*, Bonn, Deutschland, September 2005.

[BHG87] BERNSTEIN, P. A., V. HADZILACOS und N. GOODMAN: *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[BHPC04] BETTSTETTER, C., H. HARTENSTEIN und X. PÉREZ-COSTA: *Stochastic Properties of the Random Waypoint Mobility Model*. Wireless Networks, 10(5):555–567, 2004.

[Bir93] BIRMAN, K. P.: *The Process Group Approach to Reliable Distributed Computing*. Commun. ACM, 36(12):37–53, 1993.

[Bre06] BREGULLA, F.: *Symore - A System For Mobile Replication*. Diplomarbeit, Institut für Informatik, Freie Universität Berlin, 2006.

[BRS05] BITTNER, S., W.-U. RAFFEL und M. SCHOLZ: *The Area Graph-based Mobility Model and its Impact on Data Dissemination*. In: *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '05)*, Seiten 268–272, Kauai Island, Hawaii, März 2005.

[BW02] BETTSTETTER, C. und C. WAGNER: *The Spatial Node Distribution of the Random Waypoint Mobility Model*. In: *Mobile ad-Hoc Netzwerke, 1. deutscher Workshop über Mobile ad-Hoc Netzwerke WMAN 2002*, Seiten 41–58. GI, 2002.

[CBD02] CAMP, T., J. BOLENG und V. DAVIES: *A Survey of Mobility Models for Ad hoc Network Research*. Wireless Communication & Mobile Computing (WCMC): Special is-

sue on Mobile Ad Hoc Networking: Research, Trends and Applications, 2(5):483–502, 2002.

[CDK02] COULOURIS, G., J. DOLLIMORE und T. KINDBERG: *Verteilte Systeme: Konzepte und Design*. Pearson Studium, 2002.

[Ced10] CEDERQUIST, P.: *Version Management with CVS*. (Link besucht am 10.06.2010), Juni 2010. `http://www.cvshome.org/docs/manual/`.

[Com99] COMMITTEE, I. S.: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*. In: *IEEE 802.11 Standard*. IEEE, New York, 1999.

[Cri89] CRISTIAN, F.: *Probabilistic Clock Synchronization*. Distributed Computing, Seiten 146–158, 1989.

[CS03] CARTIGNY, J. und D. SIMPLOT: *Border Node Retransmission based Probabilistic Broadcast Protocols in ad-hoc Networks*. In: *Telecommunication Systems*, Seiten 189–204, 2003.

[CT96] CHANDRA, T. und S. TOUEG: *Unreliable Failure Detectors for Reliable Distributed Systems*. Journal of the ACM, Seiten 374–382, 1996.

[Dav00] DAVIES, V. A.: *Evaluating Mobility Models within an Ad hoc Network*. Thesis, M. Sc., Colorado School of Mines, 2000.

[der10] *Apache Derby*. (Link besucht am 17.05.2010), Mai 2010. `http://db.apache.org/derby/`.

[DH00] DIEKMANN, O. und J.A.P. HEESTERBEEK: *Mathematical Epidemiology of Infectious Diseases: Model Building, Analysis and Interpretation*. John Wiley & Son, 2000.

[DHA03] DATTA, A., M. HAUSWIRTH und K. ABERER: *Updates in Highly Unreliable, Replicated Peer-to-Peer Systems*. In: *Proceedings of the 23rd International Conference on Distributed Computing Systems, ICDCS2003*, 2003.

[DPS+94] DEMERS, A. J., K. PETERSEN, M. J. SPREITZER, D. B. TERRY, M. M. THEIMER und B. B. WELCH: *The Bayou Architecture: Support for Data Sharing Among Mobile Users*. In: *Proceedings IEEE Workshop on Mobile Computing Systems & Applications*, Seiten 2–7, Santa Cruz, California, 1994.

[DQA04] DATTA, A., S. QUARTERONI und K. ABERER: *Autonomous Gossiping: A Self-Organizing Epidemic Algorithm for Selective Information Dissemination in Mobile ad-hoc Networks*. In: *International Conference on Semantics of a Networked World (IC-SNW'04)*, Paris, Juni 2004.

[Ein05] EINSTEIN, A.: *Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen*. Annalen der Physik, 17:549–560, 1905.

[eKBF03] ÇETINTEMEL, U., P. J. KELEHER, B. BHATTACHARJEE und M. J. FRANKLIN: *Deno: A Decentralized, Peer-to-Peer Object-Replication System for Weakly Connected Environments*. IEEE Trans. Comput., 52(7):943–959, 2003.

[EMRP01] EKENSTAM, T., C. MATHENY, P. L. REIHER und G. J. POPEK: *The Bengal Database Replication System*. Distributed and Parallel Databases, 9(3):187–210, 2001.

[EN02] ELMASRI, R. und S. B. NAVATHE: *Fundamentals of Database Systems, 3rd Edition*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 2002.

[epc10] *Eee PC*. (Link besucht am 10.06.2010), Juni 2010. `http://www.eee-pc.de/`.

[ESHF04] ENGEL, M., M. SMITH, S. HANEMANN und B. FREISLEBEN: *Wireless ad-hoc Network Emulation using Microkernel-Based Virtual Linux Systems*. In: *Proceedings of the 5th EUROSIM Congress on Modeling and Simulation*, Seiten 198–203, Marnee la Vallee, France, 2004. EUROSIM Publishers.

[Fid91] FIDGE, C.: *Logical Time in Distributed Computing Systems*. IEEE Computer, 24(8):28–33, 1991.

[FLP85] FISCHER, M., N. LYNCH und M. PATERSON: *Impossibility of Distributed Consensus with one Faulty Process*. Journal of the ACM, 32(2):374–382, 1985.

[FLS01] FEKETE, A., N. LYNCH und A. SHVARTSMAN: *Specifying and Using a Partitionable Group Communication Service*. ACM Trans. Comput. Syst., 19(2):171–216, 2001.

[gdc10] *Google Docs*. (Link besucht am 29.05.2010), Mai 2010. `http://docs.google.com`.

[GHOS96] GRAY, J., P. HELLAND, P. O'NEIL und D. SHASHA: *The Dangers of Replication and a Solution*. In: *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, Seiten 173–182, New York, NY, USA, 1996. ACM Press.

[Gol92] GOLDING, R. A.: *Weak-Consistency Group Communication and Membership*. Doktorarbeit, University of California at Santa Cruz, Santa Cruz, CA, USA, 1992.

[Gol05] GOLLMICK, C.: *Konzept, Realisierung und Anwendung nutzerdefinierter Replikation in mobilen Datenbanksystemen*. Doktorarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, Jena, Deutschland, 2005.

[Gra10] GRAVELEY, A.: *Tomboy*. (Link besucht am 29.05.2010), Mai 2010. `http://www.gnome.org/projects/tomboy/`.

[Ham95] HAMMOND, B.: *Wingman, A Replication Service for Microsoft Access and Visual Basic*. Technischer Bericht, Microsoft White Paper, 1995.

[HB99] HEMANT, S. und P. B. R. BADRINATH: *Conflict Resolution and Reconciliation in Disconnected Databases*. In: *DEXA '99: Proceedings of the 10th International Workshop on Database & Expert Systems Applications*, Seite 76, Washington, DC, USA, 1999. IEEE Computer Society.

[HB06] HINZE, A. und G. BUCHANAN: *The Challenge of Creating Cooperating Mobile Services: Experiences and Lessons Learned*. In: *Proceedings of the Twenty-Ninth Australasian Computer Science Conference (ACSC 2006)*, Seiten 207–215, 2006.

[HBR03] HÄHNER, J., C. BECKER und K. ROTHERMEL: *A Protocol for Data Dissemination in Frequently Partitioned Mobile Ad Hoc Networks*. IEEE Symposium on Computers and Communication, 0:633, 2003.

[hdb10] *HSQLDB*. (Link besucht am 10.05.2010), Mai 2010. `http://hsqldb.org/`.

[Hek06] HEKMAT, RAMIN: *Ad-hoc Networks: Fundamental Properties and Network Topologies*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[HJRW07] HAUSHERR, M., T. JAKOB, O. RODE und B. WYSS: *Replikation und Synchronisation in Oracle Database Lite*. IMVS Fokus Report, 2007.

[HKB99] HEINZELMAN, W. R., J. KULIK und H. BALAKRISHNAN: *Adaptive Protocols for Information Dissemination in Wireless Sensor Networks*. In: *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, Seiten 174–185, New York, NY, USA, 1999.

[HOTV99] HO, C., K. OBRACZKA, G. TSUDIK und K. VISWANATH: *Flooding for Reliable Multicast in Multi-hop Ad Hoc Networks*. In: *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 1999)*, Seiten 64–71, Seattle, USA, August 1999.

[HSAA03] HOLLIDAY, J., R. STEINKE, D. AGRAWAL und A. EL ABBADI: *Epidemic Algorithms for Replicated Databases*. IEEE Transactions on Knowledge and Data Engineering, 15(5):1218–1238, 2003.

[HTKR05] HÖPFNER, H., C. TÜRKER und B. KÖNIG-RIES: *Mobile Datenbanken und Informationssysteme. Konzepte und Techniken*. dpunkt.verlag Heidelberg, 2005.

[Hup09] HUPFELD, F.: *Causal Weak-Consistency Replication - A Systems Approach*. Doktorarbeit, Humboldt-Universität zu Berlin, Berlin, Deutschland, 2009.

[JBRAS03] JARDOSH, A., E. M. BELDING-ROYER, K. C. ALMEROTH und S. SURI: *Towards Realistic Mobility Models for Mobile Ad hoc Networks*. In: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom 2003)*, Seiten 217–229, San Diego, USA, September 2003.

[JM93] JAJODIA, S. und R. MUKKAMALA: *Measuring the Effect of Commutative Transactions on Distributed Database Performance*. Inf. Sci., 68(1-2):91–111, 1993.

[JM96] JOHNSON, D. B. und D. A. MALTZ: *Dynamic Source Routing in ad-hoc Wireless Networks*. In: *Mobile Computing*, Band 353. Kluwer Academic Publishers, 1996.

[JMR97] JAGADISH, H. V., I. S. MUMICK und M. RABINOVICH: *Scalable Versioning in Distributed Databases with Commuting Updates*. In: *ICDE*, Seiten 520–531, 1997.

[KBH+88] KAWELL, L., S. BECKHARDT, T. HALVORSEN, R. OZZIE und I. GREIF: *Replicated Document Management in a Group Communication System*. In: *CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, Seite 395, New York, NY, USA, 1988. ACM Press.

[KBTR02] KHELIL, A., C. BECKER, J. TIAN und K. ROTHERMEL: *An Epidemic Model for Information Diffusion in MANETs*. In: *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, Seiten 54–60, Atlanta, USA, 2002.

[KE06] KEMPER, A. und A. EICKLER: *Datenbanksysteme - Eine Einführung, 5. Auflage*. Oldenbourg, 2006.

[Kel99] KELEHER, P. J.: *Decentralized Replicated-Object Protocols*. In: *PODC '99: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, Seiten 143–151, New York, NY, USA, 1999. ACM Press.

[Khe07] KHELIL, A.: *A Generalized Broadcasting Technique for Mobile Ad Hoc Networks*. Doktorarbeit, Elektrotechnik und Informationstechnik, Universität Stuttgart, 2007.

[KM05] KOUVATSOS, D. und I. MKWAWA: *Broadcasting Methods in Mobile Ad Hoc Networks: An Overview*. In: *Proceedings of the HetNet*, UK, 2005.

[KNG+04] KOTZ, D., C. NEWPORT, R. S. GRAY, J. LIU, Y. YUAN und C. ELLIOTT: *Experimental evaluation of wireless simulation assumptions*. In: *MSWiM '04: Proceedings of the*

7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, Seiten 78–82, New York, NY, USA, 2004. ACM Press.

[KRSD01] KERMARREC, A., A. ROWSTRON, M. SHAPIRO und P. DRUSCHEL: *The IceCube Approach to the Reconciliation of Divergent Replicas*. In: *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, Seiten 210–218, New York, NY, USA, 2001. ACM Press.

[LA06] LIU, C. und P. ALBITZ: *DNS and BIND (5th Edition)*. O'Reilly Media, Inc., 2006.

[Lam78] LAMPORT, L.: *Time, Clocks, and the Ordering of Events in a Distributed System*. Commun. ACM, 21(7):558–565, 1978.

[LK00] LIM, H. und C. KIM: *Multicast Tree Construction and Flooding in Wireless ad hoc Networks*. In: *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, Seiten 61–68, New York, NY, USA, 2000. ACM Press.

[LL84] LUNDELIUS, J. und N. A. LYNCH: *An Upper and Lower Bound for Clock Synchronization*. Information and Control, 62(2/3):190–204, 1984.

[LLSG92] LADIN, R., B. LISKOV, L. SHRIRA und S. GHEMAWAT: *Providing High Availability Using Lazy Replication*. ACM Trans. Comput. Syst., 10(4):360–391, 1992.

[LW04] LINDEMANN, C. und O. P. WALDHORST: *Exploiting Epidemic Data Dissemination for Consistent Lookup Operations in Mobile Applications*. SIGMOBILE Mob. Comput. Commun. Rev., 8(3):44–56, 2004.

[Mar07] MARTENS, S.: *Erweiterung und Evaluation eines verteilten Datenbanksystems für mobile ad-Hoc Netze*. Diplomarbeit, Institut für Informatik, Freie Universität Berlin, 2007.

[Mat89] MATTERN, F.: *Virtual Time and Global States of Distributed Systems*. In: *Proceedings Workshop on Parallel and Distributed Algorithms*, Seiten 215–226, Amsterdam, Holland, 1989. Cosnard, M. et al. (eds).

[MBT04] MEIER, L., P. BLUM und L. THIELE: *Internal Synchronization of Drift-Constraint Clocks in ad-hoc Sensor Networks*. In: *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, Seiten 90–97, New York, NY, USA, 2004. ACM Press.

[mck10] *Diehl and Associates, Inc. Mckoi SQL Database*. (Link besucht am 27.05.2010), Mai 2010. `http://mckoi.com/database/`.

[Mil92] MILLS, D.: *Network Time Protocol (Version 3) Specification, Implementation and Analysis*. RFC 1305, 1992.

[mob10] *Mobiletour*. (Link besucht am 10.06.2010), Juni 2010. `http://www.handytour.com/`.

[Mol95] MOLNÁR, P.: *Modellierung und Simulation der Dynamik von Fußgängerströmen*. Doktorarbeit, II.Institut für Theoretische Physik der Universität Stuttgart, 1995.

[MRBV05] MAHADEVAN, P., A.O RODRIGUEZ, D. BECKER und A. VAHDAT: *MobiNet: a Scalable Emulation Infrastructure for ad hoc and Wireless Networks*. In: *WiTMeMo '05: Papers presented at the 2005 workshop on Wireless traffic measurements and modeling*, Seiten 7–12, Berkeley, CA, USA, 2005. USENIX Association.

[MS06] MUTSCHLER, B. und G. SPECHT: *Mobile Datenbanksysteme: Architektur, Implementierung, Konzepte (Xpert.press)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[ns10] *The Network Simulator - ns-2*. (Link besucht am 05.01.2010), Januar 2010. `http://nsnam.isi.edu/nsnam/index.php/Main_Page`.

[NTCS99] NI, S.-Y., Y.-C. TSENG, Y.-S. CHEN und J.-P. SHEU: *The Broadcast Storm Problem in a Mobile Ad Hoc Network*. In: *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, Seiten 151–162, Seattle, USA, August 15–19 1999.

[OS05] O'BRIEN, J. und M. SHAPIRO: *An Application Framework for Collaborative, Nomadic Applications*. Rapport de recherche RR-5745, Inria, Rocquencourt (France), November 2005.

[OV99] O., TAMER M. und PATRICK V.: *Principles of Distributed Database Systems (2nd Edition)*. Prentice Hall, Januar 1999.

[PR99] PERKINS, C. E. und E. M. ROYER: *Ad-hoc On-Demand Distance Vector Routing*. In: *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, Seiten 90–100, New Orleans, USA, Februar 1999.

[PS00] PAPADOPOULI, M. und H. SCHULZRINNE: *Seven Degrees of Separation in Mobile ad hoc Networks*. In: *Proceedings of IEEE Global Telecommunications Conference(GLOBECOM)*., San Francisco, California, 2000.

[PS01] PAPADOPOULI, M. und H. SCHULZRINNE: *Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination Among Mobile Devices*. In: *ACM*

International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc), Long Beach, California, Oktober 2001.

[PSM03] PREGUIÇA, N., M. SHAPIRO und C. MATHESON: *Semantics-Based Reconciliation for Collaborative and Mobile Environments*. In: *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Seiten 38–55. Springer-Verlag GmbH, Januar 2003.

[PST+97] PETERSEN, K., M. J. SPREITZER, D. B. TERRY, M. M. THEIMER und A. J. DEMERS: *Flexible Update Propagation for Weakly Consistent Replication*. In: *SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles*, Seiten 288–301, New York, NY, USA, 1997. ACM Press.

[RÖ6] RÖSCH, H.: *Implementierung und experimentelle Untersuchung einer Replikationskomponente für mobile Endgeräte in mobilen Ad-hoc Netzwerken*. Diplomarbeit, Institut für Informatik, Freie Universität Berlin, 2006.

[Rat98] RATNER, D. H.: *Roam: A Scalable Replication System for Mobile and Distributed Computing*. Doktorarbeit, University of California, Los Angeles, 1998.

[Ric92] RICHARD, G. G.: *FICUS: A Very Large Scale Reliable Distributed File System*. Doktorarbeit, University of California, Los Angeles, 1992.

[RKHH04] RYU, J., M. KIM, S. HWANG und K. HAN: *An Adaptive Probabilistic Broadcast Scheme for ad-Hoc Networks*. In: *High Speed Networks and Multimedia Communications: Proceedings of the 7th IEEE Int. Conference HSNMC*, Seiten 646–654, 2004.

[Röm01] RÖMER, K.: *Time Synchronization in ad hoc Networks*. In: *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, Seiten 173–182, New York, NY, USA, 2001. ACM Press.

[Sai01] SAITO, Y.: *Consistency Management in Optimistic Replication Algorithms*. Technical Report, University of Washington, 2001.

[SBH07] SCHOLZ, M., F. BREGULLA und A. HINZE: *Using Physical Clocks for Replication in MANETs*. In: *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW' 07)*, White Plains, NY, USA, März 2007.

[Sch06] SCHRÖDER, Y.: *Datenverteilung in mobilen ad-hoc-Netzen, Implementierung und Benutzung verschiedener Protokolle*. Examensarbeit, Institut für Informatik, Freie Universität Berlin, Mai 2006.

[SKK+90] SATYANARAYANAN, M., J. J. KISTLER, P. KUMAR, M. E. OKASAKI, E. H. SIEGEL und D. C. STEERE: *Coda: A Highly Available File System for a Distributed Workstation Environment*. IEEE Trans. Comput., 39(4):447–459, 1990.

[SL98] SPENCER, H. und D. LAWRENCE: *Managing Usenet*. O'Reilly, 1998.

[Spi01] SPITZER, F.: *Principles of Random Walk*. Springer-Verlag, New York, 2001.

[SS05] SAITO, Y. und M. SHAPIRO: *Optimistic Replication*. ACM Comput. Surv., 37(1):42–81, 2005.

[TF06] TONGUZ, O. K. und G. FERRARI: *Ad Hoc Wireless Networks: A Communication-Theoretic Perspective*. Wiley, 2006.

[THB+02] TIAN, J., J. HAEHNER, C. BECKER, I. STEPANOV und K. ROTHERMEL: *Graph-Based Mobility Model for Mobile Ad Hoc Network Simulation*. In: *Proceedings of the 35th Annual Simulation Symposium (SS '02)*, Seiten 337–344, San Diego, USA, April 2002.

[TNS03] TSENG, Y., S. NI und E. SHIH: *Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network*. IEEE Transactions on Computers, 52(5):545–557, 2003.

[Tsc06] TSCHIRNER, D.: *Experimentelle und analytische Untersuchungen des Area Graph-based Mobility Models*. Diplomarbeit, Institut für Informatik, Freie Universität Berlin, 2006.

[TTP+95] TERRY, D. B., M. M. THEIMER, KARIN PETERSEN, A. J. DEMERS, M. J. SPREITZER und C. H. HAUSER: *Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System*. In: *SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles*, Seiten 172–182, New York, NY, USA, 1995. ACM Press.

[Urb08] URBANO, R.: *Oracle Streams Replication Administrator's Guide, 10g Release 2*. Oracle, 2008.

[Var01] VARGA, A.: *The OMNeT++ Discrete Event Simulation System*. In: *Proceedings of the 15th European Simulation Multiconference (ESM '2001)*, Prague, Czech Republic, Juni 2001.

[WC02] WILLIAMS, B. und T. CAMP: *Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks*. In: *Proceedings of the 3rd ACM International Symposium on Mobile*

Ad Hoc Networking & Computing (MOBIHOC '02), Seiten 194–205, Lausanne, Switzerland, Juni 2002.

[wik09a] *Wikipedia Statistics*. (Link besucht am 18.05.2009), Mai 2009. `http://stats.wikimedia.org/EN/TablesDatabaseEdits.htm`.

[wik09b] *Wikipedia Statistics*. (Link besucht am 18.05.2009), Mai 2009. `http://stats.wikimedia.org/EN/TablesPageViewsMonthly.htm`.

[WLW09] WEINGÄRTNER, E., H. LEHN und K. WEHRLE: *A Performance Comparison of Recent Network Simulators*. Proceedings of the IEEE International Conference on Communications, 2009.

[ZAZ04] ZHAO, W., M. AMMAR und E. ZEGURA: *A Message Ferrying Approach for Data Delivery in Sparse Mobile ad hoc Networks*. In: *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, Seiten 187–198, New York, NY, USA, 2004. ACM Press.

Anhang A

Glossar

FIFO Ordnung S. 25

Flutungsprotokoll S. 10

Forscherszenario S. 12

Gesamtverzögerung S. 87

Geschwisterknoten S. 99

Gruppenansicht S. 31

Gruppentransaktion S. 31

Gruppenverwaltung S. 31

Implizites Commit S. 118

Implizite Einigung S. 27

Indirekte Verbindung (mobiler Knoten) S. 10

Indirekter Konflikt S. 101

Intervallbestimmtes Commit S. 117

insertGraph S. 107

Katastrophneszenario S. 12

Kausale Ordnung S. 26

Kennungslisten S. 62

Kindknoten S. 99

Kommutative Schreiboperation S. 96

Kommutative Transaktion S. 96

Kommutativer Konflikt S. 101

Kommutativitätsgruppe S. 96

Konflikt (Allgemein) S. 23

- direkt S. 100
- indirekt S. 101
- kommutativ S. 101
- nicht kommutativ S. 101
- semantisch S. 21
- syntaktisch S. 21

Konflikt (Systemmodell) S. 29, S. 102

Konfliktelement S. 95

Konfliktgruppe S. 101

Konfliktrate S. 159

Konfliktwahrscheinlichkeit S. 139

Leseoperation S. 28

Lokale Datenbank S. 28

Lokale Transaktion S. 29

Makromodell S. 52

MANET S. 9

Manuelles Commit S. 117

Master S. 14

Merge S. 24

Mikromodell S. 52

Mobile ad-hoc Netz S. 9

Mobiler Knoten S. 9

Multi-hop Verbindung → Indirekte Verbindung

Multicast S. 10

Anhang B

Weitere Untersuchungen

In diesem Abschnitt werden die Ergebnisse weiterer Untersuchung der Evaluation (siehe Kapitel 10) aufgeführt.

B.1 Katastrophenszenario mit gleichbleibender Netztopologie

Im Gegensatz zum Katastrophenszenario aus Abschnitt 10.4.2 ändert sich die Anzahl der Netzknoten nicht, sondern nur die Anzahl der Teilnehmer der Replikationsgruppe. Die anderen Teilnehmer im Netz nehmen nicht an der Replikation teil. Die Ergebnisse dieses Katastrophenszenarios sind in Abbildung B.1 dargestellt.

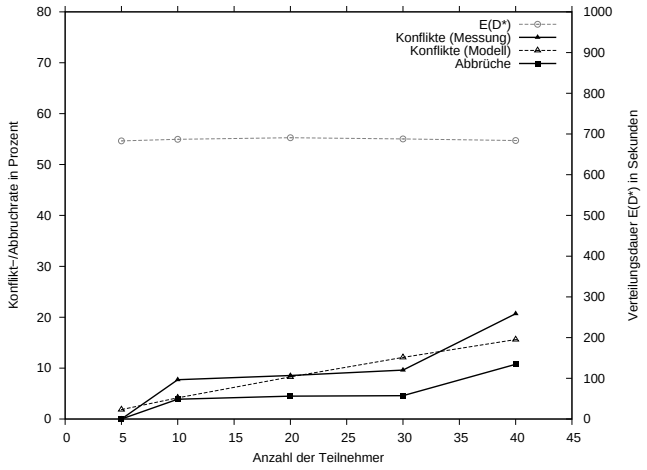


Abbildung B.1: Ergebnis des zweiten Katastrophenszenarios (gleichbleibende Netztopologie)

Beobachtung Aufgrund der gleichbleibenden Netzdichte ergibt sich ebenfalls eine gleichbleibende Verteilungsdauer von fast 700s. Im Vergleich zu den Ergebnissen des vorigen Szenarios sind nur die Konfliktraten für 5 und 20 Teilnehmer deutlich reduziert. Für 10 und 30 Teilnehmer ergeben sich

nur geringfügige Verbesserungen. Das Ergebnis für 40 Teilnehmer stimmt erwartungsgemäß (bis auf Messungenauigkeiten) mit dem Ergebnis des letzten Szenarios überein.

Die Abbruchrate entspricht wie zuvor jeweils ungefähr der Hälfte der Konfliktrate. Die durchschnittliche Abweichung zwischen den gemessenen und den von der Konfliktanalyse berechneten Werten beträgt 2,8 Prozentpunkte.

Diskussion Der nur geringe Einfluss der höheren Netzdichte auf die Konfliktrate bei 30 Teilnehmern kann dadurch erklärt werden, dass sich die Verteilungsdauer von 958s (erstes Katastrophenszenario) auf 688s reduziert. Dies ist im Gegensatz zu den Unterschieden, die sich bei den anderen Gruppengrößen ergeben, eine relativ geringe Differenz.

Die reduzierten Konfliktraten bei 5 und 20 Teilnehmern sind erwartungsgemäß auf die großen Unterschiede in der Verteilungsdauer zurückzuführen. Die nur relativ geringe Verbesserung der Konfliktrate um 1,2 Prozentpunkte bei 10 Teilnehmern ist dadurch nicht zu erklären. Betrachtet man den Standardfehler, so stellt man fest, dass dieser in beiden Katastrophenszenarien für die Werte mit 10 Teilnehmern relativ hoch ist. Die Standardfehler von 1,5 Prozentpunkten (erstes Katastrophenszenario) und 1,7 Prozentpunkten (dieses Szenario) sind im Gegensatz zum durchschnittlichen Standardfehler beider Szenarien (1,0 Prozentpunkte) relativ hoch. Aus diesem Grund lässt sich der relativ hohe Wert für die Konfliktrate von 10 Teilnehmern wahrscheinlich auf den Standardfehler zurückführen. Auch der Wert der Konfliktanalyse ist für 10 Teilnehmer deutlich geringer und unterstützt diese Vermutung.

B.2 Touristenszenario

Das Touristenszenario wird mit Hilfe des Area Graph-based Bewegungsmodells modelliert. Das Modell ist in Abbildung B.2 gezeigt und soll einen innerstädtischen Bereich mit verschiedenen Sehenswürdigkeiten und Plätzen darstellen. Die Touristen bewegen sich innerhalb der Cluster nur zu Fuß (1,3 $\frac{\mathrm{m}}{\mathrm{s}}$) und verweilen dort jeweils 15 - 60 Minuten. Zwischen den Clustern können die Touristen öffentliche Verkehrsmittel benutzen und bewegen sich somit schneller.

Die Touristen verwenden ein Informationssystem, das sie in erster Linie lesend benutzen und in das sie nur selten neue Information einfügen. Das emulierte Zeitfenster des Szenarios beträgt vier Stunden. Alle weiteren Parameter sind in Tabelle B.1 angegeben. Die Ergebnisse des Touristenszenarios sind in Abbildung B.3 dargestellt.

Beobachtung Die Kurve der durchschnittlichen Verteilungsdauer fällt von fast 3000s (10 Teilnehmer) bis auf 2000s für eine Gruppengröße von 40 Teilnehmern. Das Gefälle der Kurve ist zwischen den Werten von 10 und 20 Teilnehmern deutlich höher als im restlichen Bereich. Die Konfliktrate bei 30 Teilnehmern ist mit einem Wert von 19,8% auffällig hoch. Lässt man diesen Wert außer acht, so

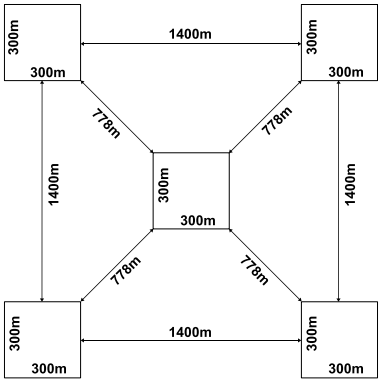


Abbildung B.2: Bewegungsmodell des Touristenszenarios

Anzahl der Teilnehmer	10, 20, 30, 40
Datenbankgröße	150 Elemente
Transaktionsgröße	1 Operation
Transaktionsintervall	7200s

Tabelle B.1: Parameter des Touristenszenarios

ergibt sich eine annähernd lineare Entwicklung der Konfliktrate von 0% (10 Teilnehmer) bis 14,1% (40 Teilnehmer).

Die Abbruchrate ist für alle Teilnehmerzahlen ungefähr halb so groß wie die jeweilige Konfliktrate und hat bei 30 Teilnehmern einen Maximalwert von 11,0%. Die durchschnittliche Abweichung zwischen den gemessenen und den von der Konfliktanalyse berechneten Werten beträgt 4,2 Prozentpunkte.

Diskussion Die relativ geringe Reduktion der Verteilungsdauer bei zunehmender Gruppengröße lässt sich auf die Eigenschaften des Area Graph-based Modells zurückführen. Im Gegensatz zum Random Waypoint Modell sind die mobilen Knoten auf mehrere Cluster verteilt, deshalb erhöht eine größere Anzahl von Teilnehmern in erster Linie nur die Dichte innerhalb der Cluster. Dies wirkt sich auf die Datenverteilung nur geringfügig aus, da sich die mobilen Knoten immer noch von einem Cluster zum nächsten bewegen müssen, um die Daten weiter zu verteilen.

Aufgrund des langen Transaktionsintervalls des Szenarios resultieren die Schwankungen der Messungen in einem höheren Standardfehler von durchschnittlich 4,4 Prozentpunkten. Dies erklärt auch zum Teil die hohe Konfliktrate bei 30 Teilnehmern, die mit 10 Prozentpunkten relativ stark von den Werten der Konfliktanalyse abweicht. In erster Linie ist der hohe Wert jedoch auf die Verteilungsdauer zurückzuführen, die bei 20 und 30 Teilnehmern praktisch identisch ist. Somit wirkt sich nur

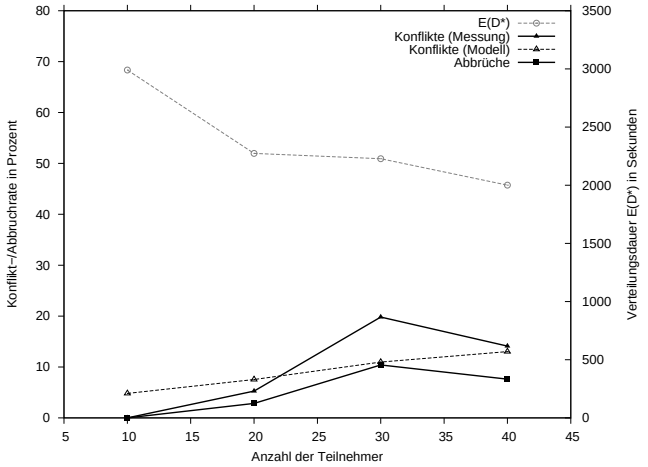


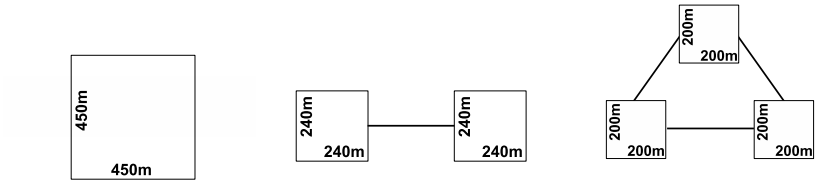
Abbildung B.3: Ergebnisse des Touristenszenarios

der negative Einfluss einer größeren Gruppe auf die Konfliktrate aus. Bei einer Gruppengröße von 40 Teilnehmern reduziert sich die Verteilungsdauer erneut, was die Konfliktrate positiv beeinflusst und auf 14,1% reduziert.

Im Gegensatz zum Katastrophenszenario stellt man fest, dass sich kleinere Änderungen der Verteilungsdauer bereits relativ stark auf die Konfliktrate auswirken. Dieses Verhalten ist auf das Verhältnis des sich für die gesamte Gruppe ergebenden Transaktionsintervalls zum Verteilungsintervall zurückzuführen. Wenn die Längen der beiden Intervalle in der gleichen Größenordnung liegen, so wirken sich leichte Veränderungen eines Intervalls stärker aus, als wenn die Längen sehr unterschiedlich sind (siehe Abbildung 9.6 auf S. 151). Im Touristenszenario sind die Längen der beiden Intervalle nicht so unterschiedlich wie bei anderen Szenarien, wodurch der relativ starke Einfluss des Verteilungsintervalls zu erklären ist.

B.3 Bewegungsmodelle mit gleicher Verteilungsdauer

In dieser Untersuchung werden verschiedene Bewegungsmodelle verglichen, welche dieselbe durchschnittliche Verteilungsdauer $E(D^*)$ besitzen. Die unterschiedlichen Topologien der Bewegungsmodelle sind in Abbildung B.4 abgebildet. Da die Verteilungsdauer eine Zielgröße darstellt, kann diese nicht künstlich festgelegt werden, sondern wird experimentell bestimmt. Aus diesem Grund ergeben sich bei den Zeiten der Verteilungsdauer der einzelnen Szenarien geringe Unterschiede. Die einzelnen Werte betragen 275s (Random Waypoint) 273s (AGM 2 Cluster) und 279s (AGM 3 Cluster). Alle weiteren Parameter sind in Tabelle B.2 angegeben. Die Untersuchungsergebnisse für Bewegungsmodelle mit gleicher Verteilungsdauer sind in Abbildung B.5 dargestellt.



(a) Random Waypoint (b) AGM - zwei Cl. (c) AGM - drei Cl.

Abbildung B.4: Topologie der Bewegungsmodelle mit gleicher Verteilungsdauer

Anzahl der Teilnehmer	20
Datenbankgröße	400 Elemente
Transaktionsgröße	1 Operation
Transaktionsintervall	480s

Tabelle B.2: Parameter der Untersuchungsreihe mit gleicher Verteilungsdauer

Beobachtung Die Konfliktraten der Messung betragen 6,9% (Random Waypoint), 4,4% (AGM - zwei Cluster) und 5,4% (AGM - drei Cluster). Alle drei Werte der Konfliktanalyse ähneln sich (5,2% bis 5,3%) und weichen bei den Area Graph-based Modellen nur wenig von den Messwerten ab (maximal 0,8 Prozentpunkte). Die Abweichung vom Random Waypoint Modell ist mit 1,6 Prozentpunkten doppelt so hoch.

Diskussion Die Konfliktraten der AGMs ähneln sich und die Unterschiede liegen im Bereich des durchschnittlichen Standardfehlers des Szenarios (1,3 Prozentpunkte). Die Konfliktrate des Random Waypoint Modells ist zwar etwas höher, der Unterschied ist jedoch deutlich geringer als bei den Untersuchungen in Abschnitt 10.5.2. Somit entspricht das Ergebnis den in den Hypothesen aufgestellten Erwartungen.

Da für die Untersuchung Szenarien mit möglichst gleicher Verteilungsdauer gewählt wurden, ergeben sich auch fast identische Werte für die Konfliktanalyse. Wie bereits in den anderen Untersuchungen beobachtet, ergibt sich die größte Abweichung der Konfliktanalyse für das Random Waypoint Modell, da dessen Verteilung der Verteilungsdauerwerte stärker von der Exponentialverteilung abweicht als die des AGMs.

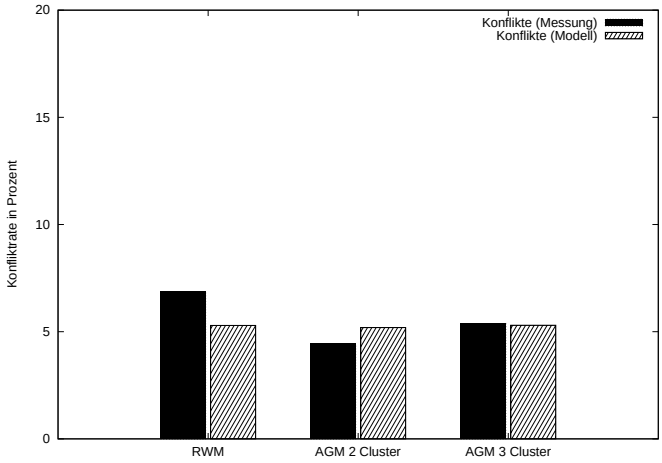


Abbildung B.5: Ergebnisse für die unterschiedlichen Bewegungsmodelle mit gleicher Verteilungsdauer

Anhang C

Anhang gemäß Promotionsordnung

Abstract

Small computing devices combined with wireless networking technologies nowadays give rise to new kinds of mobile networks which feature different characteristics than conventional wired networks. The question arises whether conventional networking techniques are suitable to support applications to be deployed in such mobile networks.

This work addresses the aforementioned question by evaluating the application of optimistic replication techniques to mobile ad-hoc networks (MANETs). MANETs are decentral and wireless networks (WLANs) without a fixed infrastructure which are spontaneously formed among mobile devices (e.g., smartphones, PDAs, laptops). Due to the mobility and unreliability of mobile nodes, MANETs are highly dynamic networks. Therefore, long-lasting connection losses between mobile nodes may occur.

To guarantee high data availability in the presence of broken network connections, optimistic replication techniques can be used. Here, a replication node updates the data first locally and then sends it to the other participants. The downside of this approach is that concurrent updates can occur causing write conflicts.

So far, experience of replication in MANETs is limited, therefore we developed a mobile replication system and evaluated its performance in mobile ad-hoc networks. The techniques of the replication system have been designed or adapted for use in MANETs because conventional replication systems are not suitable. MANET scenarios are motivated by the lack of infrastructure, e. g., due to a disaster. Thus, it is not reasonable to use a central replication server, for example, connected to the mobile devices via UMTS or GPRS.

To take the mobility of MANET scenarios into account, we considered mobility models and data dissemination protocols. We developed the *Area Graph-based Mobility Model* that considers the macroscopic as well as the microscopic motion of mobile nodes. We also developed two dissemination protocols, an *adaptive push protocol* and an *adaptive pull protocol*. Tests with simulated MANET scenarios show that both protocols produce less network load than existing dissemination protocols.

The conflict detection and resolution techniques of our approach are based on the dependencies of the transactions. The dependencies are managed by the *Predecessor Graph*: a data structure enhancing a predecessor list. It is not restricted to operation management but also supports transactions. Furthermore, the data structure also allows commutative transactions. As a proof of concept, we develop the prototype replication system SYMORE and test it in small scenarios on PDAs and laptops. A mobile wiki and a mobile puzzle game have also been successfully implemented based on SYMORE.

Finally, we evaluated these replication techniques experimentally (with emulations), as well as analytically by the number of conflicting transactions. The results show a small number of conflicting transactions for the majority of the tested scenarios and thus a small number of transaction aborts as well. In conclusion, we present an optimistic replication approach for mobile ad-hoc networks and demonstrate that it is suitable for use in small to medium-sized MANET scenarios with up to 40 participants.