Diplomarbeit

Untersuchung des Einflusses von Positionsinformationen auf Service Discovery in mobilen Ad-hoc Netzwerken

von
Matthias Bier

21. Mai 2008

Gutachter:
Prof. Dr. Heinz Schweppe
Prof. Dr. rer. nat. Mesut Güneş

Betreuerin:
Katharina Hahn

Freie Universität Berlin
Fachbereich Informatik
AG Datenbanken und Informationssysteme

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, 21. Mai 2008 _____

Zusammenfassung

Die Dienstfindung ist ein allgemeines Problem in Netzwerken. In *mobilen Ad-hoc Netzwerken* (*MANETs*) kommt diesem Problem jedoch ein besonderer Stellenwert zu, da bekannte Lösungsansätze für MANETs nicht geeignet sind. In dieser Arbeit soll experimentell, mit Hilfe des Netzwerksimulators *NS-2*, untersucht werden, inwieweit die Positions- und Bewegungsinformation mobiler Knoten bei der Dienstfindung in MANETs hilfreich ist.

Dazu wird ein existierendes Dienstfindungsprotokoll, das Protokoll *Group-based Service Discovery* (GSD), erweitert, so dass es die Mobilitätsinformationen der einzelnen Knoten in die Dienstfindung einbezieht. Das Protokoll passt seine Konfiguration selbstständig an die Bewegung der Knoten im Netzwerk an und ermöglicht den parallelen Einsatz zweier Routingverfahren. So wird in bestimmten Situationen statt dem herkömmlichen Routing geographisches Routing verwendet.

Das entworfene mGSD Protokoll wird im NS-2 implementiert, um schließlich durch Simulationen den Einfluss der Positions- und Bewegungsinformation auf die Dienstfindung zu evaluieren.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die zunehmende Verbreitung mobiler Kleingeräte wie Handys, PDAs oder Laptops macht *mobile Ad-hoc Netzwerke* (*MANETs*) zu einem wichtigen Forschungsgebiet innerhalb der Informatik. Mit immer kleiner werdenden Ausmaßen, sinkenden Kosten und steigenden Rechenkapazitäten wird der Einsatz und die Vernetzung dieser Geräte im alltäglichen Leben gefördert.

1.1 Motivation

Dienste spielen in solchen mobilen Ad-hoc Netzwerken eine große Rolle. So bietet ein Netzwerkknoten einen bestimmten Dienst an und ein anderer Netzwerkknoten möchte diesen Dienst nutzen. Hier kann man sich ein Touristenszenario vorstellen: Ein Tourist sucht Informationen über Sehenswürdigkeiten. Ein anderer Tourist hat schon einige Sehenswürdigkeiten besucht und bietet nun Informationen zu diesen an. Man kann also zwei Rollen unterscheiden: Diensterbringer und Dienstnutzer.
Das allgemeine Problem in Netzwerken jeglicher Art ist es, diese beiden, Diensterbringer und Dienstnutzer, zusammenzubringen. Ein angebotener Dienst muss nutzbar und damit auch auffindbar sein. Wenn der Dienstnutzer den angebotenen Dienst nicht suchen und finden kann, existiert der Dienst für ihn nicht. Normalerweise wird deshalb noch eine dritte Rolle eingeführt: Dienstverzeichnisse. Diese fungieren als zentrales Bindeglied zwischen Diensterbringer und Dienstnutzer. Aufgrund der dynamischen Struktur der MANETs ist dieser Ansatz jedoch für MANETs nicht geeignet, da Touristen nicht als ständig verfügbare Dienstverzeichnisse fungieren können. Dienstfindungsprotokolle für MANETs versuchen dieses Problem zu lösen und das Suchen und Auffinden von Diensten zu ermöglichen.

Die Bewegung der einzelnen Teilnehmer im MANET beeinflusst ein Dienstfindungsprotokoll maßgebend. Neben der Netzwerkgröße und der Anzahl der Teilnehmer ist die Bewegung einer der zentralen Faktoren für die Konfiguration eines Dienstfindungsprotokolls. Darum bietet es sich an, die Positions- und Bewegungsinformationen der Knoten im MANET in die Dienstfindung einzubeziehen. Wenn die Bewegung der Knoten im MANET bekannt ist, kann sich ein Dienstfindungsprotokoll selbstständig an die Gegebenheiten des Netzwerkes anpassen. Neben der Konfiguration des Protokolls bietet es sich auch an, die Positions- und Bewegungsdaten der Knoten zu benutzen, um unter bestimmten Umständen geographisches Routing einzusetzen.

1.2 Aufgabenstellung

In dieser Diplomarbeit wird untersucht, welchen Einfluss die Positions- und Bewegungsinformationen der einzelnen Knoten in einem MANET auf die Dienstfindung (*Service Discovery*) haben. So soll herausgefunden werden, ob die Mobilitätsinformationen dazu verwendet werden können, die Dienstfindung zu verbessern. Zu diesem Zweck wird das Dienstfindungsprotokoll *Group-based Service Discovery* (GSD) so erweitert, dass es die Mobilitätsinformationen der einzelnen Knoten in die Dienstfindung einbezieht. Das erweiterte Protokoll, *mGSD* genannt, wird auf Basis einer vorhandenen Implementierung des GSD Protokolls im Netzwerksimulator *NS-2* [4] implementiert, um durch Simulationen die Vor- und Nachteile einer solchen Strategie zu evaluieren.

1.3 Gliederung der Arbeit

Die weitere Arbeit gliedert sich wie folgt: In Kapitel 2 wird eine Einführung in das Gebiet der Service Discovery in MANETs gegeben. Es werden die grundlegenden Eigenschaften der MANETs vorgestellt und die Problematik der Dienstfindung erläutert. Schließlich wird ein Überblick über vorhandene Lösungsansätze gegeben.

Das GSD Protokoll, welches in dieser Arbeit als Basis für die Erweiterungen dient, wird in Kapitel 3 eingeführt. Zudem werden die nötigen Anpassungen an der Implementierung des GSD Protokolls im NS-2 und an den verwendeten Routingprotokollen beschrieben.

Kapitel 4 führt dann das mGSD Protokoll ein und beschreibt die Konzeption und Umsetzung der einzelnen Erweiterungen. Schließlich werden auch die Evaluationsweisen für die Erweiterungen erläutert.

Die eigentliche Evaluation wird dann in Kapitel 5 beschrieben. Neben einer Beschreibung der Simulationsumgebung werden die Simulationsszenarien sowie wichtige Messgrößen beschrieben. Den restlichen Teil des Kapitels nehmen die Evaluationsergebnisse ein.

Schließlich wird in Kapitel 6 eine abschließende Zusammenfassung und Bewertung vorgenommen. Bestandteil dieses Kapitels sind auch mögliche zukünftige Forschungsbereiche.

2 Dienstfindung in MANETs

In diesem Kapitel wird eine Einführung in das Gebiet der Service Discovery gegeben. Neben einer Beschreibung der Problematik wird ein Überblick über verschiedene Service Discovery Strategien geboten.

2.1 Problembeschreibung

2.1.1 MANETs

MANETs sind Netzwerke, die von mobilen Geräten gebildet werden. Während in traditionellen Funknetzen, zum Beispiel den Handy-Netzen, mobile Knoten miteinander im Allgemeinen nur über feste Knoten, die Basisstationen, kommunizieren, steht in MANETs keine solche feste Infrastruktur zur Verfügung. Vielmehr können die einzelnen mobilen Geräte über eine Funkschnittstelle direkt (*single hop* Netzwerke) bzw. über ein oder mehrere Zwischenknoten mit Routingfunktionalität (*multi hop* Netzwerke) miteinander kommunizieren.
Einsatzgebiete von MANETs finden sich besonders dort, wo keine feste Infrastruktur vorhanden bzw. gewünscht ist. So kann man sich ein Netzwerk aus mobilen Sensorknoten in einem Katastrophengebiet oder auch den Dateiaustausch unter Teilnehmern einer Konferenz vorstellen.

Bei der Betrachtung von MANETs, wie auch bei traditionellen Funknetzen, müssen im Besonderen folgende Faktoren berücksichtigt werden:

Dynamik Die Knoten im Netzwerk sind, im Gegensatz zu Festnetzen, mobil und bewegen sich in nicht vorhersagbarer Weise. Zudem können die Knoten dem Netzwerk spontan beitreten bzw. es wieder verlassen. Diese zwei Faktoren

führen zu einer dynamischen Netzwerkstruktur, die man nicht vorhersagen kann. Sie kann sich spontan und eventuell sehr schnell ändern.

Ressourcen Die Ressourcen der typischen Geräte im MANET sind sehr begrenzt. Rechen- und Speicherkapazitäten haben sich zwar in den letzten Jahren kontinuierlich verbessert, sind aber keineswegs mit denen von herkömmlichen Computersystemen vergleichbar.
Besonders problematisch sind jedoch die Energiereserven der meist batteriebetriebenen Geräte. In diesem Bereich hat sich in den letzten Jahren vergleichsweise wenig getan, so dass die Energieversorgung immer noch eine der stärksten Einschränkungen dieser Geräte ist.

Netzqualität Die Geräte verwenden für jegliche Kommunikation Funkverbindungen, wie beispielsweise *WLAN* oder *Bluetooth*. Durch die Störanfälligkeit der Funkkommunikation müssen Verbindungsabbrüche als Normalfall betrachtet und dementsprechend behandelt werden. Auch wenn Verbindungen über mehrere Zwischenknoten möglich sind, steigt bei diesen die Gefahr von Kommunikationsstörungen erheblich.

2.1.2 Dienstfindung

Service Discovery, also das Auffinden von verfügbaren Diensten im Netz, ist eine grundlegende Vorraussetzung für *Serviceorientierte Architekturen* (*SOA*) jeder Art. Dienstanbieter (*Service Provider*) müssen ihre Dienste im Netz bekannt machen können und Dienstnutzer benötigen die Möglichkeit nach Diensten zu suchen. In dieser Arbeit wird evaluiert, inwieweit die Positions- und Bewegungsinformationen der einzelnen Knoten in einem MANET genutzt werden können, um Service Discovery zu verbessern. Dazu wird ein existierendes Dienstfindungsprotokoll, das GSD Protokoll, erweitert, so dass es die Mobilitätsinformationen nutzt. Das so entworfene mGSD Protokoll wird dann durch Simulation im NS-2 mit dem Orginalprotokoll verglichen.

2.2 Vorhandene Ansätze

Die Ansätze für Service Discovery in MANETs basieren meist auf Techniken aus dem Bereich der Netzwerke mit fester Infrastruktur. Einen kurzen Überblick über die verschiedenen klassischen Ansätze und deren Problematik in Bezug auf MANETs wird im nächsten Abschnitt gegeben. Im darauf folgenden Abschnitt werden dann einige explemarische Ansätze vorgestellt, die speziell für MANETs entwickelt wurden.

2.2.1 Klassische Ansätze

Die klassischen Ansätze lassen sich in zwei Kategorien unterteilen:

Verzeichnis-basierter Ansatz

Das Verwenden einer zentralisierten Struktur, wie z.B. beim *Service Location Protocol* [9], ist ein klassischer Ansatz für Service Discovery. Die Beschreibungen der angebotenen Dienste werden *proaktiv* im Netz verteilt, also ohne vorherige Anfrage eines Knotens (*Clients*). Dabei dienen ein oder mehrere Knoten als zentrale Verzeichnisknoten (*Directory Agents*) für vorhandene Dienste. Ein Knoten, der einen bestimmten Dienst sucht, wendet sich dann an einen der zentralen Verzeichnisknoten und erhält die passende Antwort. Der Einsatz eines solchen Systems in einem MANET ist aber wenig sinnvoll, da die Verfügbarkeit der einzelnen Knoten nicht sicher ist.

Verzeichnisloser Ansatz

Das sogenannte Fluten (*Broadcasting*) ist ein zweiter klassischer Ansatz für Service Discovery. Es kommt ohne eine zentrale Struktur aus. Bei diesem *reaktiven* Verfahren speichern die Knoten eigene Dienste nur lokal und verteilen sie nicht im Netz. Ein Knoten, der einen Dienst sucht, muss daher seine Anfrage an alle seine

Nachbarn senden. Die Empfänger prüfen, ob sie selber einen passenden Dienst anbieten, und antworten gegebenenfalls. Sie leiten die Anfrage aber auch an alle ihre Nachbarn weiter. Die Vorteile dieses Verfahrens liegen in der leichten Erweiterbarkeit des Netzwerkes und der Robustheit. Knotenausfälle oder eine sich ändernde Netzstruktur beeinflussen das Verfahren wenig. Von Nachteil im Hinblick auf MANETs ist aber die erzeugte Nachrichtenflut. Diese kann vor allem schmalbandige Netze schnell überlasten. Eine andere Variante dieses Ansatzes ist folgender: Nicht die Anfragen werden per Broadcast an das ganze Netz gesendet, sondern es werden die eigenen Dienstangebote im gesamten Netz verteilt. Die einzelnen Knoten brauchen dann keine Anfragen mehr senden, sondern können die Dienstfindung lokal durchführen. Zusätzlich zur erzeugten Nachrichtenflut ist hier jedoch auch der benötigte Speicherplatz auf den einzelnen Knoten problematisch.

2.2.2 Ansätze für MANETs

Die Lösungsansätze aus den Bereichen der Netzwerke mit fester Infrastruktur scheinen für den Einsatz in MANETs wenig geeignet.
Entweder wird die ständige Verfügbarkeit einzelner Knoten vorrausgesetzt, oder die benötigten Ressourcen in Bezug auf Kommunikation, Energie und Speicherplatz übersteigen die Fähigkeiten der typischerweise kleinen Geräte im MANET. In [16] werden Service Discovery Ansätze vorgestellt, die für MANETs entwickelt wurden. Sie werden nach zwei Aspekten unterschieden:

1. Vorhandensein einer Verzeichnisstruktur
2. Vorhandensein einer Overlaystruktur

Wenn man vom Vorhandensein einer **Verzeichnisstuktur** spricht, ist damit ein Verzeichnis gemeint, in dem die im Netzwerk verfügbaren Dienste gespeichert sind. Das Verzeichnis wird in der Regel nicht auf einem speziellen Knoten existieren, sondern dezentral sein, d.h. mehrere Knoten fungieren als Verzeichnisknoten und beantworten Dienstanfragen. Die Einteilung der Verzeichnisknoten ist dabei im Gegensatz zu klassischen Ansätzen, aufgrund der Eigenschaften eines MANETs, nicht statisch, sondern dynamisch. Eine Architektur ohne Verzeichnis funktioniert dagegen grundsätzlich nach dem Broadcast-Prinzip, d.h. Angebote und Anfragen werden

an alle Nachbarn gesendet. Im Unterschied zu den klassischen Ansätzen wird im MANET aber versucht das Fluten von Nachrichten zu vermeiden bzw. lokal einzugrenzen.

Unter einer **Overlaystruktur** versteht man eine logische Netzwerkstruktur, die über die physikalische Netzwerkstruktur gelegt wird. Die Overlaystruktur besteht in der Regel aus logischen Verbindungen zwischen einzelnen Knoten. Mit logischer Verbindung ist hier eine, möglicherweise über mehrere Zwischenknoten gehende, Kommunikationsverbindung zwischen Knoten gemeint. Die logische Struktur wird in der Regel gezielt aufgebaut, um *Multicasting* anzuwenden. D.h. Angebote und Anfragen werden an Gruppen von Knoten gesendet, nicht aber an alle Knoten. So wird das Fluten von Nachrichten vermieden und die Netzwerkbelastung reduziert.

Aus diesen zwei Aspekten ergeben sich 4 Kategorien:

- Verzeichnis-Basierte Ansätze mit Overlaystruktur
- Verzeichnis-Basierte Ansätze ohne Overlaystruktur
- Verzeichnislose Ansätze mit Overlaystruktur
- Verzeichnislose Ansätze ohne Overlaystruktur

Nach [16] sind die Verzeichnis-Basierten Ansätze ohne Overlaystruktur sowie die Verzeichnislosen Ansätze mit Overlaystruktur eher untypisch. Bei den Verzeichnis-Basierten Ansätzen ohne Overlaystruktur wird nur *Splendor* [21] als typischer Vertreter genannt und bei den Verzeichnislosen Ansätzen mit Overlaystruktur nur *Allia* [19]. Die verbleibenden zwei Ansätze werden hier anhand typischer Vertreter kurz vorgestellt.

Verzeichnis-Basierte Ansätze mit Overlaystruktur Ein Verzeichnis-Basiertes Protokoll mit Overlaystruktur ist *Service Rings* [13]. Die Knoten im Netzwerk gruppieren sich dabei abhängig von Position und angebotenen Diensten in sogenannten Ringen. Somit ensteht eine Overlaystruktur. In jedem Ring existiert ein sogenannter *Service Access Point* (SAP), der als Verzeichnisknoten für die im Ring angebotenen Dienste fungiert. Die SAPs sind wiederum mit anderen SAPs verbunden und bilden somit eine hierarchische Struktur. Diese Overlaystruktur wird einmalig initialisiert

und muss danach jedoch überwacht und instand gehalten werden. Ähnliche Ansätze verfolgen z.B. *Lanes* [12] und *DSDP* [14].

Verzeichnislose Ansätze ohne Overlaystruktur Ein Vertreter aus der Kategorie der Verzeichnislosen Ansätze ohne Overlaystruktur ist GSD [8]. Dieses dient in dieser Arbeit als Referenzprotokoll und wird daher in Abschnitt 3.1 ausführlich erläutert. Ein weiterer Vertreter für diese Kategorie ist *Konark* [15]. Die Dienstfindung in Konark geschieht verteilt und stützt sich auf den sogenannten *SDP Manager*, der als Dienst auf jedem Knoten läuft. Der SDP Manager speichert lokal verfügbare Dienste in einer sogenannten *Registry* und macht sie anderen Knoten im Netzwerk bekannt. Zudem erlaubt er das Auffinden von Diensten anderer Knoten im Netzwerk.

Nachdem in diesem Kapitel eine Einführung in das Gebiet der Service Discovery gegeben wurde, wird im nächsten Kapitel das GSD Protokoll vorgestellt und die für die Erweiterungen nötigen Anpassungen an der Implementierung des Protokolls erläutert.

3 Anpassung des GSD Protokolls

In dieser Diplomarbeit soll evaluiert werden, wie sich die Positions- und Bewegungsinformationen der Knoten auf Service Discovery in MANETs auswirken. Zu diesem Zweck wird das schon erwähnte GSD Protokoll um einige Komponenten erweitert und anschließend durch Simulationen im NS-2 Simulator die Evaluation durchgeführt. Dazu wird eine vorhandene Implementierung des GSD Protokolls für den NS-2 angepasst und schließlich erweitert. Im folgenden Abschnitt wird zunächst das GSD Protokoll eingeführt und in den zwei darauf folgenden Abschnitten die Anpassungen an der Implementierung des Protokolls und dem Routing des Simulators erläutert. Die einzelnen Erweiterungen für das GSD Protokoll werden dann in Kapitel 4 ausführlich vorgestellt.

3.1 Beschreibung des GSD Protokolls

Das GSD Protokoll ist ein verteiltes Service Discovery Protokoll. Es arbeitet mit einem hierarchischen Verfahren und gruppiert Dienste in bestimmte Kategorien anhand einer vorherbestimmten Ontologie [7]. Kern des Protokolls ist die selektive Weiterleitung von Suchanfragen auf Basis der Gruppeninformationen.

Das Protokoll verwendet kein zentrales Dienstverzeichnis. Vielmehr verfügt jeder Knoten über einen *Cache*, einem lokalen Speicher, in dem sowohl eigene als auch fremde Dienstbeschreibungen gespeichert werden. Ein *Service Provider* (SP) verbreitet deshalb seine Angebote in seiner Umgebung. Knoten, die einen Dienst suchen, wenden sich nicht an einen bestimmten Verzeichnisknoten, sondern an einige bzw. alle Knoten in ihrer Umgebung.

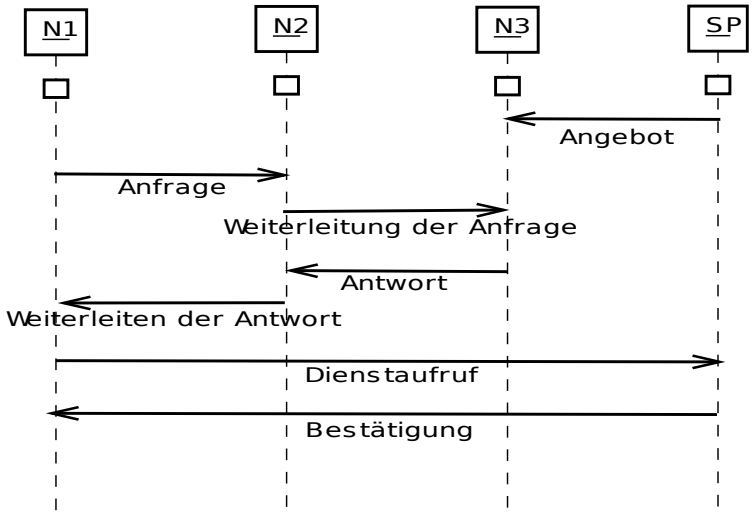

Abbildung 3.1: Nachrichtenübersicht des GSD Protokolls

Dienstangebote Service Provider machen ihre Dienste regelmäßig ihrer Umgebung bekannt. Das Angebot enthält u.a. die Beschreibung des Dienstes mit der Dienstgruppe, die Lebenszeit des Angebots (TTL), die Information, welche Gruppen von Diensten der Anbieter in seiner Umgebung gesehen hat, und wie oft das Angebot weitergeleitet werden darf. Die Angebotsrate drückt aus, wie oft ein Knoten pro Zeiteinheit ein Angebot sendet. Ist die Angebotsrate hoch, ist der Zeitabstand zwischen zwei Angeboten klein. Ist die Rate niedrig, ist der Zeitabstand zwischen zwei Angeboten groß. Der Zeitabstand zwischen zwei Angeboten wird im Folgenden auch als Angebotsabstand bezeichnet.

Cache Knoten speichern empfangene und eigene Angebote im lokalen Cache. Er enthält zu jedem Angebot die Quelladresse, die Dienstbeschreibung mit der Dienstgruppe, die Gruppen von Diensten, die der Anbieter in seiner Umgebung gesehen hat, und die Lebenszeit (TTL) des Angebots. Zudem wird gespeichert, ob der Dienst lokal oder entfernt angeboten wird.

Dienstanfrage Ein suchender Knoten sieht zuerst im lokalen Cache nach, ob dort ein passender Dienst verzeichnet ist und sendet in diesem Fall direkt einen Dienstaufruf an den Service Provider. Andernfalls sucht er im Cache nach Anbietern, die Dienste der gleichen Gruppe in ihrer Umgebung gesehen haben,

und leitet die Anfrage an diese weiter. Falls der Knoten keinen solchen Anbieter findet, sendet er die Anfrage per Broadcast. Beim Weiterleiten der Anfrage wird die Route aufgezeichnet, damit die Antwort, bei einer erfolgreichen Anfrage, auf dem gleichen Weg zum Client zurückgesendet werden kann. Falls die Route nicht mehr existiert, wird das herkömmliche Routing-Protokoll benutzt.

GSD vermeidet also das Fluten der Dienstanfragen, indem es die Semantik der Dienstanfragen beachtet und Anfragen selektiv weiterleitet. Desweiteren kommt das Protokoll ohne ein zentrales Dienstverzeichnis aus und ist deshalb für MANETs gut geeignet.

3.2 Anpassung der vorhandenen GSD Implementierung

Die Evaluation des erweiterten GSD Protokolls wird mittels Simulationen im NS-2 durchgeführt. Dazu wird eine vorhandene Implementierung [2] des Protokolls für den Simulator verwendet. Diese Implementierung befindet sich jedoch in einem recht frühen Reifestadium, so dass wesentliche Teile des Protokolls fehlen und dementsprechend ergänzt werden müssen. Die nötigen Anpassungen und die Vorbereitungen für die Erweiterungen werden in diesem Abschnitt vorgestellt.

3.2.1 Protokollteil

Bei der vorhandenen Implementierung wurde stark auf den Suchprozess an sich fokussiert. Die betreffende Untersuchung hatte zum Ziel den optimalen Abstraktionsgrad für Suchanfragen zu ermitteln. Deshalb wurde von der Implementation nur der Weg einer Suchanfrage bis zu dem Knoten, bei dem ein passender Dienst im Cache verzeichnet ist, abgedeckt. Für den Zweck dieser Arbeit ist aber auch der weitere Protokollablauf wichtig: Die Antwortübermittlung zum Client und der anschließende Dienstaufruf. So wurde die Implementierung um folgende Protokollschritte erweitert: Die Antwortübermittlung und der Dienstaufruf.

Antwortübermittlung

Erhält ein Knoten eine Suchanfrage, prüft er, wie in Listing 3.1, ob er einen passenden Dienst in seinem Cache, entweder lokal oder auf einem Nachbarknoten, verzeichnet hat. Falls ja, generiert er eine Antwort an den Client. Diese enthält, neben der Adresse des Clients, die Adresse des Service Providers und eine Beschreibung des angebotenen Dienstes. Gemäß dem GSD Protkoll wird diese Antwort auf dem gleichen Weg zurück geroutet, auf dem die betreffende Anfrage weitergeleitet wurde, sofern die Route noch intakt ist. Die Vermeidung nicht mehr funktionierender Routen wird folgendermaßen erreicht: Jeder Knoten merkt sich, von welchen Knoten aus ihn eine bestimmte Suchanfrage erreicht hat. Diese Information wird mit einem Zeitstempel versehen und nach Ablauf einer gewissen Frist automatisch gelöscht. Danach wird die Antwort per Unicast direkt an den Client geschickt, ohne die sogenannte *Reverse Route* zu benutzen. Um diesen Protokollteil zu implementieren wurden das GSD Paketformat und die Sende- und Empfangsfunktionen des GSD-Agenten erweitert.

Listing 3.1: Anpassungen in GSD.cc

```
1  void GSD::recv_rqst(Packet* packet) {
2      ...
3      //first we check against the local cache
4      if (cache_.matchService(wanted, (*en)) ) {
5          ...
6          send_reply(packet, en);
7          return;
8      }
9      ...
10 }
```

Dienstaufruf

Mit der Antwort auf eine Suchanfrage erhält der Client die Adresse des Service Providers und kann nun den Dienst aufrufen. Normalerweise gehört dieser Schritt nicht unmittelbar zur Dienstfindung. Jedoch ist es für den Zweck dieser Arbeit interessant, ob ein Dienst auch wirklich erfolgreich genutzt werden kann, oder ob sich

der Service Provider vielleicht schon längst außer Reichweite befindet. Deshalb wird der Dienstaufruf im GSD Protokoll implementiert. Vereinfachend findet hierbei nur ein Pseudoaufruf statt, d.h. der Client sendet den Aufruf und der Service Provider bestätigt den erfolgreichen Aufruf. Dabei überprüft der Service Provider vor der Bestätigung, ob der angefragte Dienst überhaupt noch aktiv ist, ansonsten erfolgt keine Bestätigung. Beide Nachrichten werden per Unicast direkt an den Zielknoten gesendet. Im Gegensatz zu den Antworten, die ohne Hilfe des Routingprotokolls auf der Reverse Route zurückgesendet werden, wird hier also das herkömmliche Routingprotokoll verwendet.

3.2.2 Positions- und Bewegungsinformationen

Im erweiterten GSD Protokoll werden die Positions- und Bewegungsinformationen der einzelnen Knoten in den Protokollablauf einbezogen, um die Effizienz der Service Discovery zu verbessern. Diese Informationen müssen dazu zwischen den einzelnen Knoten ausgetauscht werden. Statt einen Verzeichnisdienst zur Verwaltung der Daten zu verwenden, tauschen die Knoten bei Bedarf ihre Positions- und Bewegungsinformationen aus.

Normalerweise wird ein Positionsbestimmungsdienst, wie etwa das *GPS*-System, gebraucht, damit ein Knoten Informationen über seine Position, Bewegungsrichtung und Geschwindigkeit erhalten kann. Die Positionsbestimmung ist für die Geräte im MANET also immer mit erheblichem Aufwand verbunden. Dieser Aspekt soll jedoch in dieser Arbeit außen vor bleiben. Vielmehr wird vorrausgesetzt, dass die Knoten ihre Position und Bewegung kennen. Deshalb wird für die Simulationen auf eine im NS-2 enthaltene Funktion zurückgegriffen, welche die Positions- und Bewegungsdaten des Knotens zur Verfügung stellt.

Um die Position und Bewegung eines Knoten zu betrachten, sind die folgenden Daten nötig und müssen also zwischen den Knoten ausgetauscht werden:

- X- und Y-Koordinate der aktuellen Position,

- X- und Y-Koordinate der aktuellen Zielposition und die Bewegungsgeschwindigkeit bzw. ein Bewegungsvektor mit Geschwindigkeit in X- und Y-Richtung und
- ein Zeitstempel, der angibt, wann diese Informationen bestimmt wurden.

Während in einem realistischen Szenario die Angabe der Bewegungsrichtung und der Geschwindigkeit in Form eines Bewegungsvektors sinnvoll ist, weil die Knoten sich nicht unbedingt auf eine festgelegte Zielposition zubewegen, wird im NS-2 immer eine konkrete Zielposition und eine Geschwindigkeit zur Bewegungsbeschreibung angegeben. Deshalb werden in der unten dargestellten Datenstruktur *PosInfo*, siehe Listing 3.2, beide Angaben bereitgestellt.

Listing 3.2: PosInfo Datenstruktur in gsd_pkt.h

```
1  struct PosInfo {
2      double pos_x_; //X-Koordinate
3      double pos_y_; //Y-Koordinate
4      double dst_x_; //X-Ziel-Koordinate
5      double dst_y_; //Y-Ziel-Koordinate
6      double d_x_; //Geschwindigkeit in X-Richtung
7      double d_y_; //Geschwindigkeit in Y-Richtung
8      double speed_; //Geschwindigkeit
9      double ts_; //Zeitstempel
10 };
```

Im erweiterten GSD Protokoll existieren verschiedene Fälle, in denen der Austausch dieser Informationen nötig bzw. sinnvoll ist:

Dienstangebot Um die Position und Bewegung eines Service Providers im MANET zu verteilen, müssen diese in allen Angeboten des Knotens enthalten sein. Dann kennt jeder Knoten, der einen Dienst dieses Service Providers benutzen will, dessen Position. Die Knoten, die ein solches Angebot empfangen, speichern bzw. aktualisieren die Informationen zusammen mit den angebotenen Diensten im eigenen Cache. Wenn die Lebenszeit dieses Angebots abgelaufen ist, gehen auch die Positions- und Bewegungsinformationen verloren.

Suchanfrage Bei einer Suchanfrage ist vor allem die Position und Bewegung des Clients sowie der Zwischenknoten, welche die Anfrage weiterleiten, wichtig. So

kennt der Knoten, der einen passenden Dienst im Cache verzeichnet hat und eine Antwort generiert, die Position des Clients und des letzten Zwischenknotens auf der zurückgelegten Route. Eine Anfrage enthält also immer die Positonsdaten des Clients und des letzten weiterleitenden Knotens.

Antwort Die Antwort auf eine erfolgreiche Suchanfrage wird im Normalfall auf derjenigen Route zurückgeroutet, auf welcher die betreffende Anfrage bis zum Suchtreffer weitergeleitet wurde (Reverse Route). Jeder Knoten auf der Route kennt die Adresse und Position des letzten Zwischenknotens und kann so die Antwort weiterleiten. Da es aber sein kann, dass die Gültigkeitsdauer des Reverse Route Eintrags schon abgelaufen ist, muss jede Antwort die Adresse, Positions- und Bewegungsinformation des Clients enthalten.

Dienstaufruf Um den Dienstaufruf zu simulieren, sendet der Client eine Nachricht an den Service Provider, die dieser bestätigen muss. Dazu schickt er die Nachricht mit einer Bestätigung an den Client zurück. Auf dem Weg zum Service Provider enthält das Paket die Positions- und Bewegungsinformationen des Clients, damit der Service Provider weiß, wohin er das Paket schicken muss, und zudem die Positions- und Bewegungsinformationen des Service Providers. Auf dem Weg zurück zum Client enthält das Paket die Positions- und Bewegungsinformationen des Clients und die aktualisierten Informationen über den Service Provider.

In jedem Fall sind mindestens für zwei Knoten Positions- und Bewegungsinformationen enthalten. So bietet es sich an, diese in das allgemeine GSD Paketformat aufzunehmen. In zwei Fällen sind auch noch die Postions- und Bwegungsinformationen eines dritten Knotens wichtig: Bei der Suchanfrage und der Antwort. Deshalb werden sie dem speziellen Paketformat für die beiden Nachrichten hinzugefügt.

3.3 Anpassung des Routings

Das mGSD Protokoll bezieht die Positions- und Bewegungsinformationen der Knoten in die Dienstfindung mit ein. Diese Informationen machen dann auch ein geographisches Routingprotokoll wie das *Gready Perimeter Stateless Routing* (GPSR)[11]

möglich. Wenn die Positionsinformationen der Knoten zwischen den Knoten ausgetauscht werden müssen, liegt es nahe, diese auch für das geographische Routing zu benutzen. Beim geographischen Routing werden Datenpakete anhand ihrer Zielkoordinaten durch das Netz geleitet. Dabei leitet ein Knoten ein Paket immer an denjenigen Knoten aus seiner Nachbarschaft weiter, der den Zielkooridinaten am nächsten liegt. So gelangt das Paket bis zur Zielposition.

Das Protokoll wird zwei Routingverfahren verwenden können: Ein herkömmliches und ein geographisches Routingprotokoll. So ist es möglich für jedes zu sendende Paket festzulegen, ob es mittels des herkömmlichen Routingprotokolls oder mittels geographischen Routings weitergeleitet werden soll. Der NS-2 realisiert das Routing in Netzwerken durch sogenannte Routing-Agenten. Diese übernehmen die Erzeugung und Verwaltung von Netzwerkrouten und ermöglichen so das gezielte Weiterleiten von Paketen durch das Netzwerk. Jedoch kann innerhalb einer Simulation immer nur ein solcher Routing-Agent aktiv sein. Der gleichzeitige Einsatz von zwei Routingverfahren, wie oben beschrieben, ist also erstmal nicht möglich. Deshalb wird im Rahmen dieser Diplomarbeit ein Verfahren entwickelt, das es erlaubt im NS-2 Simulator zwei unterschiedliche Routingverfahren parallel innerhalb einer Simulation zu benutzen. Dieses Verfahren wird im folgenden *Paralleles Routing* genannt.

3.3.1 Verwendete Routingverfahren

Im mGSD Protokoll werden zwei eigenständige Routingverfahren verwendet: Das *Ad-hoc On-Demand Distance Vector* (AODV) Protokoll aus [18] und das GPSR Protokoll. Diese beiden Protokolle werden nun kurz eingeführt.

AODV AODV ist ein reaktives Routingprotokoll für MANETs. Reaktiv bedeutet hierbei, dass Routen innerhalb des Netzwerkes nur bei Bedarf ermittelt werden. Erst wenn eine Route zu einem Zielknoten benötigt wird, beginnt AODV eine Route zu ermitteln. Dazu wird eine Routen-Anfrage, eine sogenannte *RouteRequest*-Nachricht, per Broadcast gesendet. Andere Knoten empfangen diese, speichern den Herkunftsknoten der Anfrage und leiten sie per Broadcast weiter. Durch das Speichern des Herkunftsknotens kennt der Knoten für nachfolgende Routen-Anfragen eine Route

zum Herkunkftsknoten: Er leitet die Nachricht an den Knoten weiter, von dem er die Anfrage bekommen hat. Kennt ein Knoten eine Route zum Ziel, sendet er eine *RouteReply*-Nachricht an den Herkunftsknoten zurück. Dies geschieht auf der gleichen Route, auf der auch die Anfrage weitergeleitet wurde, jedoch in umgekehrter Richtung. Der Herkunftsknoten wählt aus den ermittelten Routen dann diejenige mit der geringsten Hop-Anzahl aus, also die Route mit der geringsten Anzahl von Zwischenknoten. Jede ermittelte Route wird nach einer gewissen Zeit gelöscht, um Routingfehlern, augrund nicht mehr existierender Routen, vorzubeugen. Kommt es doch zu einem Fehler, wird der vorherige Knoten auf der Route informiert und eine neue Route ermittelt.

GPSR GPSR ist ebenfalls ein reaktives Routingverfahren für MANETs. Jedoch im Gegensatz zum AODV Protokoll bezieht GPSR für die Weiterleitung von Paketen die Positionsinformationen der Knoten mit ein. Vorraussetzung hierfür ist aber, dass jeder Knoten seine eigene Position, die seiner Nachbarn und die des Zielknotens kennt. Die Knoten teilen deshalb ihren Nachbarn regelmäßig ihre eigene Position mit, damit diese ihre Nachbarschaftstabellen aktualisieren können. Datenpakete werden immer an denjenigen Nachbarknoten weitergeleitet, der dem Zielknoten am nächsten ist. Dies wird als *Greedy Forwarding* bezeichnet. In Fällen, bei denen ein solcher Weg nicht existiert, setzt der sogenannte *Perimeter*-Modus ein. Wenn ein Knoten keinen Nachbarn hat, der näher am Ziel ist als er selbst, wird das Paket auf den Kanten eines Polygons entgegen dem Uhrzeigersinn solange weitergeleitet, bis ein Knoten einen Nachbarn findet, der näher am Ziel ist als er selbst. Dann wird wieder in den *Greedy*-Modus gewechselt.

3.3.2 Aufbau eines Knotens im NS-2

Bevor die Anpassungen im Routing des NS-2 Simulators beschrieben werden, wird zunächst die allgemeine Funktionsweise eines Knotens im Simulator erklärt. Abbildung 3.2 zeigt den strukturellen Aufbau eines Knotens im Netzwerksimulator.

Sogenannte *Agenten* übernehmen die verschiedenen Aufgaben des Knotens. Diese Agenten sind dabei in verschiedenen Schichten angeordnet, vergleichbar mit dem

ISO-OSI-Schichtenmodell. So übernimmt der GSD-Agent die Ausführung des GSD Protokolls und fungiert dabei als sogenannter *Application*-Agent. Für das Routing ist der Routing-Agent zuständig. In dieser Arbeit ist dies der AODV-Agent. Er

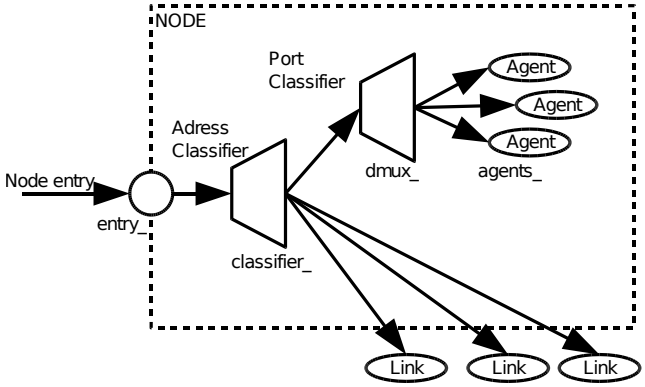

Abbildung 3.2: Aufbau eines Knotens im NS-2 Simulator

ermittelt und verwaltet Routen zwischen den einzelnen Knoten und ermöglicht so das gezielte Senden von Paketen. Wenn ein Application-Agent ein Paket an einen bestimmten Knoten senden will, erzeugt er das Paket, setzt die Zieladresse im *IP Header* und sendet das Paket. Im Simulator wird das Paket nun an den Routing-Agenten übergeben und hier die Route zum Zielknoten ermittelt. Diese Übergabe übernimmt der Port Classifier des Knotens. Der *Port Classifier* nimmt anhand des Zielports eines Pakets die Übergabe an den richtigen Agenten vor.

3.3.3 Paralleles Routing

Wie oben schon erläutert, kann innerhalb einer NS-2 Simulation immer nur ein Routing-Agent verwendet werden. Da in dieser Arbeit jedoch zwei Routingverfahren parallel eingesetzt werden sollen, wird dies mit dem folgenden Verfahren ermöglicht. Ausgangspunkt für die Anpassungen sind die im NS-2 enhaltene Implementierung des AODV Protokolls und eine vorhandene GPSR Implementierung [3].

Die Abbildungen 3.3 und 3.4 zeigen die Struktur des erweiterten Routingverfahrens. Im mGSD Protokoll können Pakete mit AODV oder GPSR weitergeleitet werden. AODV ist dabei das herkömmliche Routingverfahren und nur in bestimmten Fällen

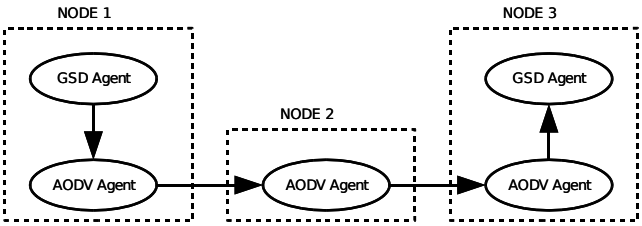

Abbildung 3.3: Herkömmlich geroutetes Paket

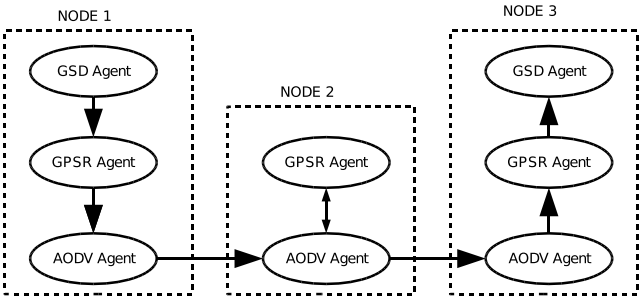

Abbildung 3.4: Geographisch geroutetes Paket

wird GPSR zur Routenwahl eingesetzt. Um den Anpassungsaufwand am NS-2 nicht unnötig zu erhöhen, wurde folgendes Verfahren entwickelt: Der AODV-Agent wird als normaler Routing-Agent innerhalb der Simulation benutzt. An ihm sind keine Änderungen nötig, außer die in Listing 3.3 hinzugefügte Fallunterscheidung. Sie sorgt dafür, dass der AODV-Agent Pakete, die mit GPSR geographisch geroutet werden sollen, gesondert behandelt. Ankommende Pakete werden dem Port Classifier übergeben (Zeile 10), der sie an den GPSR-Agenten weiterleitet. Ausgehende Pakete werden ohne weitere Behandelung sofort gesendet (Zeile 13). Es wird also keine erneute Routenberechnung durchgeführt.

Listing 3.3: Anpassungen in aodv.cc

```
1 void AODV::recv(Packet *p, Handler*) {
2 ...
3 //handle GPSR routed packets separately
4 if(ch->ptype() == PT_GSD) {
5     gsd_pkt* gsd= PKT_GSD(p);
6     if (gsd->rt_type_ == GSD_RT_GPSR) {
7         //just pass packet up or down
```

```
8         //all fields are set
9         if (ch->direction() == hdr_cmn::UP) {
10            dmux_->recv(p,0);
11            return;
12        } else {
13            Scheduler::instance().schedule(target_, p, 0.);
14            return;
15        }
16    }
17 }
18 ...
19 }
```

Der GPSR-Agent wird dem Knoten zusätzlich hinzugefügt, vergleichbar mit einem Application-Agent. Wenn ein Paket mittels geographischem Routing gesendet werden soll, muss der jeweilige Application-Agent, in diesem Fall der GSD-Agent, das Paket an den lokalen GPSR-Agenten übergeben. Dazu sind die Anpassungen in Listing 3.4 nötig. Die ursprüngliche Zieladresse wird im GSD-Paket gespeichert, weil diese sonst verloren geht, und das Paket an den lokalen GPSR-Agenten adressiert. Da das Paket über zwei zusätzliche Stationen läuft, muss zusätzlich der TTL Wert des Paketes um 2 erhöht werden.

Listing 3.4: Anpassungen in GSD.cc

```
1  void GSD::prepareGPSRRoutingPacket(Packet* p) {
2      gsd_pkt* gsd= PKT_GSD(p);
3      hdr_ip* ih = HDR_IP(p);
4
5      gsd->dst_addr().addr_ = ih->dst().addr_;
6      gsd->dst_addr().port_ = ih->dst().port_;
7      gsd->rt_type_ = GSD_RT_GPSR;
8      ih->ttl() += 2;
9      ih->daddr() = this->here_.addr_;
10     ih->dport() = GPSRPORT;
11 }
```

Der GPSR-Agent erhält das Paket und stellt, wie in Listing 3.5, zuerst die ursprüngliche Zieladresse im IP Header wieder her. Danach kann das Paket wie ein normales

GPSR-Paket weiter behandelt werden. Der Agent wählt gemäß dem GPSR Protokoll den nächsten Zwischenknoten für das Paket aus und übergibt das Paket dem AODV-Agenten.

Listing 3.5: Anpassungen in gpsr.cc

```
1  void GPSRAgent::recv(Packet *p, Handler *h) {
2  ...
3  //change ip header dst to original dst
4  if (cmh->ptype_ == PT_GSD ) {
5     gsd_pkt* gsd= PKT_GSD(p);
6     if (gsd->rt_type_ == GSD_RT_GPSR) {
7        iph->dst_.addr_ = gsd->dst_addr().addr_;
8     }
9  }
10 ...
11 }
```

Der Agent wird das GPSR-Paket gesondert behandeln und keine Routenberechnung durchführen, sondern es, wie oben beschrieben, sofort senden, da der nächste Zwischenknoten ja schon festgelegt wurde. Kommt das Paket beim festgelegten Zwischenknoten an, erhält der AODV-Agent das GPSR-Paket und behandelt es wiederum gesondert, indem er es an den GPSR-Agenten übergibt. Wenn das Paket am Ziel ist, wird es dem Application-Agenten, hier dem GSD-Agenten, übergeben. Sonst wählt der GPSR-Agent den nächsten Zwischenknoten aus und sendet das Paket, wie schon beschrieben, weiter.

Um dieses Verhalten zu erreichen, wird also der GPSR-Agent als Application-Agent eingesetzt. Der AODV-Agent dagegen fungiert als normaler Routing-Agent. Die verschiedenen Agenten werden dabei an unterschiedliche Ports gebunden: Der AODV-Agent an den normalen Routingport des NS-2 *255*, der GSD-Agent an Port *113* und der GPSR-Agent an Port *114*. So kann der oben erwähnte Port Classifier die Pakete an die verschiedenen Agenten übergeben.

Abbildung 3.4 zeigt den Ablauf beim geographischen Routing. Zusammengefasst ergeben sich folgende Schritte um ein Paket geographisch routen zu lassen:

1. Der GSD-Agent adressiert das Paket an den lokalen GPSR-Agenten auf Port *114*. Das ursprüngliche Ziel aus dem IP Header wird dabei mit im GSD-Paket gespeichert, da diese Information sonst verloren geht.
2. Der lokale GPSR-Agent empfängt das Paket und stellt zuerst die ursprüngliche Zieladresse im IP Header wieder her. Der Zielport bleibt jedoch gleich, weil das Paket dem entfernten GPSR-Agenten zugestellt werden soll. Dann wählt der GPSR-Agent für das Paket wie üblich den nächsten Hop aus und übergibt das Paket dem AODV-Agenten.
3. Dieser behandelt das GSD Paket gesondert und sendet es ohne Änderungen, besonders nicht am nächsten Hop, direkt weiter.
4. Das GPSR Paket wird dann vom entfernten AODV-Agenten empfangen und ohne weitere Behandlung an den GPSR-Agenten übergeben.
5. Der GPSR-Agent erhält das Paket und sendet es an den lokalen GSD-Agent auf Port *113*.
6. Der GSD-Agent empfängt dann das Paket, welches geographisch geroutet wurde.

Ein Sonderfall sind die sogenannten *Hello*-Nachrichten, die ein GPSR-Agent regelmäßig sendet, damit seine Nachbarn ihre Nachbarschaftstabellen aktualiseren können. Diese müssen wie die normalen Datenpakete den AODV-Agenten passieren. Da diese Nachrichten jedoch alle als Broadcast gesendet werden, sind keine Änderungen, außer dem Anpassen des Zielports im IP Header, nötig. Die Broadcast Pakete werden vom AODV-Agenten einfach weitergeleitet.

Ein zentraler Aspekt des geographischen Routings ist die Möglichkeit, Nachrichten in ein bestimmtes Gebiet zu senden, anstatt an einen bestimmten Knoten. In Bezug auf Service Discovery bedeutet dies, dass man eine Suchanfrage in ein entferntes Gebiet senden kann, ohne den Zielknoten vorher zu kennen. Somit sind entfernte Suchanfragen möglich. Dieser Fall wurde von der vorhandenen GPSR Implementierung nicht abgedeckt. Hier konnte ein Paket immer nur an einen festgelegten Zielknoten gesendet werden. Um die oben beschriebenen entfernten Suchanfragen zu realisieren, wurde die GPSR Implementierung wie folgt erweitert: In der ursprünglichen Implementierung wird ein GPSR Paket solange in Richtung der Zielkoordinaten weiter

geleitet, bis es zum Zielknoten gelangt ist. Im Fall der entfernten Suchanfragen ist dieser Zielknoten aber vorher nicht bekannt. Also würde eine solche Anfrage nie einen GSD-Agenten erreichen, da kein Knoten sich für die Anfrage zuständig sieht. Deshalb wurde die Bedingung für das Ende des Weiterleitungsprozesses geändert: Wenn ein Paket eine Suchanfrage in ein entferntes Gebiet enthält, wird das Weiterleiten des Pakets beendet, falls ein GPSR-Agent das Paket erhält, der innerhalb eines festgelegten Radius um die Zielposition liegt. Dieser Radius wird zusammen mit den Zielkoordinaten im Paket gespeichert. So lassen sich also Suchanfragen innerhalb eines bestimmten Radius um eine Zielposition realisieren. Eine ausführliche Beschreibung der Suchanfragen in ein entferntes Gebiet wird in Abschnitt 4.2.1 gegeben.

Mit dem beschriebenen Verfahren ist nun der parallele Einsatz zweier Routing-Agenten in einer Simulation möglich. Die Verwendung des *Parallelen Routings* im mGSD Protokoll wird in Abschnitt 4.2 beschrieben.

4 Konzeption des mGSD Protokolls

In dem vorhergehenden Kapitel wurden die Anpassungen an der GSD Implementierung und die Konzeption des Parallelen Routings im NS-2 ausführlich erläutert. Im folgenden Kapitel wird nun der Aufbau des mGSD Protokolls beschrieben. Zu Beginn wird der adaptive Konfigurationsmechanismus, der es dem Protokoll erlaubt, sich an die Bewegung der Knoten im MANET anzupassen, vorgestellt. Dann wird das Parallele Routing und dessen Einsatz in Form vom zwei Funktionserweiterungen vorgestellt. Jede der Erweiterungen wird zunächst konzeptionell vorgestellt, um dann die Umsetzung in der Implementierung zu betrachten. Schließlich wird die Evaluationsweise erläutert.

4.1 Adaptive Konfiguration

Die Performanz eines Dienstfindungsprotokolls hängt maßgeblich von seiner Konfiguration ab. Beim GSD Protokoll sind vor allem die Konfiguration des Angebotsabstands und der Lebenszeit der Angebote wichtig. Mit ihnen wird das Protokoll an die äußeren Gegebenheiten angepasst. Ein wichtiger Faktor ist dabei die Bewegung der einzelnen Knoten im MANET. Ob sich die Knoten kaum bewegen oder ihre Position schnell verändern, ist entscheidend für die Konfiguration. Bewegen sich die Knoten sehr langsam, genügt ein größerer Zeitabstand zwischen zwei Angeboten und eine lange Lebenszeit der Angebote. D.h. es werden pro Zeiteinheit nur wenige Angebotsnachrichten gesendet, diese bleiben aber lange gültig. Bewegen sich die Knoten jedoch schnell, wird ein niedrigerer Zeitabstand und eine kürzere Lebenszeit nötig sein. Deshalb muss vor dem Einsatz des Protokolls bekannt sein, wie sich die Knoten bewegen. In einem realistischen Szenario wird man jedoch die Bewegung der Knoten

nur abschätzen können. Zudem werden die Knoten sich mit verschiedenen Geschwindigkeiten bewegen. Das macht eine statische Konfiguration des Protokolls schwierig. Die zwei folgenden Erweiterungen setzen hier an: Sie sollen das mGSD Protokoll automatisch an die jeweilige Bewegung der Knoten im MANET anpassen.

4.1.1 Adaptive Verteilung der Angebote

Die Angebote, die ein Knoten in regelmäßigen Abständen sendet, um seine Dienste in der Umgebung bekannt zu machen, tragen in erheblichem Umfang zur Netzwerkbelastung im MANET bei. Abhängig von der Häufigkeit dieser Angebote ist die Aktualität und Vollständigkeit der auf den Knoten vorhanden Dienstbeschreibungen. Ist der Angebotsabstand, der zeitliche Abstand zwischen zwei Angeboten, zu groß, werden die eigenen Angebote im Netz nur unzureichend verteilt und können nicht oder nur wenig genutzt werden. Wenn der Angebotsabstand zu klein gewählt wird, sind zwar die eigenen Angebote im Netzwerk gut verteilt und können von anderen Knoten genutzt werden, die Belastung des Knotens und des Netzwerkes aber ist höher als notwendig. Gerade bei den meist "schwachen" Geräten im MANET ist dies unerwünscht. Deshalb gilt es einen Kompromiss zwischen Kosten (Netzwerklast) und Nutzen (Angebotsverteilung) zu finden, bei dem der Angebotsabstand so gewählt wird, dass die Angebote eines Knotens ausreichend verteilt sind, die Belastung für Knoten und Netzwerk aber möglichst gering ist. Wenn ein Knoten seine Umgebung kennt, kann er den Angebotsabstand an die Umgebung anpassen.

Bei der adaptiven Verteilungsstrategie aus [5] wird die Verteilung von Datenobjekten in Netzwerken betrachtet. Im Gegensatz zu statischen Verteilungsstrategien wird die Verteilung von Datenobjekten vom Auftreten neuer Objekte in der eigenen Umgebung abhängig gemacht. Im vorgestellten Protokoll wird eine sogenannte *Pull*-Methode verwendet, d. h., ein Knoten muss aktiv nach neuen Datenobjekten fragen. Die adaptive Verteilungsstrategie besagt hierbei: Der Client fragt erst dann nach neuen Objekten, wenn er weiß, dass die Anzahl ihm unbekannter Objekte in seiner Umgebung einen Grenzwert x übersteigt.
Im GSD Protokoll finden solche *Pull*-Anfragen nicht statt, da die Service Provider ihre Dienste selbstständig bekannt machen. Die grundlegende Idee hinter diesem Ansatz kann man jedoch auch in dieser Arbeit benutzen, um den Angebotsabstand

an die eigene Umgebung anzupassen. So kann man die Häufigkeit der Angebote davon abhängig machen, was in der eigenen Umgebung passiert. In einer sich schnell ändernden Umgebung muss ein Knoten seine Dienste häufiger bekannt machen als in einer relativ statischen Umgebung.

Durch die adaptive Verteilung der Angebote soll sich das Verhältnis von Kosten zu Nutzen, also von Netzwerkbelastung zu Verteilung der Angebote, erhöhen, damit die Angebote besser verteilt sind, jedoch die Netzwerkbelastung durch das Senden der Angebote nicht steigt.

Konzeption

Bei der adaptiven Verteilungsstrategie aus [5] wird die Häufigkeit, mit der eigene Dienste im Netz bekannt gemacht werden, an die Umgebung angepasst. Eine solche Anpassung wird auch für die Dienstverteilung im MANET angewendet. Um Änderungen in der eigenen Umgebung zu erkennen, können zwei Informationen genutzt werden. Zum einen kann ein Knoten die Informationen über die eigene Bewegung auswerten: Wenn ein Knoten sich bewegt, ändert sich seine Umgebung und er gelangt in Gebiete, in denen er seine Dienste noch nicht bekannt gemacht hat. Zum anderen kann er aus dem Auftreten neuer Knoten in der eigenen Umgebung darauf schließen, dass diesen seine Dienste noch unbekannt sind. In diesem Fall sendet er dann sofort ein Angebot.
Die Erweiterung *Adaptive Verteilung* setzt sich folglich aus zwei Teilen zusammen:

Adaptive Verteilung Teil I Anpassung der Verteilung der Angebote an die eigene Bewegung des Knotens

Adaptive Verteilung Teil II Anpassung der Verteilung der Angebote an die Anzahl neuer Knoten in der Umgebung des Knotens

Eigene Bewegung Vor dem Senden der eigenen Angebote wird der Knoten den Zeitabstand bis zum nächsten Angebot, abhängig von der eigenen Bewegung, ermitteln. Ein Bestandteil der eigenen Bewegung ist die Bewegungsgeschwindigkeit. Um so schneller der Knoten sich bewegt, desto schneller gelangt er in eine neue

Umgebung. Bewegt er sich jedoch nur langsam, wird seine Umgebung relativ gleich bleiben. Dies wäre also ein erstes Kriterium für die Erkennung einer veränderten Umgebung.

Bei den nachfolgenden Betrachtungen wird idealistischer Weise davon ausgegangen, dass ein Knoten mit seinem Funk eine kreisförmige Fläche mit Radius seiner Funkreichweite r erreichen kann, auch wenn diese Annahme in der Realität aufgrund verschiedener Einflußfaktoren nicht zutrifft. In diesem Sinn hat sich ein Knoten zwischen zwei Angeboten in eine neue Umgebung bewegt, falls er mit seinem Funk zum Zeitpunkt des zweiten Angebots mehr neue Fläche (F_{neu}) erreicht als alte Fläche (F_{alt}). Alte Fläche bezeichnet die Fläche, die er schon beim ersten Angebot erreichen konnte. Abbildung 4.1 zeigt die Änderung der vom Knoten erreichbaren Fläche zwischen zwei Angeboten. Der gestrichelte Kreis stellt die Fläche dar, die der Knoten beim zweiten Angebot erreichen kann.

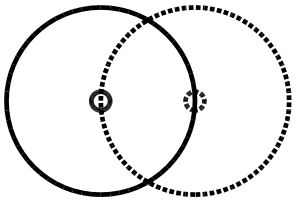

Abbildung 4.1: Veränderung der erreichbaren Fläche nach Bewegung

Um zu berechnen, welche Entfernung ein Knoten zurücklegen muss, um mit seinem Funk mehr neue als alte Fläche zu erreichen, muss man bestimmen, wann die Schnittfläche beider Kreise kleiner als die Differenz der neu erreichten Fläche und der Schnittfläche beider Kreise ist. Dabei gilt für die neu erreichte Fläche F_{neu}, abhängig von der Funkreichweite r und der zurückgelegten Entfernung d:

$$F_{neu} = r^2 * (\pi - 2 * arccos(\frac{d}{2r})) + \frac{d * \sqrt{4 * r^2 - d^2}}{2} \tag{4.1}$$

Für die schon beim letzten Angebot erreichte Fläche F_{alt} gilt:

$$F_{alt} = \pi * r^2 - F_{neu} \tag{4.2}$$

Wenn man beide Funktionen betrachtet, stellt man fest, dass etwa ab einem Verhältnis von

$$d = 0,82 * r \tag{4.3}$$

mehr neue Fläche erreicht wird als alte Fläche. Der Faktor wird als F_{adv} bezeichnet. Anhand dieser Betrachtung kann man dann definieren, wann ein Knoten ein neues Angebot senden sollte. Nämlich dann, wenn er eine Entfernung von

$$d = F_{adv} * r \tag{4.4}$$

zurückgelegt hat.

Wenn man die Bewegungsgeschwindigkeit v des Knotens kennt, kann man berechnen, wann er die oben beschriebene Entfernung zurückgelegt hat und daraus die Zeit t_{adv} bis zum nächsten Angebot bestimmen:

$$t_{adv} = \frac{F_{adv} * r}{v} \tag{4.5}$$

Es ist offensichtlich, dass der Faktor F_{adv} die Häufigkeit der Angebote reguliert. Umso niedriger er ist, desto häufiger werden Angebote gesendet. Die Konfiguration dieses Parameters ist deshalb maßgebend für die Performanz des Protokolls und soll deshalb in der Evaluation untersucht werden.

Problematisch an der Berechnung der Zeit bis zum nächsten Angebot ist jedoch, dass sehr hohe bzw. niedrige Werte auftreten können. So würde bei einer Bewegungsgeschwindigkeit von *0,1m/s* und *30m* Funkreichweite eine Zeitdauer von *300s* bis zum nächsten Angebot berechnet werden. Bei einer Geschwindigkeit von *10m/s* würde hingegen eine Zeitdauer von *3s* berechnet werden. Um solche langen Pausen bzw. das ständige Senden von Angeboten zu verhindern, wird zusätzlich eine obere und untere Grenze für den Abstand zwischen zwei Angeboten eingeführt, so dass die Zeitdauer bis zum nächsten Angebot zum Beispiel im Intervall von *10s* bis *30s* liegt. Die obere Grenze wird mit $t_{adv.max}$ und die untere Grenze mit $t_{adv.min}$ bezeichnet.

Anzahl neuer Knoten Durch die Betrachtung der eigenen Bewegung kann ein Knoten einschätzen, wann er selbst in neue Regionen vordringt und dementsprechend

häufiger oder seltener Angebote senden. Jedoch ist diese Betrachtung noch einseitig. Der Knoten kann nichts darüber aussagen, ob sich seine Umgebung vielleicht deshalb ändert, weil andere Knoten in seine Umgebung vordringen. Wenn ein Knoten erkennt, dass viele neue Knoten in seine Umgebung gelangen, macht er seine Dienste den anderen Knoten sofort bekannt.

Um zu erkennen, ob neue Knoten in die eigene Umgebung gelangen, kann ein Knoten die Angebote, die er empfängt, auf neue Knoten hin auswerten. Denn wenn neue Knoten in die Nachbarschaft gelangen, werden sie ihre Dienste auch bekannt machen. Hierbei ist zu beachten, dass dies nicht unbedingt die eigenen Dienste sein müssen, sondern auch Dienste eines anderen Service Providers sein können. Wenn ein Knoten erkennt, dass zwischen zwei Angeboten in der Nachbarschaft eine bestimmte Anzahl neuer Knoten auftritt, sendet er sofort ein Angebot. Dazu zählt der Knoten wie viele Angebote von neuen Knoten er erhält. Wenn der Knoten selber ein Angebot sendet, wird der Zähler wieder zurückgesetzt. Steigt der Zählerwert über einen festgelegten Grenzwert, sendet der Knoten ein Angebot. Dieser Grenzwert wird als T_N bezeichnet.

Um zu verhindern, dass auf ein solches, nicht reguläres, Angebot sofort das nächste reguläre Angebot folgt, wird die Zeitdauer bis zum Senden des nächsten Angebots, nach dem Senden eines zusätzlichen Angebots, neu berechnet. So wird also das nächste Angebot vorgezogen, und damit der Abstand zwischen zwei Angeboten einmalig verkleinert, wenn eine bestimmte Anzahl neuer Knoten in der Umgebung beobachtet wurde.

Durch diese zwei Erweiterungen passt das mGSD Protokoll den Angebotsabstand an die Umgebungssituation der Knoten an.

Umsetzung

Die Anpassung des Angebotsabstands wird wie folgt umgesetzt: Direkt nach dem Senden eines Angebots, wird anhand der aktuellen Geschwindigkeit gemäß Formel 4.5 die Zeit bis zum nächsten Angebot berechnet. Sollte die Geschwindigkeit $0m/s$ betragen und sich der Knoten überhaupt nicht bewegen, wird der obere Grenzwert $t_{adv.max}$ als Zeitdauer bis zum nächsten Angebot gewählt. Wenn die berechnete Zeit

außerhalb der Grenzen des Intervalls $[t_{adv.min}, t_{adv.max}]$ liegt, wird die nächstliegende Intervallgrenze als Zeitdauer gewählt.

Um die Anzahl neuer Anbieter seit dem letzten Angebot zu zählen, muss jedes Mal, wenn ein Angebot im eigenen Cache gespeichert wird und der Anbieter vorher nicht bekannt war, ein Zähler hochgezählt werden. Die Implementierung des Caches erlaubt nur einen Eintrag pro Knoten. Bevor ein Angebot im Cache gespeichert wird, wird deswegen überprüft, ob schon ein Eintrag für diesen Knoten existiert. Ist dies nicht der Fall, handelt es sich um einen neuen Knoten und der Zähler wird hochgezählt. Ansonsten wird der Eintrag aktualisiert ohne den Zähler hochzuzählen.
Beim Empfang jedes Angebots muss der Knoten den Zählerwert prüfen. Sofern der aktuelle Zählerwert den Grenzwert T_N erreicht bzw. überschreitet, sendet der Knoten ein zusätzliches Angebot, und setzt sowohl den Zähler als auch den *Timer*, der die Zeitdauer bis zum nächsten Angebot bestimmt, zurück. Danach fährt der Knoten mit dem normalen Protokollverlauf fort, d.h. die Zeit bis zum nächsten Angebot wird wieder auf den Standardwert zurückgesetzt bzw. anhand der aktuellen Geschwindigkeit, wie oben beschrieben, neu berechnet.

Evaluationsplanung

Die Anpassung des Zeitabstands zwischen den Angeboten hat zum Ziel eine bessere Verteilung der Angebote zu erreichen, ohne dabei eine gestiegene Netzwerkbelastung inkauf nehmen zu müssen. Während die Beurteilung der Netzwerkbelastung anhand der Anzahl von gesendeten Angeboten erfolgen kann, stellt sich die Frage, wie die Verteilung der Dienstangebote beurteilt werden kann. Im Folgenden wird davon ausgegangen, dass das mGSD Protokoll so konfiguriert wurde, dass Angebote nur an die direkten (*1-Hop*) Nachbarn eines Knotens gesendet werden[1]. Die Verteilung der Angebote eines Knotens N_0 wäre in diesem Fall dann optimal, wenn zu jedem Zeitpunkt t alle direkten Nachbarn $N_1 ... N_c$ die Dienste von N_0 in ihrem Cache haben.

[1] Hierbei ist jedoch nicht die Verteilung der Gruppeninformation im GSD Protokoll gemeint, da diese auch an entferntere Knoten weitergegeben wird.

Den Verteilungsgrad V der Angebote von Knoten N_0 zum Zeitpunkt t kann man dann folgendermaßen definieren:

$$V(N_0, t) = \frac{Anzahl\ der\ Nachbarn,\ die\ Angebote\ von\ N_0\ im\ Cache\ haben}{Anzahl\ der\ Nachbarn} \tag{4.6}$$

Der Verteilungsgrad über alle Knoten, mit n als Knotenanzahl, ist dann:

$$V(t) = \frac{\sum_{i=1}^{n} V(i,t)}{n} \tag{4.7}$$

Für die Evaluation wird dazu periodisch, z.B. alle *5s*, der Verteilungsgrad über alle Knoten bestimmt und daraus ein Mittelwert berechnet. Im Folgenden wird für den Verteilungsgrad auch der Begriff *Right-Positives* in Form einer Prozentzahl verwendet. Umso höher der Right-Positives Wert ist, desto mehr Nachbarn eines Knotens N_0 haben die Angebote von N_0 in ihrem Cache gespeichert.

Wenn man den Right-Positives Wert betrachtet, muss man jedoch auch eine andere Größe betrachten: Wie viele der Cache-Informationen sind falsch bzw. veraltet? Die Definition des *False-Positives* Werts wird in der Evaluationsplanung in Abschnitt 4.1.2 vorgestellt.

Bei der Evaluation der Erweiterung soll natürlich auch betrachtet werden, inwiefern das Ergebnis von einem einzelnen Teil der Erweiterung ausgeht oder von beiden gleichermaßen. Um die Auswirkungen von Teil I der Erweiterung, also der dynamischen Berechnung des Zeitabstands zwischen zwei Angeboten, zu untersuchen, muss man den Grenzwert T_N so hoch setzen, dass er nie erreicht wird. Also zum Beispiel, mit n als Knotenanzahl:

$$T_N = n \tag{4.8}$$

Um Teil II der Erweiterung einzeln zu untersuchen, setzt man die obere und untere Grenze für den Zeitabstand zwischen zwei Angeboten gleich:

$$t_{adv.max} = t_{adv.min} \tag{4.9}$$

In den Simulationen zu Teil I der Erweiterung wird der Wert des F_{adv} Faktors vari-

iert, um zu untersuchen, welcher Wert für den Faktor in Bezug auf das Verhältnis von Netzwerkbelastung zu Verteilung der Angebote angemessen ist. In den Simulationen zu Teil II der Erweiterung wird hingegen der Grenzwert T_N variiert.

4.1.2 Adaptive Lebenszeit der Angebote

Jedes Angebot hat eine begrenzte Lebenszeit. Nach Ablauf dieser Zeit, der *Time To Live* (TTL), muss der Eintrag für diesen Dienst aus den Caches der Knoten entfernt werden, falls kein erneutes Angebot zu diesem Dienst empfangen wird. So wird die Gefahr begrenzt, auf nicht mehr verfügbare Dienste zuzugreifen. Durch diesen Mechanismus wird auch bestimmt, in welchem Umkreis die Dienste eines Knotens bekannt sind. Je höher die Lebenszeit, desto länger verbleibt das Angebot in den Caches der Knoten. Je nachdem wie schnell sich ein Knoten bewegt, kann er sich innerhalb der Gültigkeitsdauer des Angebots erheblich vom Service Provider entfernen. Somit haben Knoten Angebote in ihren Caches, dessen Service Provider sich nicht mehr unbedingt in ihrer Nachbarschaft befinden. Durch diesen Effekt werden mit steigender Lebenszeit mehr Treffer für eine Suchanfrage gefunden[2]. Dies bringt jedoch einen Aktualitätsverlust mit sich, so dass mit steigender Lebenszeit auch die Gefahr steigt, dass die Informationen über die Dienste veraltet sind und die Wahrscheinlichkeit für einen erfolgreichen Dienstaufruf sinkt.

Wenn man die Bewegungsinformation der Knoten kennt, kann man jedoch diesen Nachteil reduzieren und die Lebenszeit an die Bewegung der Knoten anpassen. Ein Service Provider, der sich schnell bewegt, wird für einen Client in der Regel bald nicht mehr erreichbar sein, selbst wenn die Lebenszeit des Angebots noch nicht abgelaufen ist. Die Lebenszeit wäre hier also zu hoch. Der gegenteilige Fall tritt ein, wenn sich ein Service Provider kaum bewegt. Die Lebenszeit könnte eigentlich länger sein, da der Service Provider über eine längere Zeit abgefragt werden kann. Im ersten Fall wird die Erfolgsquote beim Dienstaufruf sinken, weil häufiger nicht erreichbare Dienste aufgerufen werden. Im zweiten Fall wird es weniger Treffer auf eine Suchanfrage geben, da Angebote zu früh aus den Caches der Knoten entfernt

[2] Allerdings wird dieser Effekt nach oben begrenzt, da eine Suchanfrage nur begrenzt weitergeleitet wird.

werden. Eine Anpassung der Lebenszeit an die Bewegung des Service Providers erscheint also durchaus sinnvoll.

Ein zweiter Aspekt bei der Betrachtung der Lebenszeit eines Angebots ist die Bewegung desjenigen Knotens, der das Angebot empfängt. Ein Knoten, der ein Angebot noch in seinem Cache gespeichert hat obwohl er den Service Provider nicht mehr erreichen kann, hat einen unsicheren Eintrag in seinem Cache. Die Lebenszeit sollte also auch an die Bewegung desjenigen Knotens angepasst werden, der das Angebot empfängt. So kann der Knoten anhand der Positions- und Bewegungsinformationen bestimmen, wie lange eine Funkverbindung zum Service Provider noch möglich ist. Falls die ursprüngliche Lebenszeit des Angebots höher ist als die vom Knoten berechnete Zeitdauer, sollte die kleinere Zeitdauer als neue Lebenszeit des Angebots gewählt werden. Im anderen Fall, wenn die berechnete Zeitdauer größer als die ursprüngliche Lebenszeit des Angebots ist, hat es keinen Sinn sich für die höhere Lebenszeit zu entscheiden. Der Service Provider sendet ja nach Ablauf der von ihm berechneten Lebenszeit ein neues Angebot.

Auch wenn diese Erweiterung eng mit der *Adaptiven Verteilung* verknüpft ist, wird sie doch gesondert behandelt, da sie auch in der Evaluation einzeln betrachtet werden soll. Interessant ist aber dennoch, wie sich die beiden Erweiterungen zusammen verhalten. Dazu werden Versuche mit der Kombination beider Erweiterungen durchgeführt.

Konzeption

Bei der Bestimmung der Lebenszeit sollte, wie schon erläutert, die Bewegung der beteiligten Knoten beachtet werden und die Lebenszeit der Angebote entsprechend angepasst werden. Zum einen passt der Service Provider die Lebenszeit seiner Angebote an seine Bewegung an, und zum anderen passen die Knoten, die ein Angebot empfangen, die Lebenszeit an ihre Bewegung relativ zum Service Provider an. Somit gliedert sich diese Erweiterung ebenfalls in zwei Teile:

Adaptive Lebenszeit Teil I Anpassung der Lebenszeit an die Bewegung des Service Providers

Adaptive Lebenszeit Teil II Anpassung der Lebenszeit beim Empfänger an die Bewegung relativ zum Service Provider

Eigene Bewegung des Service Providers Die Lebenszeit wird grundsätzlich, wie der Zeitabstand zwischen zwei Angeboten, anhand der aktuellen Geschwindigkeit eines Knotens bestimmt. Jedoch wird noch der zusätzliche Faktor F_{ttl} eingeführt, um das Verhältnis von Angebotsabstand zu Lebenszeit zu definieren. So wird in [8] ein Angebotsabstand von *15s* und eine Lebenszeit von *40s* benutzt. Der Faktor wäre dann etwa *2,6*. Die Berechnung der Lebenszeit beim Service Provider $t_{ttl.sp}$ wird wie folgt durchgeführt:

$$t_{ttl.sp} = F_{ttl} * t_{adv} \tag{4.10}$$

Wobei bei dynamischer Berechnung des Angebotsabstands gilt:

$$t_{ttl.sp} = F_{ttl} * \frac{F_{adv} * r}{v} \tag{4.11}$$

Wie bei der Anpassung des Angebotsabstands können auch hier sehr hohe bzw. sehr niedrige Werte auftreten. Deshalb wird auch hier eine obere und eine untere Grenze, $t_{ttl.max}$ und $t_{ttl.min}$, für die Lebenszeit beim Service Provider eingeführt und die berechnete Lebenszeit bei Bedarf nach oben oder unten korrigiert.

Wie schon erläutert, reguliert der F_{ttl} Faktor die Anzahl und die Qualität der Treffer für eine gesendete Suchanfrage. Deshalb wird die Konfiguration dieses Parameters in der Evaluation untersucht.

Durch dieses Verfahren wird also die Lebensdauer der Angebote eines Knotens an seine Bewegung angepasst. Die Lebensdauer wird umso kürzer, desto schneller sich ein Knoten bewegt.

Bewegung des Empfängers Wie schon erläutert, passt auch der Empfänger die Lebenszeit des Angebots an seine Bewegung relativ zum Service Provider an. Nämlich dann, wenn er vor Ablauf der Lebenszeit die Funkverbindung zum Service Provider verlieren wird. Um abzuschätzen, wie lange er den Service Provider noch erreichen

kann, hat der Knoten die Positions- und Bewegungsinformationen des Service Providers zur Verfügung. Mit diesen Informationen aus dem Angebot kann er den Abstand zwischen ihm und dem Service Provider als zeitabhängige Funktion berechnen. Der Abstand d zwischen zwei Knoten stellt sich dar als:

$$d = \sqrt{(x_2 - x_1)^2 + (y_y - y_1)^2} \tag{4.12}$$

Wenn sich beide Knoten bewegen kann man den Abstand d als zeitabhängige Funktion darstellen:

$$d(t) = \sqrt{(x_2(t) - x_1(t))^2 + (y_2(t) - y_1(t))^2} \tag{4.13}$$

Dabei gilt für $x(t)$ und $y(t)$ mit dv als Bewegungsvektor:

$$x(t) = x(t = 0) + dv_x * t \tag{4.14}$$

$$y(t) = y(t = 0) + dv_y * t \tag{4.15}$$

Um abzuschätzen, wann zwischen beiden Knoten die Funkverbindung abbricht, berechnet man die Nullstellen der Funktion $f(t)$:

$$f(t) = d(t) - r \tag{4.16}$$

Da $f(t)$ eine quadratische Funktion ist, kann man die Nullstellen mit der *PQ-Formel* berechnen. So kann man berechnen, ob und wann die Funkverbindung zwischen beiden Knoten abbrechen wird. Hat der Knoten berechnet, nach welcher Zeitdauer t_{loss} die Verbindung zum Service Provider abbrechen wird, kann er das Minimum der ursprünglichen Lebenszeit $t_{ttl.sp}$ und der berechneten Zeitdauer als neue Lebenszeit $t_{ttl.client}$ für das Angebot benutzen.
Da beim Service Provider die Lebenszeit noch durch den Faktor F_{ttl} gesteuert wird, muss natürlich auch beim Empfänger des Angebots dieses Verfahren angewendet werden. Also wird die berechnete Zeit bis zum Verbindungsverlust noch mit dem Faktor F_{ttl} multipliziert und erst dann mit der vom Service Provider vorgegebenen Lebenszeit verglichen:

$$t_{ttl.client} = MIN((F_{ttl} * t_{loss}), t_{ttl.sp}) \tag{4.17}$$

Umsetzung

Die Anpassung der Lebenszeit beim Service Provider und beim Empfänger werden direkt vor dem Senden bzw. nach dem Empfangen durchgeführt. Diese Anpassungen sind denen für die Adaptive Verteilung sehr ähnlich.
Anhand der aktuellen Geschwindigkeit wird gemäß Formel 4.11 beim Service Provider die angepasste Lebenszeit berechnet und gegebenenfalls nach oben oder unten korrigiert, falls sie außerhalb des festgelegten Intervalls $[t_{ttl.min}, t_{ttl.max}]$ liegt.
Beim Empfänger des Angebots werden die Positions- und Bewegungsinformationen des Service Providers aus der Angebotsnachricht gelesen und die vorraussichtliche maximale Verbindungsdauer berechnet. Die berechnete Zeitdauer wird mit dem Faktor F_{ttl} multipliziert und mit der ursprünglichen Lebenszeit verglichen. Der kleinere der beiden Werte wird als neue Lebenszeit $t_{ttl.client}$ gewählt und der Eintrag in den Cache geschrieben.

Evaluationsplanung

Durch die Anpassung der Lebenszeit der Angebote soll sich die Aktualität der in den Caches der Knoten gespeicherten Informationen erhöhen. Neben dem Paketverlust soll deshalb auch der Inhalt der Caches untersucht werden. Wie schon bei der Adaptiven Verteilung wird auch hier davon ausgegangen, dass Angebote nur an die *1-Hop* Nachbarschaft gesendet werden.
Ausgehend von dieser Betrachtung kann man folgern, dass die Caches der Knoten aktuell sind, wenn:

1. Jeder der Nachbarn $N_1 ... Nc$ eines Knotens N_0 die Angebote von N_0 im Cache gespeichert hat,
2. und jeder Knoten, der nicht Nachbar von N_0 ist, die Angebote von N_0 nicht im Cache gespeichert hat.

Während der Right-Positives Wert, aus Abschnitt 4.1.1, die erste Bedingung auswertet, sagt der dort erwähnte False-Positives Wert etwas über die Erfüllung der zweiten Bedingung aus.

Bei der Messung der False-Positives in der Evaluation zur Beurteilung der Aktualität, wird jeweils ein Knoten N_0 betrachtet und gezählt, wie viele Knoten Angebote von N_0 im Cache haben, obwohl sie nicht Nachbarn von N_0 sind. Dies wird als Fehlergrad F der Angebote von Knoten N_0 zur Zeit t bezeichnet:

$$F(N_0, t) = \frac{Anzahl\ der\ NichtNachbarn\ von\ N_0\ mit\ dessen\ Angeboten}{Anzahl\ aller\ NichtNachbarn} \tag{4.18}$$

Ähnlich wie beim Verteilungsgrad, kann man auch hier über alle Knoten iterieren:

$$F(t) = \frac{\sum_{i=1}^{n} F(i, t)}{n} \tag{4.19}$$

Wie schon bei der Adaptiven Verteilung wird für die Evaluation F periodisch bestimmt und am Ende der Mittelwert als Ergebnis berechnet. Der Begriff False-Positives ist gleichbedeutend mit dem Fehlergrad F. Wie schon erläutert, ist der False-Positives Wert jedoch, je nach Verhältnis zwischen Angebotsabstand und Lebenszeit, mit einem gewissen Fehler behaftet.

Wenn F kleiner wird, sinkt die Zahl der *false-positive* Fälle, also derjenigen Fälle, in denen ein Knoten Angebote eines Knotens im Cache hat, der nicht sein Nachbar ist. Allerdings muss hier auch der Right-Positives Wert betrachtet werden. Durch die Anpassung der Lebenszeit an die Bewegung von Empfänger und Service Provider sollte sich der Right-Positives Wert jedoch nicht verkleinern. Wäre dies der Fall, würde das bedeuten, dass Angebote zu früh aus den Caches entfernt werden, obwohl noch länger eine Verbindung zum Service Provider möglich ist.

In der Evaluation soll auch untersucht werden, welchen Einfluss die einzelnen Teile der Erweiterung auf das Ergebnis haben. Um Teil I zu untersuchen, wird die Anpassung der Lebenszeit beim Empfänger deaktiviert. Teil II wird einzeln betrachtet, in dem man die obere und untere Grenze für die Lebenszeit gleich setzt:

$$t_{ttl.min} = t_{ttl.max} \tag{4.20}$$

In den Simulationen zu dieser Erweiterung wird der Wert des F_{ttl} Faktors variiert, um zu untersuchen, welche Konfiguration des Faktors im Hinblick auf das Verhältnis

von Trefferanzahl zu Qualität der Treffer die besten Ergebnisse bringt.

4.2 Einsatz des Parallelen Routings

Das mGSD Protokoll wird die Positions- und Bewegungsinformationen der Knoten auch dazu benutzen, geographisches Routing einzusetzen. Dabei soll das geographische Routing aber als Zusatz dienen und das herkömmliche Routing Protokoll in bestimmten Fällen ersetzen. Die Realisierung dieses parallelen Routings wurde in Abschnitt 3.3.3 beschrieben. So kann für jedes zu sendende Paket das Routingverfahren ausgewählt werden. In den folgenden zwei Abschnitten werden nun zwei Einsatzmöglichkeiten für das geographische Routing im mGSD Protokoll vorgestellt. Eine Motivation für den parallelen Einsatz verschiedener Routingverfahren liegt in den grundlegenden Eigenschaften der MANETs selbst. Laut [17] erfordern die verschiedenen Einsatzgebiete für MANETs auch verschiedene Routingverfahren, die an das jeweilige Szenario angepasst sind. So wird je nach Netzwerkgröße, Knotenanzahl und Knotenverteilung ein speziell angepasstes Routingverfahren nötig sein. Dann liegt die Idee nahe, je nach Situation ein passendes Routingverfahren zu benutzen. Wenn man an ein lokal begrenztes MANET denkt, wird diese Idee vielleicht nicht sofort verständlich. Denkt man jedoch an größere Netzwerke, wird klar, dass innerhalb eines solchen Netzwerkes verschiedenste Bedingungen in Bezug auf Knotendichte und -bewegung vorherrschen. Dann scheint es durchaus sinnvoll zwischen Routingverfahren zu wechseln. Da im mGSD Protokoll sowieso schon Positionsdaten zwischen den Knoten ausgetauscht werden, hat es Sinn diese auch zum Routing zu benutzen.
Die Integration beider Routingverfahren in dieser Arbeit ist sicherlich eher rudimentär, so dass die aufgrund des Routings entstehende Netzwerkbelastung sich in etwa verdoppelt. Jedoch könnte man sich vorstellen beide Protokolle so zu verbinden, dass diejenigen Nachrichten, die zur Erkundung der Nachbarschaft periodisch versendet werden, in einer gemeinsamen Nachricht vereint sind. Dann könnte man die Vorteile beider Routingverfahren nutzen, ohne die Netzwerkbelastung dramatisch zu erhöhen. Ein ähnlicher Ansatz, ein Routing Framework, wird in [17] vorgestellt.

4.2.1 Entfernte Dienste

Für einen Knoten im MANET sind in der Regel vor allem die in seiner unmittelbaren Umgebung verfügbaren Dienste interessant. Also zum Beispiel, ob es in der Nähe einen Drucker gibt. Mit den etablierten Service Discovery Protokollen kann man solche Anfragen recht gut beantworten. Jedoch lassen sich Anfragen, die nicht auf die unmittelbare Umgebung abzielen, schwierig oder kaum beantworten. In einem Touristenszenario scheinen solche Anfragen jedoch durchaus sinnvoll. Eine solche

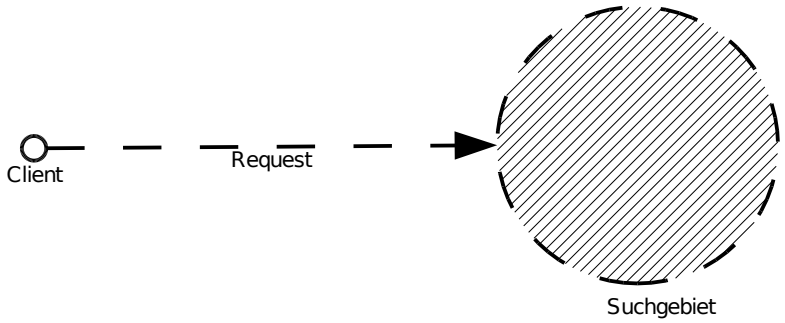

Abbildung 4.2: Entfernte Suchanfrage

Situation ist in Abbildung 4.2 dargestellt. Dienste, die nicht in der unmittelbaren Umgebung erfragt werden, werden im weiteren Verlauf dieser Arbeit als entfernte Dienste behandelt.

Das Konzept der Standortbezogenen Dienste (siehe etwa [20]), auch *Location based Services* (*LBS*) genannt, scheint den entfernten Diensten sehr ähnlich zu sein. Dabei ist es jedoch eher das Ziel der LBS nach Diensten in der eigenen Umgebung zu fragen. Dabei werden anhand der Position eines Knotens für ihn verfügbare bzw. passende Dienste herausgefiltert. Beim Konzept der entfernten Dienste wird aber nach Diensten gefragt, die nicht in der eigenen Umgebung sind, sondern an einer bestimmten Position. Die Dienste sind aber nicht unbedingt auf einen Standort bezogen. So könnte ein Tourist am Alexander Platz in Berlin, nach einem Dienst in der Nähe des Potsdamer Platzes suchen, weil er sich demnächst dorthin bewegt.

Das GSD Protokoll kann diese Art von Anfragen nicht beantworten. Die Problematik liegt dabei darin, dass zwar Dienstanfragen weitergeleitet werden können, aber

nicht nach geographischen Kriterien. Eine Weiterleitung in eine entfernte Suchregion ist somit also nicht möglich. Dazu müsste der suchende Knoten einen Knoten im Zielgebiet kennen und an ihn die Anfrage senden. Dieser Knoten im Zielgebiet ist jedoch nicht bekannt.

Konzeption

Erst mit Einbeziehung geographischer Information, kann eine gezielte Weiterleitung in das Zielgebiet erfolgen. Dies nennt man auch geographisches Routing. Wenn ein Knoten die Position seiner Nachbarn und die Zielposition kennt, kann er die Nachricht an denjenigen Nachbarn weiterleiten, welcher der Zielposition am nächsten ist. So wird die Nachricht solange gezielt in Richtung der Zielposition weitergeleitet, bis sie auf einen Knoten im Zielgebiet trifft. Somit sind Anfragen an einem entfernten Ort möglich.

Wenn ein Knoten eine Suchanfrage in ein entferntes Gebiet senden möchte, muss er die Zielposition und den Zielradius angeben. Das Paket kann dann mittels geographischem Routing in das Zielgebiet weitergeleitet werden. Erreicht das Paket einen Knoten, der sich im Zielgebiet befindet, behandelt dieser die Suchanfrage wie jede andere Suchanfrage. Falls er einen passenden Dienst in seinem Cache hat, sendet er eine Antwort an den Client zurück. Falls nicht, leitet er die Anfrage gemäß dem GSD Protokoll in seiner Umgebung weiter. Trifft die Anfrage dann auf einen Knoten, der einen passenden Dienst im Cache hat, wird die Antwort an den Client zurückgesendet. Die Antwort an den Client wird in beiden Fällen mittels geographischem Routing zurückgesendet. Dazu gibt der Client in der Suchanfrage seine Positions- und Bewegungsdaten an.

Umsetzung

Um Suchanfragen in ein entferntes Gebiet zu unterstützen, wurde das mGSD Paketformat für Suchanfragen, siehe Listing 4.1, um die Felder *target_type*, *target_x*, *target_y*, *target_radius* erweitert:

target_type Das Feld *target_type* beschreibt das Ziel der Suchanfrage. Der Wert *LOCAL* steht für Suchanfragen in der eigenen unmittelbaren Umgebung. Dies ist der Standardfall im GSD Protokoll. *REMOTE* steht für Suchanfragen in ein entferntes Gebiet

target_x Das Feld *target_x* beschreibt die X-Koordinate des Zielgebietes.

target_y Das Feld *target_y* beschreibt die Y-Koordinate des Zielgebietes.

target_radius Das Feld *target_radius* gibt die Größe des Zielgebiets an. Ein mGSD-Agent, der dieses Paket annimmt, muss innerhalb des Radius um die Zielposition liegen.

Listing 4.1: Anpassungen in gsd_pkt.h

```
1 struct rqst_msg {
2         Service wanted_service_;
3         ns_addr_t last_addr_;
4         u_int32_t target_type_;
5         double target_x;
6         double target_y;
7         double target_radius;
8 };
```

Der Ablauf einer Suchanfrage in ein entferntes Gebiet gestaltet sich wie folgt: Der Client setzt die entsprechenenden Felder im mGSD Paket und übergibt es an den GPSR-Agenten. Dieser wählt den nächsten Hop, d.h. den nächsten Zwischenknoten auf dem Weg zum Empfänger, aus und sendet es. Nun wird das Paket in Richtung der Zielposition weitergeleitet. Wenn es auf einen Knoten trifft, der innerhalb des Zielgebiets liegt, ist es am Ziel und wird vom entfernten mGSD-Agenten bearbeitet. Wenn er einen passenden Dienst im Cache verzeichnet hat, generiert er eine Antwort und sendet diese mittels geographischen Routings an den Client zurück. Wenn der entfernte mGSD-Agent keinen passenden Dienst in seiner Umgebung kennt, kann er die Anfrage gemäß des GSD Protokolls weiterleiten. Eine Antwort wird aber auch in diesem Fall mittels geographischen Routings direkt an den Client zurück gesendet.

Falls sich kein Knoten im Zielgebiet befindet, kann dabei ein unerwünschter Nebeneffekt auftreten. Dann kreist die Suchanfrage immer nur um das Zielgebiet herum,

ohne irgendwann von einem Knoten im Zielgebiet bearbeitet zu werden. Durch die Begrenzung der TTL des Pakets werden solche Suchanfragen aber nach einer bestimten Anzahl von Hops verworfen.

Evaluationsplanung

Da Anfragen in entfernte Gebiete im Originalprotokoll nicht möglich sind, ist ein Vergleich beider Protokolle bei der Evaluation nicht möglich. Deshalb wird in der Evaluation dieser Erweiterung untersucht, wie oft solche Anfragen in ein entferntes Gebiet erfolgreich sind, d.h., ob solche entfernten Anfragen praktikabel sind. Bei der Auswertung soll zum einen darauf geachtet werden, für wie viele der Anfragen ein passender Dienst gefunden wird und wie oft die Antwort den Client erreicht. Zum anderen soll untersucht werden, wie oft ein Dienstaufruf nach einer entfernten Suchanfrage erfolgreich ist, und ob der Service Provider wirklich aus dem Suchgebiet stammt oder nicht.

In den Simulationen werden nur solche entfernten Dienste angefragt. Dabei wird die Entfernung zum Zielgebiet variiert. So kann untersucht werden, inwieweit der Erfolg einer entfernten Anfrage von der Entfernung zum Zielgebiet abhängt.

4.2.2 Dynamische Routing Entscheidung

In dieser Arbeit soll der Nutzen der Bewegungs- und Positionsinformationen bei der Service Discovery evaluiert werden. Das Routing ist in MANETs einer der zentralen Faktoren für die Performanz des Netzwerkes. Dabei ist jedoch die Wahl eines geeigneten Routingprotokolls abbängig von den Einsatzbedingungen. So kann es, wie schon erläutert, durchaus Sinn haben zwischen verschiedenen Routingverfahren zu wechseln. Die Entscheidung, welches Routingverfahren für eine bestimmte Situation am besten geeignet ist, kann ein Knoten anhand der Informationen über die Bewegung der Knoten im Netzwerk treffen. Man spricht hier von einer dynamischen Routingentscheidung.

Konzeption

Die Stärke des geographischen Routings liegt, wie schon erläutert, darin, dass keine expliziten Routen berechnet werden müssen. Bei dem in den Simulationen verwendeten herkömmlichen Routingprotokoll AODV ist dies jedoch der Fall. Es liegt darum nahe, sich in den Fällen, in denen AODV zuerst eine Route berechnen müsste, für geographisches Routing zu entscheiden. Zwei Fälle, bei denen eine dynamische Entscheidung über das Routingprotokoll sinnvoll ist, werden hier vorgestellt.

Antwortübermittlung Wird bei einer Suchanfrage ein passender Dienst gefunden, wird eine Antwort erzeugt und an den Client zurück gesendet. Abbildung 4.3 zeigt eine solche Situation. Knoten N_1 erhält die Anfrage vom Client. Diese Anfrage wird nun schrittweise von Knoten zu Knoten weitergeleitet, bis sie Knoten N_4 erreicht. Dieser Knoten hat einen passenden Dienst in seinem Cache, nämlich den Dienst des Service Providers *SP*. Knoten N_4 muss nun die Antwort an den Client zurück schicken. Dafür wird im GSD Protokoll die gleiche Route benutzt wie für das Weiterleiten der Suchanfrage, nämlich die sogenannte Reverse Route. Wenn sich der Client jedoch zwischenzeitlich erheblich von seiner früheren Sendeposition entfernt hat, wird die Reverse Route wahrscheinlich nicht mehr intakt sein und es müsste dann eine neue Route zum Client berechnet werden. In diesem Fall sendet der Knoten N_4, der die Anfrage beantwortet, die Antwort mit geographischem Routing.

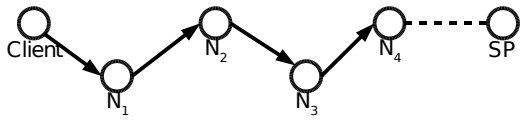

Abbildung 4.3: Ablauf einer Suchanfrage

Dienstaufruf Der Dienstaufruf soll in dieser Arbeit sicherstellen, dass ein Service auch wirklich erfolgreich genutzt werden kann. Die Nachricht wird per Unicast direkt an den Service Provider gesendet. Wenn Service Provider oder Client sich bewegen, scheint es sinnvoll den Dienstaufruf bzw. die Bestätigung geographisch zu routen, da eine alte AODV Route wahrscheinlich nicht mehr funktioniert und das Routing Protokoll zuerst eine neue Route ermitteln muss.

Die Information, ob sich der Service Provider bewegt, ist in der Antwort enthalten, die der Client erhält. In ihr ist die aktuelle Position, die Zielposition und die Geschwindigkeit des Service Provider, sowie ein Zeitstempel der Information, gespeichert. So kann der Client berechnen, wo sich der Service Provider aufhält und den Dienstaufruf geographisch routen lassen. Dies hat, wie schon erwähnt, vor allem dann Sinn, wenn sich der Service Provider bewegt.
Die Bestätigung für den Dienstaufruf sendet der Service Provider ebenso per Unicast direkt an den Client. Auch hier muss man entscheiden, ob herkömmlich oder geographisch geroutet werden soll. Da der Service Provider mit dem Dienstaufruf auch die Positions- und Bewegungsinformationen des Client erhält, kann er berechnen, ob er sich erheblich von seiner früheren Position entfernt hat oder nicht, und dementsprechend das Routingverfahren wählen.

Die Entscheidung ob geographisch geroutet wird, hängt in beiden Fällen, bei Antwort und Aufruf, davon ab, wie weit sich der betreffende Knoten von seiner früheren Position entfernt hat. Als frühere Position gilt dabei die aus der letzten Positionsinformation hervorgehende Position. Der Grenzwert, ab welcher Entfernung geographisch geroutet wird, wird T_{Geo} genannt. Ein plausibel erscheinender Grenzwert bei Funkreichweite r ist:

$$T_{Geo} = \frac{r}{2} \tag{4.21}$$

Dann hat sich der Knoten um eine halbe Funkreichweite von seiner früheren Position entfernt und befindet sich wahrscheinlich in einer neuen Umgebung und eine bestehende Route wird wahrscheinlich nicht mehr funtkionieren.

Evaluationsplanung

Das Ziel dieser Erweiterung ist es geographisches Routing in den Momenten des Protokollablaufs einzusetzen, in denen sich Vorteile gegenüber dem herkömmlichen Routingprotokoll ergeben. Das geographische Routing sollte sich hier vor allem in dem Punkt Antwortzeit vorteilhaft gegenüber dem AODV Protokoll zeigen. In Bezug auf den Paketverlust besteht hingegen beim geographischen Routing, besonders in dünn besetzten Netzwerken die Gefahr, dass mehr Pakete verloren gehen. Deshalb

soll in der Evaluation nicht nur der Einfluss auf die Antwortzeiten untersucht werden, sondern auch der Paketverlust.

Es wurden zwei konkrete Protokollschritte herausgearbeitet, in denen sich geographisches Routing anbietet: Die Antwortübermittlung und der Dienstaufruf. In dem Simulationsszenario sollen dann natürlich gerade solche Fälle erzwungen werden, in denen das mGSD geographisches Routing einsetzt. Hier kann ein Szenario erzeugt werden, in dem sich der Client relativ schnell bewegt. So werden die angesprochenen Fälle gefördert.

5 Evaluation

Nachdem im vorherigen Kapitel die einzelnen Erweiterungen für das GSD Protokoll erläutert wurden, geht es nun um die Evaluation dieser Erweiterungen. Zunächst wird über die Evaluationsweise geschrieben und dann in die Evaluation der einzelnen Erweiterungen und des Parallelen Routings übergegangen.

Um den Nutzen der entworfenen Erweiterungen zu evaluieren, werden sie durch Simulationen im NS-2 Simulator untersucht.

5.1 Simulationsumgebung

Der NS-2 Simulator ist ein diskret ereignisgesteuerter Simulator für Netzwerke, ob kabellos oder verdrahtet. Der Simulator ging *1989* aus dem *REAL* Simulator hervor und wurde im Laufe der Jahre von verschiedensten Institutionen unterstützt. Der oft benutzte Namen *NS-2* meint dabei die aktuelle Version *2* des Simulators. Die *Open Source* Software ist in Forschung und Lehre weit verbreitet und unterstützt die Simulation von *TCP*, *Routing* und *Multicast* Protokollen. Der Simulator ist in *C++* und *OTcl* geschrieben und wird mit *OTcl*-Skripten gesteuert. Seine leichte Erweiterbarkeit und die Fülle an verfügbarer Dokumentation machen ihn zu einem guten Werkzeug für Netzwerksimulationen.

Für die Simulationen wird, soweit nicht explizit anders angegeben, folgendes Szenario verwendet: Auf einem rechteckigen Feld von *200m x 200m* bewegen sich *50* Knoten gemäß dem *Random-Waypoint-Model* [6, 10]. Die Simulationszeit beträgt *600s*. Die Knoten bewegen sich mit normal verteilter Geschwindigkeit zwischen *1m/s* und *3m/s* und *5s* Pausenlänge. Die Funkreichweite der Knoten beträgt *30m*. Auf einem Knoten werden jeweils *10%* der existierenden Dienste angeboten. Diese werden

zufällig verteilt. Im Zeitfenster zwischen *60s* und *590s* Simulationszeit startet alle *5s* ein zufällig ausgewählter Knoten eine Suchanfrage nach einem zufällig gewählten Dienst. Wird ein passender Dienst gefunden, wird er aufgerufen. Somit werden ca. *105* Suchanfragen gestartet. Durch die zufällige Wahl des Dienstes kann es passieren, dass ein lokal angebotener Dienst abgefragt wird. In einem solchen Fall wird die Suchanfrage abgebrochen und demnach in der Auswertung nicht berücksichtigt.

5.2 Messgrößen

Zum Vergleich der verschiedenen Protokolle bzw. Protokollkonfigurationen werden, falls nicht anders angegeben, jeweils drei Aspekte bewertet: Die Verteilung der Dienstangebote, der Nutzen und der Aufwand.

Die *Verteilung* der Dienstangebote wird anhand folgender Messgrößen bewertet:

Trefferanzahl Die Trefferanzahl sagt aus, wie viele Treffer auf eine Suchanfrage während der Simulation erzielt werden. Hierbei wird nicht zwischen verschiedenen Suchanfragen unterschieden, sondern es werden alle Treffer zusammengezählt. Auch mehrere Treffer auf eine Suchanfrage werden dabei mitgezählt.

Lokale Treffer Die Anzahl der lokalen Treffer sagt aus, wie viele Suchanfragen ohne Senden der Suchanfrage direkt lokal beantwortet werden konnten. Je höher dieser Wert ist, umso weniger wird das Netzwerk durch gesendete Suchanfragen belastet.

Right-Positives Die Right-Positives sagen aus, welcher Prozentanteil der Nachbarn eines Knotens seine Dienste in ihrem Cache verzeichnet haben. Je höher dieser Wert ist, desto besser ist die Verteilung. Dieser Wert wird auf allen Knoten periodisch gemessen und über alle Knoten gemittelt.

False-Positives Die False-Positives sagen aus, welcher Prozentanteil der Knoten die Dienste eines Knoten in ihrem Cache verzeichnet haben, obwohl sie nicht seine Nachbarn sind. Je niedriger dieser Wert ist, desto aktueller sind die in den Caches gespeicherten Informationen. Auch hier wird auf allen Knoten periodisch gemessen und über alle Knoten gemittelt.

Um den *Nutzen* der Erweiterung zu bewerten, werden folgende Messgrößen betrachtet:

Dienstaufrufe Die Zahl der gesendeten Dienstaufrufe sagt aus, wie oft in Folge einer Suchanfrage ein passender Dienst gefunden wurde und der Aufruf gestartet wurde.

Erfolgsrate Die Erfolgsrate ist das Verhältnis erfolgreich genutzter Dienste zu der Summe aller Suchanfragen. Hierbei wird also gefragt, wie oft ein gesuchter Dienst auch wirklich erfolgreich genutzt werden kann[1].

Paketverlust Die Paketverlustrate sagt aus, wie viele der gesendeten Nachrichten, wie z.B. ein Dienstaufruf, auf dem Weg zum Empfänger verloren gehen.

Die *Kosten* werden anhand der Netzwerkbelastung bewertet. Dazu werden die gesendeten Nachrichten betrachtet:

Gesendete Nachrichten Die Anzahl gesendeter Nachrichten gibt die Summe aller gesendeten Nachrichten während einer Simulation wieder. Eine Übersicht der im Protokollverlauf gesendeten Nachrichten wird in Abschnitt 3.1 dargestellt.

Um schließlich *Kosten* und *Nutzen* in Relation zu bringen, wird die *Effizienz* bzw. der betriebene *Aufwand* betrachtet:

Effizienz Die Effizienz der Protokolle kann man folgendermaßen definieren:

$$Effizienz = \frac{Erfolgreiche\ Dienstaufrufe}{Gesendete\ Nachrichten} \tag{5.1}$$

Mit dieser Berechnung würde die Effizienz allerdings einen Wert um *0,01* haben. Da solche kleinen Werte für den menschlichen Betrachter jedoch schwer vergleichbar sind, wird stattdessen in der Evaluation der Kehrwert der Effizienz dargestellt. Somit trifft man eine Aussage über die gesendeten Nachrichten

[1] Hierbei werden jedoch solche Suchanfragen nicht beachtet, bei denen ein Knoten einen Dienst sucht, den er selber anbietet. Diese Suchanfragen kommen durch die zufällige Erzeugung der Anfragen zustande.

pro erfolgreichem Dienstaufruf. Dies kann man auch als *Aufwand* pro erfolgreichem Dienstaufruf verstehen:

$$Aufwand = \frac{Gesendete\ Nachrichten}{Erfolgreiche\ Dienstaufrufe} \tag{5.2}$$

Streuung der Messwerte

Die Versuche zu den einzelnen Erweiterungen beinhalten in der Regel *100* bis *150* Simulationen pro Datenpunkt. Die Streuung der Messgrößen ist recht unterschiedlich. Während z.B. die Werte für die Right- und False-Positives eine mittlere Abweichung von nur ca. *2%* des Medians haben, streuen andere Messgrößen recht weit. So weisen die Nachrichtenanzahlen typischerweise eine mittlere Abweichung von ca. *10%* des Medians auf. Dies ist natürlich für die Evaluation ungünstig, ist aber durchaus normal in Versuchen, die von zufällig getroffenen Entscheidungen abhängen. Eine solche Entscheidung ist z.B. die Frage, auf welchen Knoten ein bestimmter Dienst angeboten wird. Um einen Einblick in die Verteilungen der einzelnen Messwerte zu geben, sind im Anhang die sogenannten *Box-Whisker-Plots* (siehe z.B. [1]) der Versuche dargestellt. Diese zeigen für jeden Datenpunkt die Verteilungen der Messwerte. Da die Abbildungen in der Evaluation des Parallelen Routings schon die Verteilungen der einzelnen Messwerte zeigen, werden diese nicht noch einmal im Anhang dargestellt. Die in diesem Kapitel getroffenen Aussagen basieren also nicht nur auf den ermittelten Medianwerten, sondern auch auf den Verteilungen der Messwerte für die einzelnen Datenpunkte.

5.3 Evaluation der Adaptiven Konfiguration

Im folgenden Abschnitt wird der adaptive Konfigurationsmechanismus des mGSD Protokolls untersucht. Nach einer Einzelbetrachtung von Adaptiver Verteilung und Adaptiver Lebenszeit, wird die Kombination beider Erweiterungen untersucht.

5.3.1 Adaptive Verteilung

Die Erweiterung der Adaptiven Verteilung nutzt die Positions- und Bewegungsinformationen der Knoten, um die Verteilung der Angebote im Netzwerk an die Bewegung der Knoten anzupassen. Dabei wird zum einen der Angebotsabstand an die Geschwindigkeit des Service Providers angepasst und zum anderen beim Auftreten einer gewissen Anzahl neuer Knoten in der Umgebung sofort ein Angebot gesendet.

Diese Anpassungen sollen die Verteilung der Dienstangebote im Netzwerk verbessern, ohne dabei eine gestiegene Netzwerkbelastung inkauf nehmen zu müssen[2].
Bei den Versuchen werden drei Aspekte bewertet: Die Verteilung der Angebote, der Nutzen und die Netzwerkbelastung. So soll sich durch die verbesserte Verteilung auch ein Vorteil beim Nutzen, also der Erfolgsrate beim Suchen und Aufrufen eines Dienstes, ergeben.

Adaptive Verteilung I: Anpassung des Angebotsabstands an die Geschwindigkeit

Versuchsaufbau Für diesen Teil der Erweiterung wird der F_{adv} Parameter variiert. Er bestimmt nach welcher zurückgelegten Entfernung erneut ein Angebot gesendet wird. Die Berechnung des Zeitabstands zwischen zwei Angeboten erfolgt dabei gemäß Formel 4.5. Die übrige Konfiguration wird in Tabelle 5.1 aufgelistet. Zum Vergleich wird das GSD Protokoll herangezogen, welches mit derselben Konfiguration gestartet wird, bei dem aber der Angebotsabstand auf den Mittelwert des in den Versuchen mit mGSD beobachteten Angebotsabstands gesetzt wird (siehe Tabelle 5.2). So soll ein möglichst fairer Vergleich ermöglicht werden. Allerdings muss auch deutlich gemacht werden, dass dieser “faire” Vergleich auch derjenige Vergleich ist, bei dem die geringsten Unterschiede zwischen beiden Protokollen erwartet werden.

[2]Die Netzwerkbelastung hängt dabei aber natürlich stark von der Bewegung der Knoten ab. So werden bei schneller Bewegung mehr Angebote gesendet als bei langsamer Bewegung. Die Annahme für diese Versuche ist daher, dass die Geschwindigkeit einer Normalverteilung unterliegt. GSD wird dann auf den Mittelwert der Geschwindigkeitsverteilung eingestellt und mGSD passt sich dynamisch an die aktuelle Bewegungsgeschwindigkeit der Knoten an. So sendet mGSD mal weniger und mal mehr Angebote als GSD. Im Mittel wird aber die Gesamtzahl an Angeboten

Parameter	Wert
$t_{adv.min}$	*10s*
$t_{adv.max}$	*30s*
$t_{ttl.sp}$	*40s*

Tabelle 5.1: Konfigurationsparameter für mGSD, Adaptive Verteilung Teil I

F_{adv}	Angebotsabstand
0,62	*11s*
0,82	*13s*
1,02	*16s*
1,22	*18s*

Tabelle 5.2: Konfigurationsparameter für GSD, Adaptive Verteilung Teil I

Auswertung In Bezug auf die **Verteilung** der Dienstangebote zeigen sich Unterschiede zwischen GSD und mGSD. Abbildung 5.1 zeigt die Suchtreffer. So liegt die Trefferanzahl bei mGSD für hohe Werte des F_{adv} Faktors zwischen *10%* und *15%* über der von GSD. Für niedrige Werte ist die Trefferanzahl jedoch fast gleich. Zudem werden beim mGSD Protokoll zwischen *6%* und *9%* mehr lokale Treffer erzielt, bei denen also keine Suchanfrage gesendet werden musste. Die Right-Positives lie-

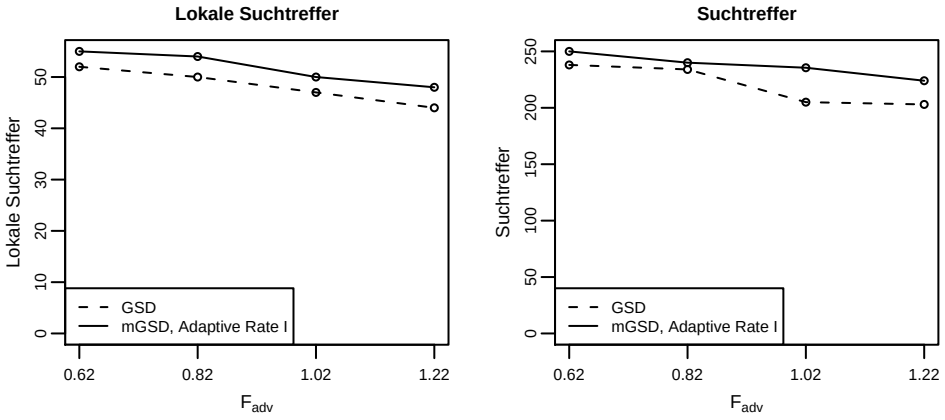

Abbildung 5.1: Adaptive Verteilung I: Treffer auf Suchanfragen

und damit auch die Netzwerkbelastung gleich sein.

gen beim mGSD Protokoll zwischen *7%* und *14%* höher als beim GSD Protokoll. Die Verteilung der Dienstangebote in der eigenen Umgebung ist also beim mGSD im Vergleich zum GSD insgesamt besser. Erstaunlich ist, dass obwohl mGSD etwas weniger Advertisements sendet, die Knoten insgesamt zwischen *11%* und *15%* mehr Angebotes empfangen. Das bedeutet, dass ein Angebot mehr Knoten erreicht als beim GSD Protokoll.

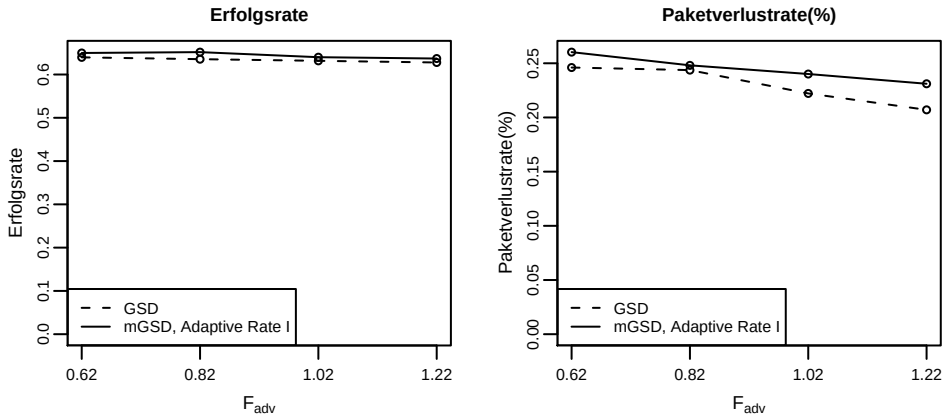

Abbildung 5.2: Adaptive Verteilung I: Erfolgsrate und Paketverlustrate

Beim **Nutzen** liegt mGSD marginal vor GSD. So liegt die Anzahl der Dienstaufrufe und die Erfolgsrate marginal höher, wie Abbildung 5.2 zeigt. Mit sinkendem F_{adv} Faktor bleibt die Erfolgsrate fast gleich und fällt kaum ab.
Das linke Diagramm in Abbildung 5.3 zeigt die **Netzwerkbelastung**. So schneidet GSD für höhere Werte des F_{adv} Faktors besser ab, d.h es sendet insgesamt bis zu ca. *13%* weniger Nachrichten als mGSD. Für niedrige Werte des F_{adv} Faktors senden dagegen beide Protokolle etwa gleichviele Nachrichten. Der Grund für den teilweise gestiegenen Nachrichtenaufwand bei mGSD liegt nicht bei den gesendeten Angeboten, denn mGSD sendet stets etwas weniger Angebote als GSD. Vielmehr ist die verbesserte Verteilung vermutlich der Grund dafür. So steigt die Nachrichtenanzahl genau für die Werte des F_{adv} Faktors, bei denen mGSD mehr Treffer erzielt als GSD. Durch die höhere Trefferanzahl bei mGSD müssen auch mehr Antworten an den Client zurückgesendet werden, so dass sich die gestiegene Netzwerkbelastung erklärt.
Wenn man die **Effizienz** der beiden Protokolle betrachtet, stellt man fest, dass

mGSD für höhere Werte des F_{adv} Faktors eine schlechtere Effizienz aufweist, also pro erfolgreichem Dienstaufruf mehr Nachrichten sendet, wie im rechten Diagramm in Abbildung 5.3 zu sehen ist. Das GSD Protokoll ist hier im Vorteil. Für F_{adv} =*0,62* weisen beide Protokolle etwa die gleiche Effizienz auf, für F_{adv} =*0,82* liegt mGSD marginal vorne (*5%*).

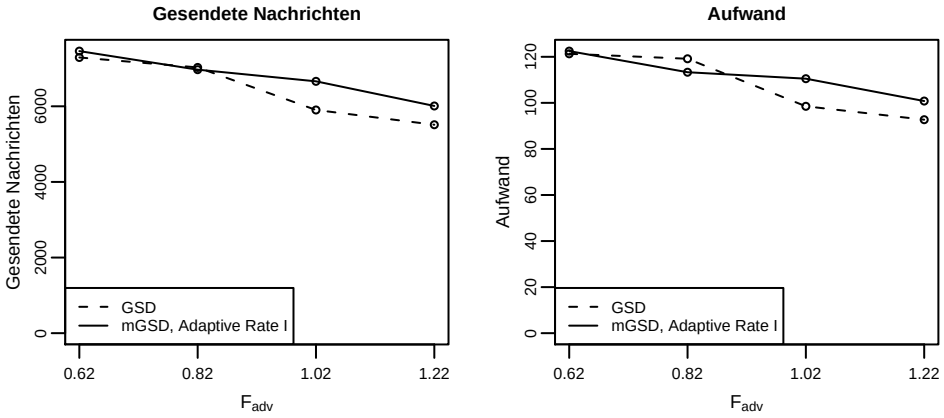

Abbildung 5.3: Adaptive Verteilung I: Netzwerkbelastung und Aufwand

Zusammenfassung Zusammenfassend lässt sich also sagen, dass mGSD, vor allem für hohe Werte des F_{adv} Faktors, eine bessere Verteilung der Dienstangebote im Netzwerk, besonders in der direkten Umgebung eines Knotens, aufweist und damit eine leicht höhere Anzahl von Suchtreffern erzielt. Dies schlägt sich jedoch nicht in einer deutlich gestiegenen Erfolgsrate nieder. Zudem werden bei mGSD für höhere Werte des F_{adv} Faktors mehr Nachrichten gesendet, so dass die Effizienz schlechter ist als bei GSD.

Die Frage, warum sich die bessere Verteilung nicht in einer höheren Erfolgsrate niederschlägt, kann man mit einer schlechteren Aktualität der Dienstangebote beantworten. So werden zwar mehr Treffer erzielt, die jedoch weniger aktuell sind. So zeigt Abbildung 5.2, dass mGSD, besonders für höhere Werte des F_{adv} Faktors, eine höhere Paketverlustrate aufweist (*8% - 12%*).
Welcher Wert für F_{adv} Faktor am besten geeignet ist, lässt sich nicht eindeutig sagen. So ist der Vorsprung von mGSD bei der Anzahl von Suchtreffern für den Wert

F_{adv} =*1,02* am größten. Die Effizienz ist dabei aber aufgrund der geringeren Aktualität etwas schlechter.

Adaptive Verteilung II: Anpassung des Angebotsabstands an neue Knoten

Versuchsaufbau Um Teil II der Erweiterung einzeln zu evaluieren, wird der Angebotsabstand t_{adv} durch Gleichsetzen der oberen und unteren Schranke fixiert und der Grenzwert T_N für das Auftreten neuer Knoten variiert. Zwischen zwei Angeboten wird gezählt, wie viele neue Knoten in der eigenen Umgebung auftreten. Wird dabei der festgelegte Grenzwert T_N erreicht, wird ein zusätzliches Angebot gesendet. Die Konfiguration wird in Tabelle 5.3 dargestellt.

Parameter	**Wert**
$t_{adv.min}$	*15s*
$t_{adv.max}$	*15s*
$t_{ttl.sp}$	*37s*

Tabelle 5.3: Konfigurationsparameter für mGSD, Adaptive Verteilung Teil II

Um die Ergebnisse mit dem GSD Protokoll vergleichen zu können, wird eine zusätzliche Versuchsreihe durchgeführt und dabei der Grenzwert T_N auf *51* gesetzt, so dass keine zusätzlichen Angebote gesendet werden. In den Abbildungen sind diese Werte rechts der Trennlinie dargestellt.

Auswertung Die **Verteilung** der Dienstangebote wird durch die Anpassung des Angebotabstands an neue Knoten leicht verbessert. Wie Abbildung 5.4 zeigt, steigt die Trefferanzahl mit sinkendem T_N Wert. mGSD mit $T_N = 7$ weist eine leicht höhere Trefferanzahl auf, mGSD mit $T_N = 1$ jedoch schon eine um *13%* höhere Trefferanzahl. Ähnlich verhält sich die Anzahl lokaler Treffer: Bei mGSD mit $T_N = 7$ steigt sie wiederum nur leicht, bei mGSD mit $T_N = 1$ jedoch um *17%*. Und auch die Right-Positives verhalten sich ähnlich: Sie steigen um *22%* bei mGSD mit $T_N = 1$ (siehe Abbildung 5.5). Mit sinkendem T_N Wert wird also die Verteilung stetig verbessert. Dies ist verständlich, da die Abbildung 5.4 zeigt, dass die Anzahl gesendeter Angebote um bis zu *212%* (bei mGSD mit $T_N = 1$) steigt.

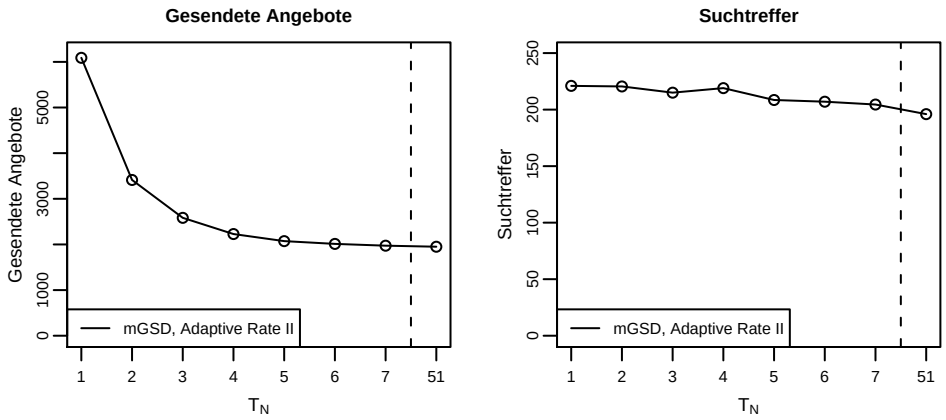

Abbildung 5.4: Adaptive Verteilung II: Gesendete Angebote und Suchtreffer

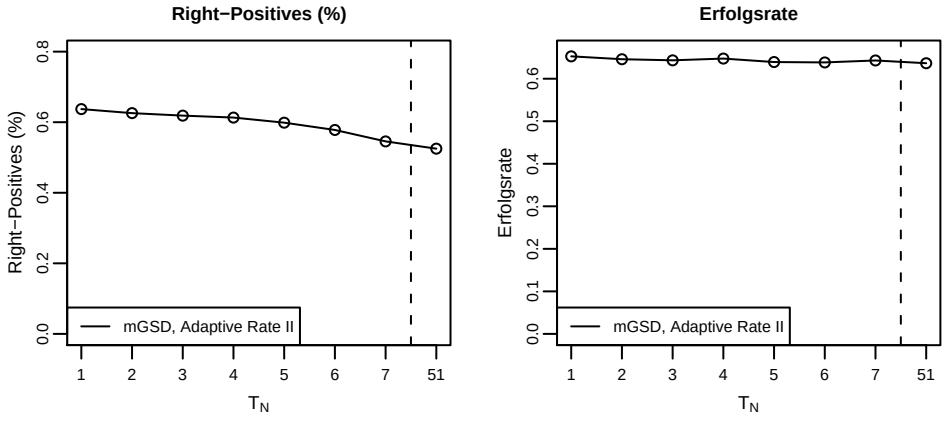

Abbildung 5.5: Adaptive Verteilung II: Right-Positives und Erfolgsrate

Wenn man den **Nutzen** betrachtet, ist eine Verbesserung durch die Erweiterung kaum feststellbar. So steigt die Anzahl der gesendeten Dienstaufrufe nur margial (*4%*) bei mGSD mit $T_N = 1$. Ebenso verhält sich auch die Erfolgsrate, wie Abbildung 5.5 zeigt.

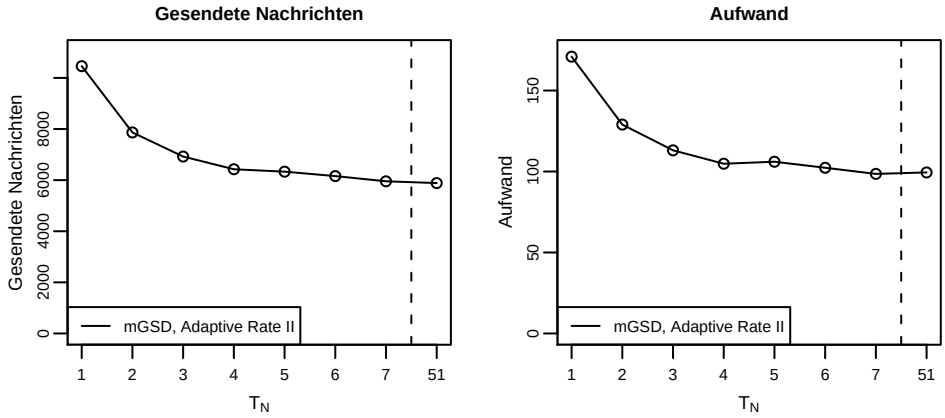

Abbildung 5.6: Adaptive Verteilung II: Gesendete Nachrichten und Aufwand

Hinsichtlich der **Netzwerkbelastung** ist im linken Diagramm in Abbildung 5.6 eine klare Tendenz zu beobachten: mGSD weist wesentlich mehr gesendete Nachrichten auf als GSD. Mit sinkendem T_N Wert werden immer mehr Nachrichten gesendet. So werden bei mGSD mit $T_N = 5$ z.B. etwa *8%* mehr Nachrichten gesendet, bei mGSD mit $T_N = 1$ hingegen ca. *78%* mehr im Vergleich zum GSD.
Das rechte Diagramm in Abbildung 5.6 zeigt, dass die **Effizienz** mit sinkendem T_N Wert deutlich zurück geht. So werden pro erfolgreichen Dienstaufruf bis zu *71%* mehr Nachrichten gesendet. Den starken Anstieg der gesendeten Nachrichten kann man vor allem mit der extrem steigenden Anzahl an Angeboten begründen. Aber auch die steigenden Trefferanzahlen tragen zur erhöhten Netzwerkbelastung bei.

Zusammenfassung Zusammenfassend kann man sagen, dass Teil II der Erweiterung je nach Konfiguration die Verteilung der Dienstangebote verbessert. Jedoch bleibt die Erfolgsrate nahezu gleich bzw. erhöht sich nur marginal. Die bessere Verteilung bringt hier also keinen eindeutig messbaren Vorteil. Zudem zeigt sich der

Nachteil, dass die Netzwerkbelastung je nach Konfiguration extrem steigt. Hier stehen Nutzen und Kosten also in einem extremen Missverhältnis.

Die Frage, warum die verbesserte Verteilung sich nicht auf den Nutzen, sprich die Erfolgsrate, auswirkt, lässt sich nicht leicht beantworten. Auffällig ist, dass die Trefferanzahl keineswegs so stark steigt, wie der Anstieg der gesendeten Angebote vermuten lässt. Dies könnte daran liegen, dass die zusätzlichen Angebote nur Knoten in der eigenen Umgebung erreichen. Wenn diese aber mit Angeboten "gesättigt" sind, wird sich keine Verbesserung der Verteilung mehr einstellen. Gegen diese Hypothese spricht allerdings, dass die Right-Positive keineswegs gegen *100%* gehen, sondern bei etwa *63%* stehenbleiben. Das bedeutet, selbst für $T_N = 1$ haben knapp *40%* der Nachbarn eines Knotens seine Angebote nicht in ihrem Cache.
Die Verbesserung der Verteilung ist im Hinblick auf den Anstieg der Nachrichtenlast vernachlässigbar. Allenfalls hohe Werte für den T_N Parameter scheinen im Hinblick auf die Netzwerkbelastung noch Sinn zu machen. In den Versuchen fällt die Effizienzkurve für Werte $T_N >= 4$ nur leicht ab. Ab $T_N < 4$ fällt die Kurve dann aber stark ab.

Adaptive Verteilung I + II: Kombination beider Teile

Versuchsaufbau Um die Kombination beider Erweiterungsteile zu evaluieren, werden zwei Versuchsreihen durchgeführt und anschließend miteinander verglichen. Zum einen wird nur Teil II der Erweiterung aktiviert und der Grenzwert T_N für das Auftreten neuer Knoten variiert. Zum anderem wird zusätzlich Teil I der Erweiterung aktiviert und die adaptive Berechnung des Angebotsabstands angewendet. So wird deutlich, welchen Einfluss beide Teile der Erweiterungen jeweils haben. Die Konfi-

Konfiguration	t_{adv}	$t_{adv.min}$	$t_{adv.max}$	F_{adv}	$t_{ttl.sp}$
mGSD, Adaptive Verteilung Teil II	*15s*	-	-	-	*40s*
mGSD, Adaptive Verteilung Teil I + II	-	*10s*	*30s*	*1,0*	*40s*

Tabelle 5.4: Konfigurationsparameter für mGSD, Kombination Teil I + II

gurationsparameter werden in Tabelle 5.4 dargestellt. Wie schon bei der Einzelun-

tersuchung von Teil II wird zusätzlich jeweils eine Konfiguration mit $T_N = 51$ zum Vergleich mit GSD untersucht.

Auswertung Man könnte die Ergebnisse der Kombination beider Teile der Erweiterung als Addition der Effekte von Teil I der Erweiterung zu den Effekten von Teil II beschreiben. Da die Ergebnisse grundsätzlich denen vom Teil II der Erweiterung ähneln, werden, ausgehend von den Beobachtungen für Teil II der Erweiterung, nur diejenigen Effekte beschrieben, die sich bei zusätzlicher Aktivierung von Teil I der Erweiterung beobachten lassen.

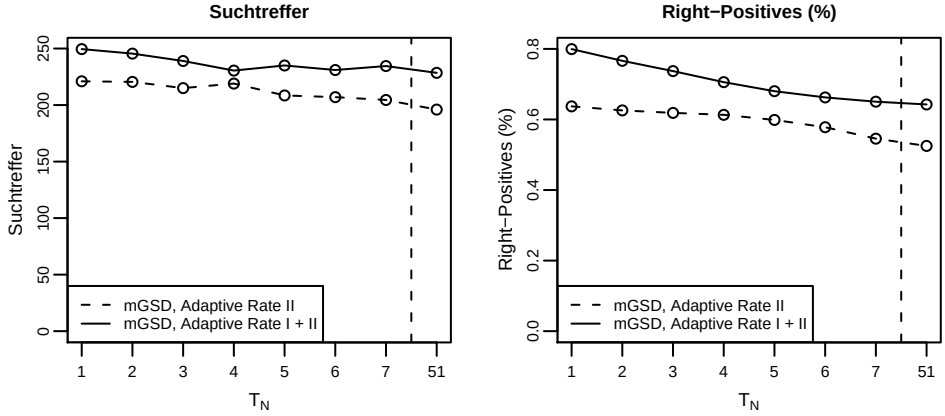

Abbildung 5.7: Adaptive Verteilung I + II: Suchtreffer und Right-Positives

In Bezug auf die **Verteilung** der Dienstangebote zeigt sich in Abbildung 5.7 eine Verbesserung durch die Anpassung des Angebotsabstands an die Geschwindigkeit. Es werden insgesamt mehr Treffer auf die Suchanfragen erzielt, fast unabhängig vom T_N Wert zwischen *11%* und *15%* mehr. Nur bei mGSD mit $T_N = 4$ ist die Erhöhung marginal (*5%*). Auch werden, je nach Konfiguration des T_N Wertes, zwischen *5%* und *13%* mehr lokale Treffer erzielt. Noch mehr verbessern sich die Right-Positives. Diese steigen durch die Adaptive Verteilung zwischen *14%* und *25%*. Die Verteilung der Dienstangebote wird also fast unabhängig vom T_N Wert verbessert.
Der **Nutzen** dieser verbesserten Verteilung ist jedoch wiederum kaum feststellbar: Die Anzahl gesendeter Dienstaufrufe steigt kaum. Ähnlich verhält sich auch die

Erfolgsrate, wie Abbildung 5.8 zeigt. Also wiederum ergibt sich trotz besserer Verteilung kein höherer Nutzen.

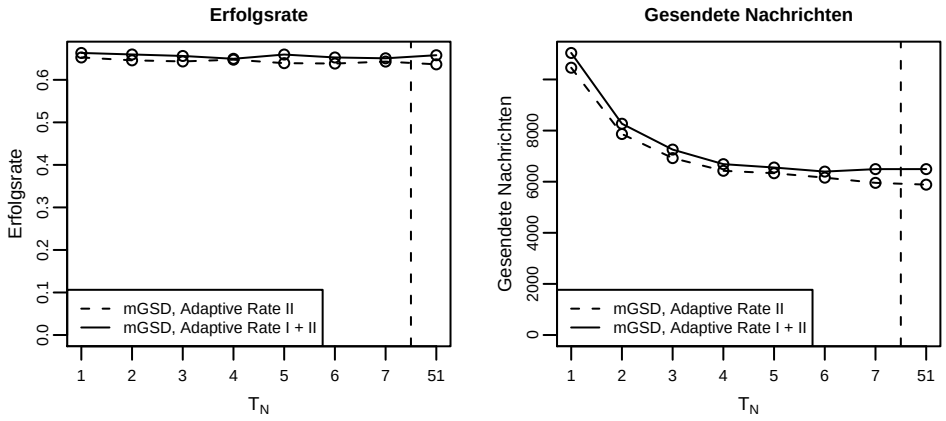

Abbildung 5.8: Adaptive Verteilung I + II: Erfolgsrate und gesendete Nachrichten

Bei der **Netzwerkbelastung** kommt es durch die Aktivierung von Teil I zu einer Erhöhung der Nachrichtenlast um *4%* bis *10%*. Wie das rechte Diagramm in Abbildung 5.8 zeigt, gilt das für alle Werte des T_N Grenzwerts.
Dementsprechend ergibt sich durch die Aktivierung von Teil I eine leichte Verschlechterung der **Effizienz**, d.h. pro erfolgreichem Dienstaufruf werden mehr Nachrichten gesendet. Nur für $T_N = 5$ wird durch die Anpassung des Angebotsabstands an die Geschwindigkeit eine marginale Verbesserung der Effizienz deutlich.

Zusammenfassung Zusammenfassend kann man also sagen, dass sich die Effekte der beiden Teile der Erweiterung gegenseitig nicht beeinflussen, sondern sich vielmehr überlagern. Wie schon in der Untersuchung der einzelnen Teile lässt sich auch hier sagen, dass die Verbesserung der Verteilung der Dienstangebote im Netzwerk keineswegs auch zu einer deutlichen Verbesserung der Erfolgsrate führt. Vielmehr steigt aber die Netzwerkbelastung deutlich. Auffällig ist, dass für mittlere T_N Werte der Vorsprung, in Bezug auf die Verteilung, durch die Anpassung des Angebotsabstands an die Geschwindigkeit kleiner ist als bei den anderen Werten.

Abschließend muss man sagen, dass die Erweiterung der Adaptiven Verteilung für sich allein genommen erstmal wenig Sinn hat, da die Verbesserung in der Verteilung

sich nicht deutlich auf den Nutzen, also die Erfolgsrate, auswirkt. Als Wert für den T_N Grenzwert empfiehlen sich nur die höheren Werte ab *4* oder *5*. In der getesteten Versuchskonfiguration war jedoch schon bei $T_N = 7$ nur ein marginaler Unterschied zur Vergleichskonfiguration mit $T_N = 51$ zu beobachten. Für den F_{adv} Faktor empfiehlt sich aus den Versuchen zu Teil I der Erweiterung der Wert *1,0*. Hier ließen sich die größten Verbesserungen in Bezug auf die Verteilung feststellen.

5.3.2 Adaptive Lebenszeit

Die Erweiterung der Adaptiven Lebenszeit nutzt die Positions- und Bewegungsinformation der Knoten, um die Lebenszeit der Dienstangebote an die Bewegung der beteiligten Knoten anzupassen. Zum einen wird dabei die Lebenszeit beim Service Provider an dessen Geschwindigkeit angepasst, zum anderen passt der Empfänger des Advertisements die Lebenszeit an seine relative Bewegung zum Service Provider an. Durch diese Anpassung soll sich die Aktualität der auf den Knoten gespeicherten Informationen erhöhen. In der Evaluation soll dies nun anhand von Simulation überprüft werden.

Teil I: Anpassung der Lebenszeit beim Service Provider

Versuchsaufbau Um zu untersuchen, wie sich die Anpassung der Lebenszeit an die Geschwindigkeit des Service Providers auswirkt, werden zwei Versuchsreihen miteinander verglichen: Zum einen GSD mit einer statischen Lebenszeit und zum anderen mGSD mit der Adaptiven Lebenszeit auf Seiten des Service Providers. Die Lebenszeit der Angebote wird bei beiden Protokollen anhand Formel 4.10 berechnet. Bei mGSD wird jedoch zusätzlich eine dynamische Berechnung des Angebotsabstands anhand der aktuellen Geschwindigkeit durchgeführt, um die angepasste Lebenszeit berechnen zu können. Der Angebotsabstand bleibt davon aber ansich unberührt. Die Konfigurationsparameter werden in Tabelle 5.5 dargestellt. In den Simulationen wird der F_{ttl} Faktor von *1,5* bis *5* variiert. Somit kann untersucht werden, wie sich die Erweiterung bei unterschiedlichen Lebenszeitwerten verhält.

Parameter	GSD	mGSD
t_{adv}	*15s*	*15s*
$t_{adv.min}$	-	*10s*
$t_{adv.max}$	-	*30s*
F_{adv}	-	*1,0*
$t_{ttl.sp}$	$F_{ttl} * t_{adv}$	-
$t_{ttl.min}$	-	$F_{ttl} * t_{adv.min}$
$t_{ttl.max}$	-	$F_{ttl} * t_{adv.max}$

Tabelle 5.5: Konfigurationsparameter, Adaptive Lebenszeit Teil I

Auswertung Zuerst werden die allgemeinen Ergebnisse beider Protokolle dargestellt und dann die Unterschiede im einzelnen betrachtet.
Generell ist zu beobachten, dass sich bei beiden Protokollen die **Verteilung** der eigenen Angebote mit zunehmender Dauer der Lebenszeit, also steigendem F_{ttl} Faktor, verbessert. Abbildung 5.9 zeigt die steigende Anzahl der Suchtreffer und die ebenfalls steigenden Right-Positives.
In Bezug auf den **Nutzen** lässt sich folgendes sagen: Die Anzahl der Dienstaufrufe steigt zuerst stetig an und scheint dann eine "Sättigung" zu erreichen (siehe linkes Diagramm Abbildung 5.10). Mit der Erfolgsrate verhält es sich jedoch anders. Das rechte Diagramm in Abbildung 5.10 zeigt, dass diese für GSD bis $F_{ttl} = 2$ bzw. für mGSD bis $F_{ttl} = 2,5$ zuerst ansteigt und dann bis knapp unter den Anfangswert abfällt. Beim Paketerlust ist hingegen wieder ein konstanter Anstieg zu beobachten. Abbildung 5.11 zeigt eine stetig steigende **Netzwerkbelastung**, also einen Anstieg der gesendeten Nachrichten mit steigendem F_{ttl} Faktor. Dies führt aufgrund der relativ konstanten Erfolgsrate zu einem Abfall der **Effizienz** mit steigendem Faktorwert: Wie in Abbildung 5.11 zu sehen, werden pro erfolgreichem Dienstaufruf mehr Nachrichten gesendet.
Dass sich durch eine Verlängerung der Lebenszeit die Verteilung der Angebote verbessert, überrascht nicht. Da die Angebote länger in den Caches bleiben, können mehr Treffer erzielt werden und somit auch mehr Dienstaufrufe stattfinden. Durch die Verlängerung der Lebenszeit sinkt aber auch die Aktualität der Caches und der Paketverlust steigt an. So sinkt die Erfolgsrate schließlich sogar.

Nun werden die einzelnen Unterschiede zwischen GSD und mGSD dargestellt.

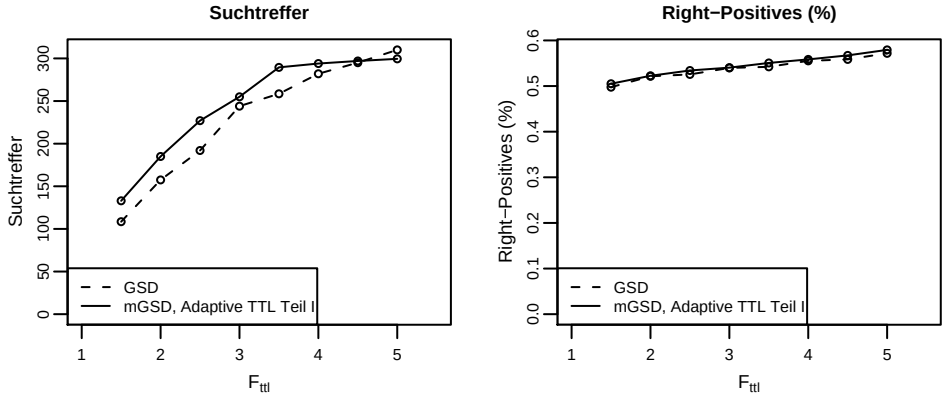

Abbildung 5.9: Adaptive Lebenszeit I: Suchtreffer und Right-Positives

Die **Verteilung** der Angebote im Netzwerk wird durch die Anpassung der Lebenszeit im Vergleich zum GSD verbessert. So steigt die Zahl lokaler Treffer zwischen *3%* und *10%* an, die Anzahl gesendeter Suchanfragen verringert sich um *3%* bis *14%*. Abbildung 5.9 zeigt, dass die Anzahl der Treffer bei mGSD bis zu *22%* höher ist als bei GSD. Jedoch fällt mGSD ab einem Wert von $F_{ttl} = 4{,}5$ leicht hinter GSD zurück. Die Werte für die Right-Positives sind dagegen etwa gleich.

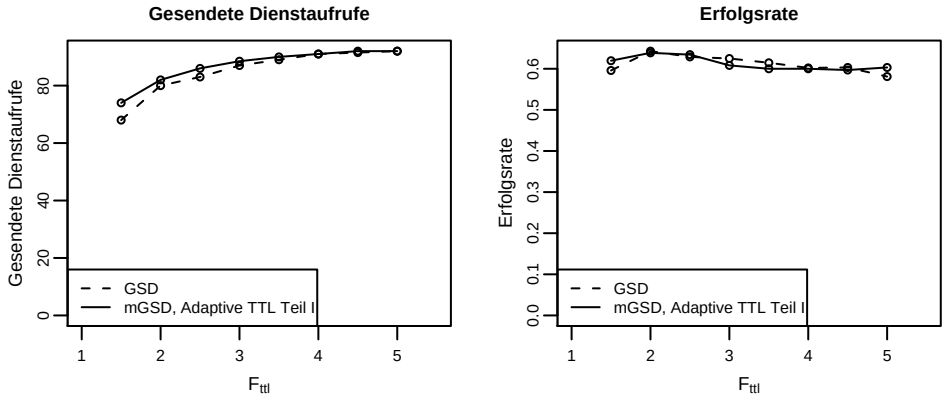

Abbildung 5.10: Adaptive Lebenszeit I: Gesendete Dienstaufrufe und Erfolgsrate

Beim **Nutzen** sind die Verhältnisse nicht so klar. Bei der Anzahl der Dienstaufrufe liegt mGSD zuerst mit bis zu *9%* vorne. Abbildung 5.10 zeigt, dass sich GSD und

mGSD jedoch immer mehr annähern. Bei der Erfolgsrate liegt mGSD meist marginal hinter GSD. Beim Paketverlust schneidet GSD besser ab. Es gehen bis zu *19%* weniger Pakete verloren als bei mGSD.
Auch hinsichtlich der **Netzwerkbelastung** kann mGSD nicht überzeugen: Es werden bis zu *10%* mehr Nachrichten gesendet. Jedoch gleichen sich GSD und mGSD für hohe Werte des F_{ttl} Faktors einander an, wie in Abbildung 5.11 zu sehen.
Bei der **Effizienz** liegt mGSD folglich etwas hinter GSD. Pro erfolgreichen Dienstaufruf werden bis zu *10%* mehr Nachrichten gesendet. Nur für Werte ab $F_{ttl} = 4{,}5$ liegt mGSD leicht vor GSD.

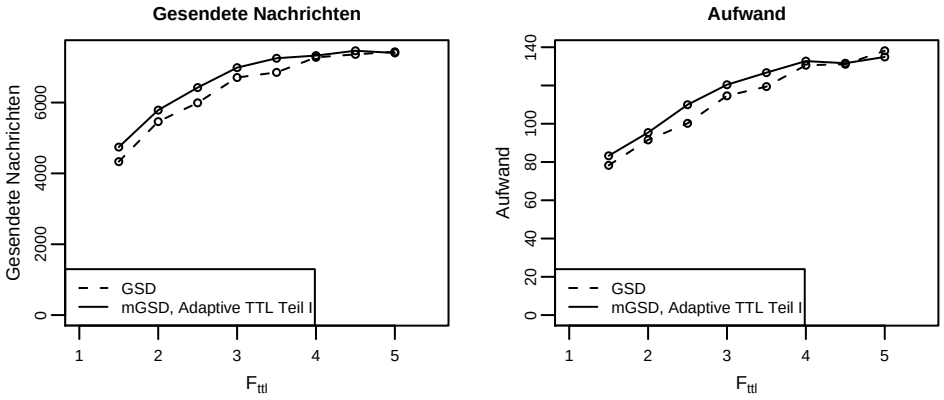

Abbildung 5.11: Adaptive Lebenszeit I: Gesendete Nachrichten und Aufwand

Zusammenfassung Zusammenfassend lässt sich also sagen, dass die Anpassung der Lebenszeit beim Service Provider zwar die Verteilung der Dienste verbessert, dies jedoch nicht zu einer höheren Erfolgsrate führt. Der Grund für die leicht verbesserte Verteilung liegt vermutlich an der im Mittel leicht höheren Lebensdauer der Dienstangebote bei mGSD. Hinsichtlich der Effizienz hat sich ein Wert von *1,5* für den F_{ttl} Faktor als am besten geeignet erwiesen.

Teil II: Anpassung der Lebenszeit beim Empfänger

Versuchsaufbau Für die Evaluation von Teil II der Erweiterung wird, wie schon zuvor, das GSD Protokoll mit statischer Lebenszeit mit dem mGSD Protokoll mit adaptiver Lebenszeit verglichen. Jedoch wird bei Teil II die Lebenszeit nur auf Seiten des Empfängers eines Angebots angepasst. Die Lebenszeit der Angebote beim Service Provider wird anhand Formel 4.10 berechnet. Beim mGSD Protokoll wird

Parameter	GSD	mGSD
t_{adv}	$15s$	$15s$
$t_{ttl.sp}$	$F_{ttl} * t_{adv}$	$F_{ttl} * t_{adv}$

Tabelle 5.6: Konfigurationsparameter, Adaptive Lebenszeit Teil II

zusätzlich beim Empfänger die Lebenszeit anhand der relativen Bewegung zum Service Provider angepasst. Dazu wird Formel 4.17 benutzt. Die Konfigurationsparameter werden in Tabelle 5.6 dargestellt. In den Versuchen wird der F_{ttl} Faktor angepasst, um zu untersuchen, wie sich die Erweiterung für verschiedene Lebenszeitwerte verhält.

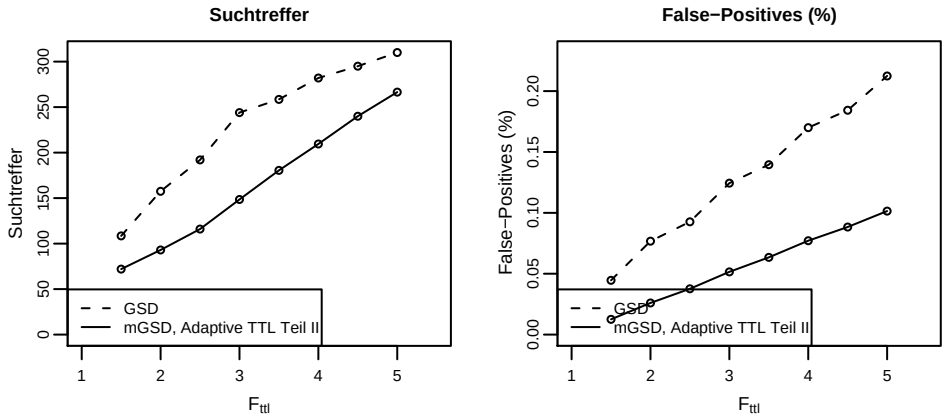

Abbildung 5.12: Adaptive Lebenszeit II: Suchtreffer und False-Positives

Auswertung Die Anpassung der Lebenszeit beim Empfänger ergibt deutliche Unterschiede in der **Verteilung** der Angebote im Netzwerk. So sinkt die Anzahl lokaler

Treffer zwischen *29%* und *35%* im Vergleich zu GSD. Abbildung 5.12 zeigt, dass zugleich auch die Zahl der nicht lokalen Treffer erheblich sinkt. Es werden zwischen *14%* und *41%* weniger Treffer auf Suchanfragen erzielt. Die Werte für die Right-Positives sinken dagegen durch die Anpassung der Lebenszeit nur etwas (*7%*). Eine drastische Veränderung ist aber bei den False-Positives zu beobachten (siehe rechtes Diagramm, Abbildung 5.12). Diese sinken durch die Anpassung der Lebenszeit beim Empfänger zwischen *52%* und *72%* im Vergleich zu GSD. Das bedeutet, dass die Caches der Knoten deutlich weniger Angebote von Knoten enthalten, die nicht in ihrer direkten Nachbarschaft sind. Dies ist eine eindeutige Verbesserung in Bezug auf die Qualität der Suchtreffer.

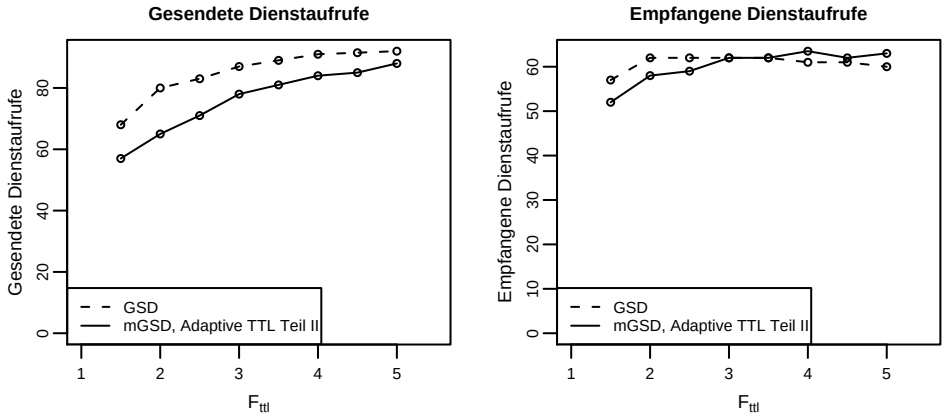

Abbildung 5.13: Adaptive Lebenszeit II: Gesendete und empfangene Dienstaufrufe

In Bezug auf den **Nutzen** lässt sich folgendes beobachten: Die Zahl gesendeter Dienstaufrufe liegt bei mGSD jeweils unter der von GSD. So werden bis zu *19%* weniger Aufrufe gesendet. Abbildung 5.13 zeigt, dass der Unterschied mit zunehmendem F_{ttl} Faktor abnimmt und bei $F_{ttl} = 5$ nur noch marginal ist (*4%*). Das rechte Diagramm in Abbildung 5.13 zeigt, dass bis $F_{ttl} = 3$ bei GSD noch bis zu *9%* mehr gesendete Aufrufe von den Service Providern empfangen werden. Darüber kommen bei mGSD etwas mehr Dienstaufrufe bei den Service Providern an. Allerdings ist der Vorsprung mit *2%* bis *5%* nur gering. Abbildung 5.14 zeigt die Erfolgsrate und den Paketverlust. So liegt bei der Erfolgsrate GSD zuerst bis zu *10%* vorne, ab $F_{ttl} = 3$ hat mGSD einen geringen Vorsprung von bis zu *7%*. Hingegen liegt der Paketverlust beim mGSD für alle Werte des F_{ttl} Faktors sehr deutlich, zwischen *26%* und *52%*,

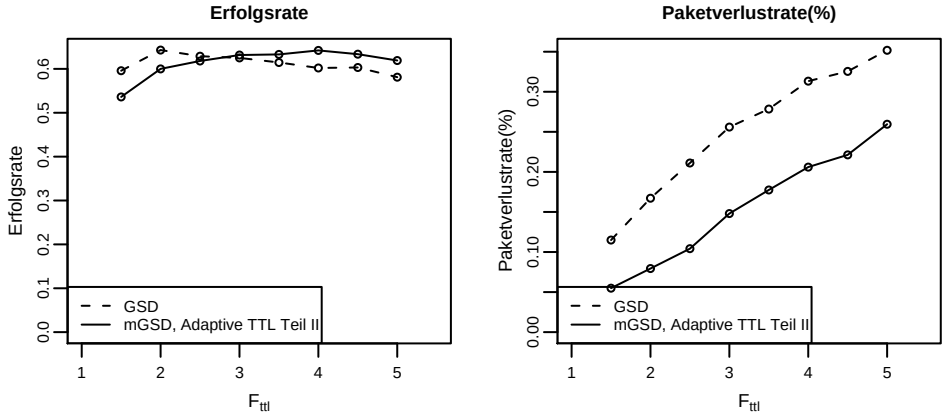

Abbildung 5.14: Adaptive Lebenszeit II: Erfolgsrate und Paketverlustrate

unter dem von GSD. Obwohl weniger Dienstaufrufe gesendet werde, ist der Nutzen zumindestens ab $F_{ttl} = 3$ leicht größer. Dies liegt an dem durch die Anpassung der Lebenszeit beim Empfänger deutlich reduzierten Paketverlust.

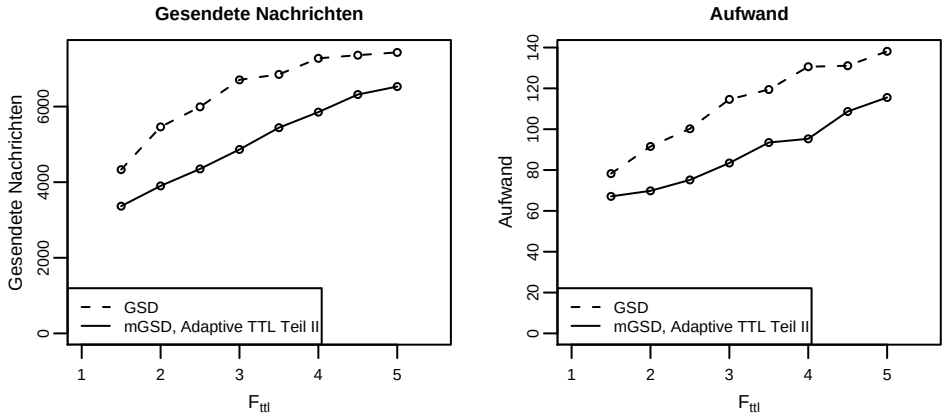

Abbildung 5.15: Adaptive Lebenszeit II: Gesendete Nachrichten und Aufwand

Die **Netzwerkbelastung** ist bei mGSD deutlich niedriger im Vergleich zu GSD. So zeigt Abbildung 5.15, dass bis zu *29%* weniger Nachrichten gesendet werden. Allerdings sinkt der Vorsprung vom mGSD mit steigendem F_{ttl} auf schließlich *12%* für $f_{ttl} = 5$. Somit liegt mGSD aber auch in Bezug auf die **Effizienz** deutlich vor GSD. Das rechte Diagramm in Abbildung 5.15 zeigt: Es werden pro erfolgreichem

Dienstaufruf bis zu *27%* weniger Nachrichten gesendet. Für Werte des F_{ttl} Faktors zwischen *2* und *4* liegt der Vorsprung immer deutlich über *20%*. Für die anderen Werte beträgt er immerhin noch etwa *15%*. Der Grund für den deutlichen Rückgang der Nachrichtenlast liegt nicht an weniger gesendeten Angeboten. Vielmehr sind andere Nachrichten der Grund dafür. Durch die verringerte Anzahl an Suchtreffern, bis zu *41%*, müssen auch weniger Antworten an die Clients zurückgesendet werden. Zudem werden auch etwas weniger Dienstaufrufe gesendet. so kommt der Rückgang der Netzwerkbelastung auch dadurch zustande, indem Nachrichten eingespart werden.

Zusammenfassung Die Anpassung der Lebenszeit beim Empfänger bringt deutliche Vorteile in der Effizienz des Protokolls. Mit rund *20%* weniger Nachrichten wird die gleiche bzw. eine etwas höhere Erfolgsrate erreicht. Da insgesamt bei mGSD deutlich weniger Suchtreffer erzielt werden, kann man folgern, dass die gefundenen Treffer jedoch, im Vergleich zu GSD, wesentlich besser sind. Die Erweiterung erhöht demzufolge die Aktualität der Caches, vermindert veraltete Treffer und damit unnötige Dienstaufrufe.
Ein Frage ergibt sich aus dem Absinken der Right-Positives. Nach der Konzeption der Erweiterung kommt dieses daher, dass einige Knoten die Dienstangebote zu früh aus ihrem Cache entfernen. Dieser Effekt kann darauf beruhen, dass die Knoten von Zeit zu Zeit ihre Geschwindigkeit und ihre Richtung ändern. In einem solchen Moment ist die Berechnung der Zeit bis zum Verbindungsverlust zwischen zwei Knoten ungenau.
Für den F_{ttl} Faktor empfehlen sich, aufgrund der Effizienz, Werte bis *3*.

Adaptive Lebenszeit I + II: Kombination beider Teile

Versuchsaufbau Um die Kombination beider Teile dieser Erweiterung zu evaluieren, werden drei Versuchsreihen gegenübergestellt: Die Kombination beider Teile, Teil II und das ursprüngliche GSD Protokoll. Dabei werde ich mich vor allem auf die Unterschiede der Kombinationsversuche zu den Versuchen mit Teil II der Adaptiven Lebenszeit konzentrieren. Der sonstige Versuchsaufbau wurde aus der Evaluation

von Teil I übernommen, nur dass zusätzlich die Anpassung der Lebenszeit beim Empfänger durchgeführt wird.

Auswertung Die **Verteilung** der Dienstangebote im Netzwerk verändert sich bei der Kombination beider Erweiterungsteile kaum. Abbildung 5.16 zeigt die Treffer und die lokalen Treffer. Diese, wie auch die Right-Positives und False-Positives, bleiben fast unverändert. Die verbesserte Verteilung, die in Teil I zu beobachten war,

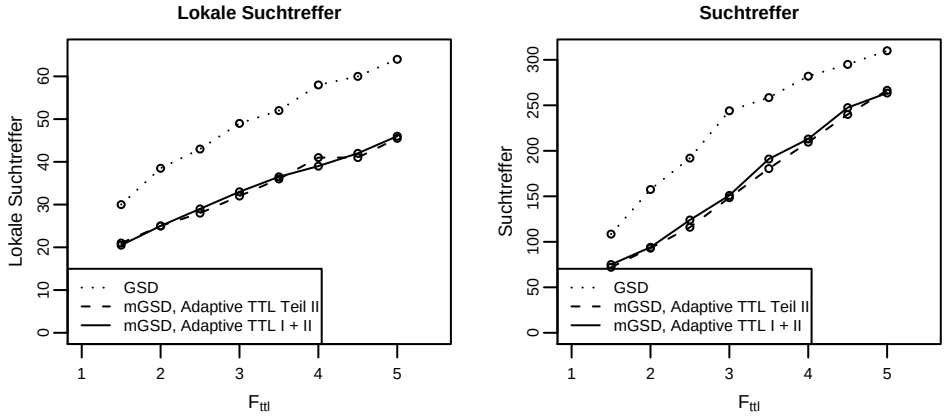

Abbildung 5.16: Adaptive Lebenszeit I + II: Lokale Suchtreffer und (nicht lokale) Suchtreffer

wird hier also nicht sichtbar. Teil II der Erweiterung scheint hier klar zu dominieren und zusätzliche Angebote in den Caches zu verhindern.
Beim **Nutzen** lassen sich ebenfalls nur geringe Veränderung durch die Kombination beider Teile feststellen. So werden im Vergleich zu Teil II der Erweiterung marginal mehr Dienstaufrufe gesendet, aber etwas weniger empfangen. Die Erfolgsrate verhält sich ebenso und liegt geringfügig unter der von Teil II. Beim Paketverlust sind in Abbildung 5.17 jedoch Unterschiede festzustellen. So steigt dieser durch die Kombination beider Teile um bis zu *10%* an.
Bezüglich der **Netzwerkbelastung** ist keine große Veränderung feststellbar. Durch die Kombination beider Erweiterungsanteile steigt die Anzahl der gesendeten Nachrichten im Vergleich zu Teil I sehr geringfügig. Dies kann man mit den teilweise etwas höheren Trefferzahlen begründen.
Somit sinkt die **Effizienz** im Vergleich zu Teil I bis auf einige Werte des F_{ttl} Faktors

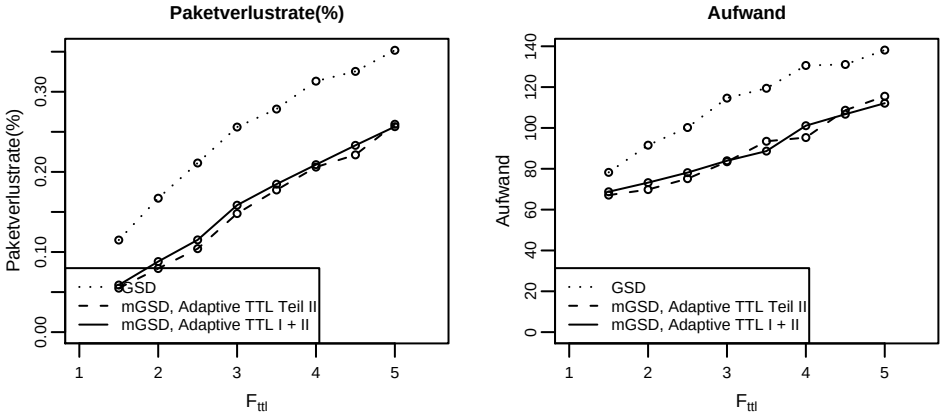

Abbildung 5.17: Adaptive Lebenszeit I + II: Paketverlustrate und Aufwand

marginal. Es werden pro erfolgreichem Dienstaufruf, wie im rechten Diagramm in Abbildung 5.16 zu sehen, also etwas mehr Nachrichten gesendet.

Zusammenfassung Zusammenfassend kann man sagen, dass Teil II der Erweiterung klar dominiert und Teil I kaum Einfluss auf die Performanz des Protokolls hat. Dies ist verständlich, wenn man bedenkt, dass die Anpassung der Lebenszeit beim Empfänger die relative Bewegung des Empfängers zum Service Provider betrachtet, der Service Provider hingegen nur seine eigene Bewegung betrachtet und die Anpassung somit deutlich ungenauer ist. Die in Teil II schon festgestellte Effizienzverbesserung gegenüber GSD bleibt also bestehen.

5.3.3 Kombination beider Erweiterungen

Nachdem beide Erweiterungen unabhängig voneinander betrachtet wurden, wird nun die Kombination beider Erweiterungen untersucht. Dazu werden zwei Versuchsreihen durchgeführt. Beim ersten Versuch wird der Versuchsaufbau nahezu vollständig von den vorhergehenden Versuchen übernommen. Beim zweiten Versuch wird die Geschwindigkeit der Knoten deutlich erhöht, ohne die Konfiguration der Protokolle anzupassen. Dieser Versuch soll zeigen, welche Auswirkungen die Adaptive Konfiguration bei sich extrem ändernden Bedingungen hat.

Adaptive Verteilung und Adaptive Lebenszeit, Versuch 1

Versuchsaufbau Um die Kombination beider Erweiterungen zu evaluieren, wird mGSD mit GSD und der bis jetzt effiziensten mGSD-Konfiguration verglichen. Hierbei handelt es sich um mGSD mit Adaptiver Lebenszeit Teil II, welches nun als mGSD-TC bezeichnet wrd. So lässt sich feststellen, welche mGSD-Konfiguration am besten abschneidet. Die Konfigurationsparameter werden in Tabelle 5.7 dargestellt.

Parameter	GSD	mGSD	mGSD-TC
t_{adv}	*15s*	-	*15s*
$t_{adv.min}$	-	*10s*	-
$t_{adv.max}$	-	*30s*	-
F_{adv}	-	*1,0*	-
T_N	-	*5*	-

Tabelle 5.7: Konfigurationsparameter für den 1. Kombinationsversuch

Die Lebenszeit der Angebote beim Service Provider wird bei statischem Angebotsabstand nach Formel 4.10 und bei dynamischer Berechnung des Angebotsabstands nach Formel 4.11 berechnet. Bei mGSD und mGSD-TC wird zusätzlich die Lebenszeit beim Empfänger nach Formel 4.17 angepasst. In den Versuchen wird der F_{ttl} Faktor zwischen *1,5* und *5* variiert.

Auswertung Bei der **Verteilung** der Dienstangebote zeigen sich deutliche Unterschiede zwischen den drei Versuchsreihen. Abbildung 5.18 zeigt die Treffer und die lokalen Treffer. Hier weisen beide, mGSD und mGSD-TC, deutlich niedrigere Werte auf als GSD. mGSD ist mGSD-TC jedoch etwas überlegen. Bei den Right-Positives schneidet mGSD-TC zwischen *9%* und *12%* schlechter ab als GSD, mGSD jedoch deutlich besser als GSD, nämlich zwischen *13%* und *23%*. Bei den False-Positives wiederum liegen beide mGSD Konfigurationen deutlich vor GSD, wenn auch mGSD-TC nochmal *11%* bis *22%* besser abschneidet als mGSD.
Hinsichtlich des **Nutzens** schneidet mGSD deutlich besser ab als mGSD-TC. So werden bei mGSD zwar zwischen *3%* und *10%* weniger Dienstaufrufe gesendet als bei GSD, jedoch im Vergleich zu mGSD-TC zwischen *4%* und *14%* mehr. Während mGSD-TC bei den empfangenen Aufrufen erst ab $F_{ttl} = 3,5$ leicht vor GSD liegt,

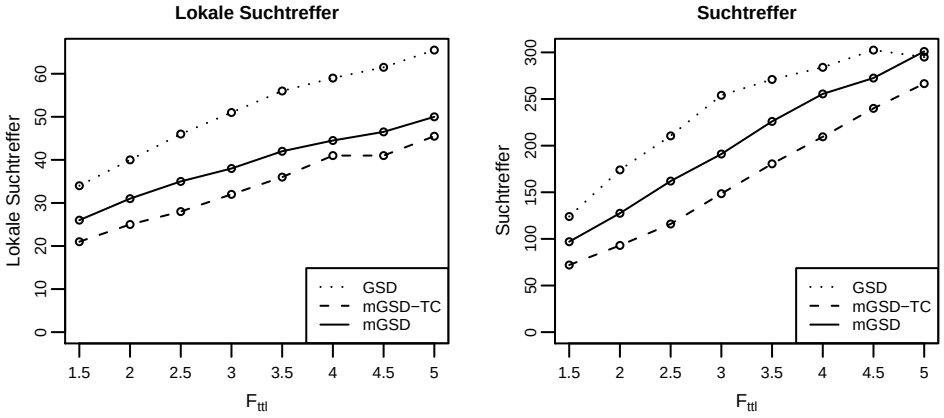

Abbildung 5.18: 1. Kombinationsversuch: Lokale und (nicht lokale) Suchtreffer

liegt mGSD schon ab $F_{ttl} = 2,5$ leicht vor GSD. Der Abstand zwischen mGSD und mGSD-TC liegt für $F_{ttl} = 1,5$ noch bei *10%*, verkleinert sich mit steigendem F_{ttl} Faktor aber solange, bis beide Protokolle für $F_{ttl} = 5$ gleichauf liegen. mGSD erreicht also wesentlich früher bessere Werte für die empfangenen Aufrufe im Vergleich zum GSD als mGSD-TC. Das linke Diagramm in Abbildung 5.19 zeigt die Erfolgsrate. Ab $F_{ttl} = 2$ liegt mGSD bis zu *9%* vor GSD. mGSD-TC liegt erst ab $F_{ttl} = 3,5$ bis zu *6%* vor GSD. Der Vorsprung von mGSD zu mGSD-TC sinkt auch hier von anfangs *11%* auf nahezu Null. Im rechten Diagramm in Abbildung 5.19 kann man sehen, dass bei der Paketverlustrate beide mGSD Konfigurationen deutlich vor GSD liegen. Von anfangs *46%* bei mGSD und *57%* bei mGSD-TC sinkt der Vorsprung beim Paketverlust mit steigendem F_{ttl} Faktor auf *23%* bei mGSD und *25%* bei mGSD-TC.

Die **Netzwerkbelastung** der drei Protokolle ist im linken Diagramm in Abbildung 5.20 dargestellt. Zu sehen ist, dass mGSD-TC für alle Werte des F_{ttl} Faktors klar vor den anderen Protokollen liegt. Aber auch mGSD liegt zumindestens für Werte bis $F_{ttl} = 4$ klar vor GSD. mGSD-TC sendet zwischen *9%* und *30%* weniger Nachrichten als GSD und *9%* bis *20%* weniger als mGSD. Jedoch werden die Unterschiede zwischen den drei Protokollen mit steigendem F_{ttl} Faktor geringer.

Wenn man nun die **Effizienz** betrachtet, lässt sich feststellen, dass mGSD-TC, wie in Abbildung 5.20 zu sehen, die wenigsten Nachrichten pro erfolgreichem Dienstaufruf sendet. mGSD erreicht hier zwar schlechtere Werte, liegt aber immer noch vor

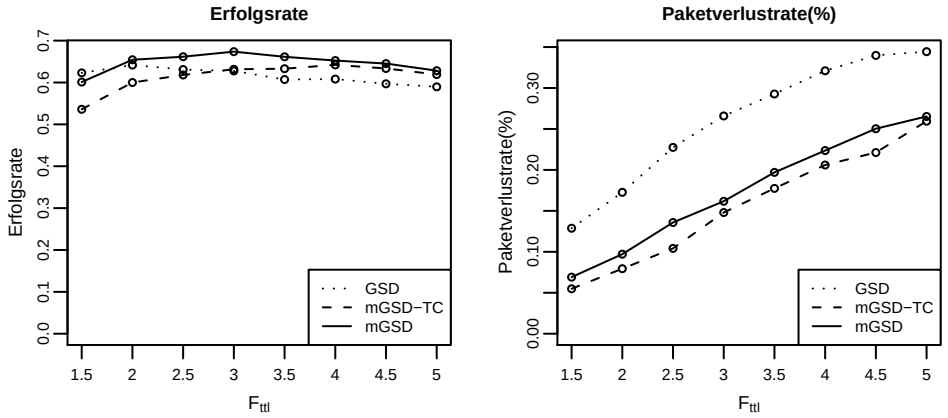

Abbildung 5.19: 1. Kombinationsversuch: Erfolgsrate und Paketverlustrate

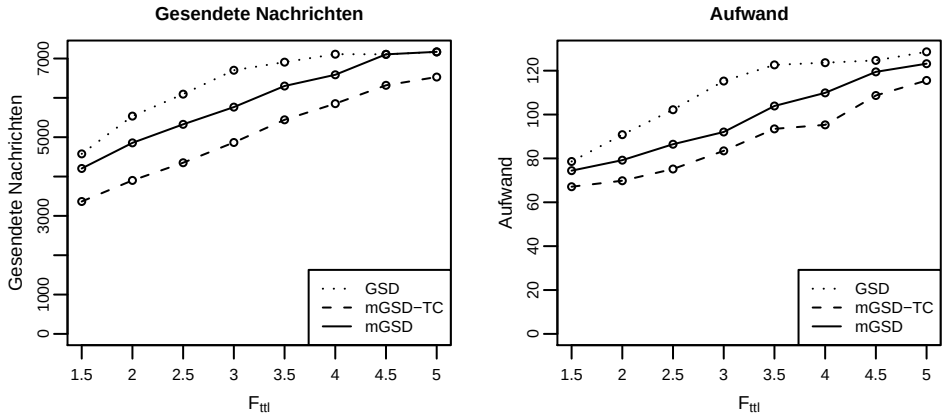

Abbildung 5.20: 1. Kombinationsversuch: Gesendete Nachrichten und Aufwand

GSD. Der Vorsprung von mGSD-TC zu GSD beträgt bis zu *27%*, liegt jedoch immer über *10%*. Der Vorsprung vor mGSD liegt hingegen nur zwischen *6%* und *13%*, ist aber immer noch deutlich.

Zusammenfassung Zusammenfassend lässt sich also sagen, dass mGSD-TC im verwendeten Szenario hinsichtlich der Effizienz am besten abschneidet, sogar besser als mGSD mit der Kombination beider Erweiterungen. Zwar sendet mGSD mehr Nachrichten, erreicht aber auch eine etwas höhere Erfolgsrate. Allerdings ist die Erhöhung der Erfolgsrate nicht so groß, dass die Effizienz über die von mGSD-TC steigt.
Als Wert für den F_{ttl} Faktor empfiehlt sich auch in diesen Versuchen wieder der Wert *3*. mGSD erreicht hier die höchste Erfolgsrate, mGSD-TC hingegen, bei gleicher Erfolgsrate wie GSD, eine wesentlich höhere Effizienz als GSD.

Adaptive Verteilung und Adaptive Lebenszeit, Versuch 2

Während die vorhergehenden Versuche alle mit einem gut konfigurierten GSD Protokoll durchgeführt wurden, soll nun eine Versuchsreihe folgen, bei der das GSD Protokoll nicht optimal konfiguriert ist. Dazu wird die Geschwindigkeit der Knoten erhöht. Zu erwarten ist, dass durch die Erhöhung die Konfiguration des Angebotsabstands und der Lebenszeit der Angebote nicht mehr optimal ist und die Performanz des Protokolls sinkt. Besonders in Bezug auf die Verteilung der Angebote und den Nutzen.
Das mGSD Protokoll passt den Angebotsabstand und die Lebenszeit der Angebote an die Bewegung der Knoten an. Deswegen ist zu erwarten, dass sich die Performanz des mGSD Protokolls weniger verschlechtert als dies beim GSD der Fall ist. So sollte die Verteilung der Dienstangebote und der Nutzen im Vergleich zum GSD Protokoll besser sein.
Zusätzlich wird mGSD mit Adaptiver Lebenszeit (Teil I + II), mGSD-TTL genannt, zum Vergleich herangezogen.

Versuchsaufbau Für die Evaluation wird wieder das Standard-Szenario verwendet. Jedoch bewegen sich die Knoten diesmal mit normal-verteilter Geschwindigkeit

zwischen *1m/s* und *3m/s* bzw. *19m/s* und *5s* Pausenlänge.
Die übrigen Konfigurationsparameter sind in Tabelle 5.8 aufgelistet. mGSD und mGSD-TTL berechnen die Lebenszeit der Angebote beim Service Provider nach Formel 4.11 und passen zusätzlich die Lebenszeit der Angebote beim Empänger nach Formel 4.17 an.

Parameter	**GSD**	**mGSD**	**mGSD-TTL**
t_{adv}	*15s*	-	*15s*
$t_{adv.min}$	-	*10s*	-
$t_{adv.max}$	-	*30s*	-
F_{adv}	-	*1,0*	-
T_N	-	*5*	-
F_{ttl}	-	*3*	*3*
t_{ttl}	*45s*	-	-

Tabelle 5.8: Konfigurationsparameter, 2. Kombinationsversuch

Auswertung Zunächst werden die Veränderungen beschrieben, die bei den Protokollen mit steigender Geschwindigkeit auftreten. Dabei werden die aussagekräftigsten Größen betrachtet.
Die **Verteilung** verschlechtert sich bei allen Protokollen. Die Right-Positives sinken bei GSD um *49%*, bei mGSD um *19%* und bei mGSD-TTL sogar um *69%*. Während für GSD und mGSD dies fast keine Auswirkung auf die Zahl gesendeter Anfragen, die Trefferanzahl und die Anzahl lokaler Suchtreffer hat, sind bei mGSD-TTL deutliche Unterschiede zu sehen: So werden *20%* mehr Anfragen gesendet, *32%* weniger Treffer auf Suchanfragen und auch *40%* weniger lokale Treffer erzielt. Abbildung 5.21 zeigt die Suchtreffer der drei Protokolle.
Beim **Nutzen** sind bei allen Protokollen in Abbildung 5.22 deutliche Unterschiede zu beobachten. So sinkt die Erfolgsrate bei GSD um *29%*, bei mGSD um *26%* und bei mGSD-TTL um *32%*. Der Paketverlust steigt bei allen Protokollen mit steigender Höchstgeschwindigkeit erheblich, wie im rechten Diagramm in Abbildung 5.22 zu sehen ist. Bei GSD um *89%*, bei mGSD um *125%* und bei mGSD-TTL um *57%*.
Wenn man die **Netzwerkbelastung** betrachtet, ist bei GSD und mGSD-TTL eine Abnahme der gesendeten Nachrichten um *13%* bzw. *34%* festzustellen. Abbildung

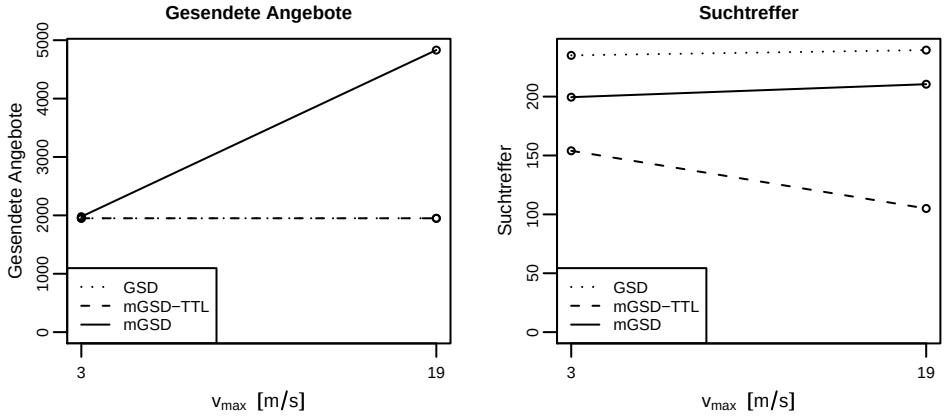

Abbildung 5.21: 2. Kombinationsversuch: Gesendete Angebote und Suchtreffer

5.23 zeigt, dass mGSD hingegen *44%* mehr Nachrichten sendet. Dieser Anstieg ist, wie in Abbildung 5.21 zu sehen[3], in der gestiegenen Anzahl an Angeboten begründet: *144%* mehr als bei einer Höchstgeschwindigkeit von *3m/s*.
Bezüglich der **Effizienz** bedeutet dies: Bei GSD werden pro erfolgreichem Aufruf *25%* mehr Nachrichten gesendet, bei mGSD sogar *92%* mehr. Lediglich mGSD-TTL sendet marginal weniger Nachrichten als bei *3m/s* Höchstgeschwindigkeit.

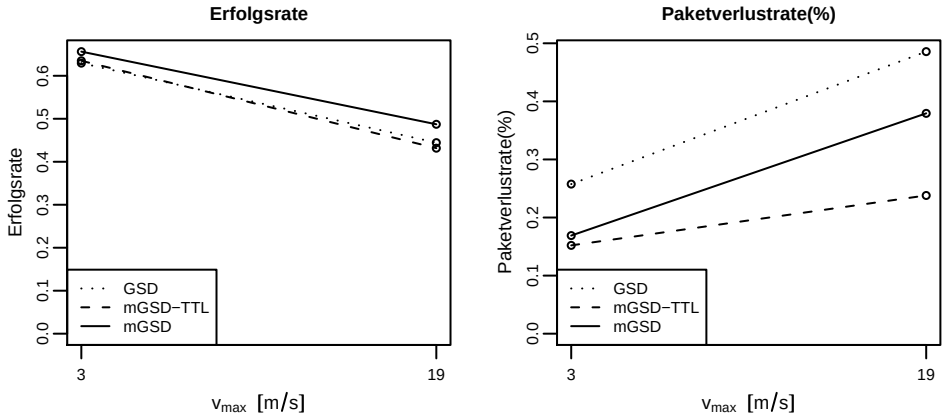

Abbildung 5.22: 2. Kombinationsversuch: Erfolgsrate und Paketverlustrate

[3] In linken Diagramm in Abbildung 5.21 liegen GSD und mGSD-TTL auf der gleichen Linie. Dies kommt dadurch, dass beide den gleichen festen Angebotsabstand verwenden.

Nun werden die Verhältnisse zwischen den einzelnen Protokollen bei einer Höchstgeschwindigkeit von *19m/s* beschrieben. Dabei wird GSD als Referenzprotokoll verwendet.
Hinsichtlich der **Verteilung** schneiden mGSD und mGSD-TTL deutlich schlechter ab als GSD. So muss mGSD *35%* und mGSD-TTL sogar *77%* mehr Suchanfragen senden als GSD. Abbildung 5.21 zeigt die Anzahl der Suchtreffer. So sinkt die Anzahl der Suchtreffer bei mGSD und mGSD-TTL um *12%* bzw. *56%* und die Anzahl lokaler Treffer sinkt um *28%* bzw. *64%* im Vergleich zu GSD. Nur bei den Right-Positives hat mGSD *97%* Vorsprung vor GSD. mGSD-TTL liegt dagegen um *42%* hinter GSD zurück.
Abbildung 5.22 zeigt die Erfolgsrate und den Paketverlust. So liegt die **Erfolgsrate** bei mGSD *10%* vor GSD, mGSD-TTL liegt hier marginal hinter GSD. Beim Paketverlust wiederum schneiden beide Protokolle deutlich besser ab als GSD: mGSD verliert *22%* und mGSD-TTL sogar *51%* weniger Nachrichten.
Wenn man die **Netzwerkbelastung** in Abbildung 5.23 beobachtet, lässt sich feststellen, dass mGSD im Vergleich zu GSD *48%* mehr Nachrichten sendet. Dies liegt, wie Abbildung 5.21 belegt, vor allem an der extrem gestiegenen Anzahl an Angeboten: mGSD sendet *148%* mehr Angebote, da es den Angebotsabstand an die gestiegene Geschwindigkeit anpasst. mGSD-TTL hingegen sendet *43%* weniger Nachrichten als GSD. Dieser Effekt kommt durch die Einsparung von Nachrichten im Protokollverlauf zustande.

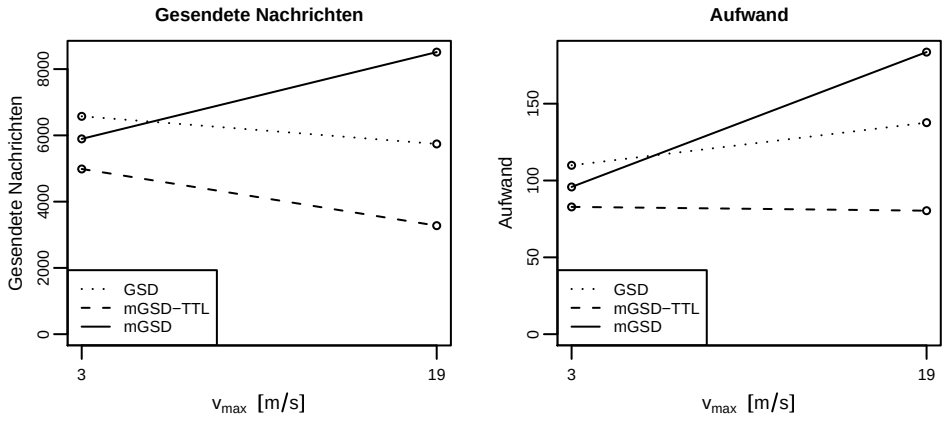

Abbildung 5.23: 2. Kombinationsversuch: Gesendete Nachrichten und Aufwand

Für die **Effizienz** bedeutet dies: mGSD sendet pro erfolgreichem Aufruf *33%* mehr Nachrichten als GSD. mGSD-TTL hingegen liegt weit vor GSD und sendet pro erfolgreichem Aufruf *42%* weniger Nachrichten.

Zusammenfassung Betrachtet man die Ergebnisse, kommt man zu dem Schluss, dass die Anpassung des Angebotsabstands an die Geschwindigkeit bei mGSD in dem verwendeten Szenario wenig Einfluss auf die Gesamtperformanz des Protokolls hat. So werden etwa gleich viele Suchtreffer erzielt und auch etwa gleich viele Dienstaufrufe gesendet. Die Verteilung hängt hier also scheinbar nicht wie vermutet zuerst vom Angebotsabstand ab, sondern von anderen Faktoren. Eine mögliche Erklärung für dieses Phänomen könnte die Netzwerkstruktur sein. So lässt sich vermuten, dass die Dichte der Knoten zu groß ist. Das würde bedeuten, dass sich viele Knoten auf einer kleinen Fläche befinden. Demnach ist die Schwierigkeiten nicht einen Service Provider zu finden, sondern ihn auch zu nutzen. So gehen ja bei GSD und mGSD *90%* bis *125%* mehr Nachrichten verloren als bei niedriger Geschwindigkeit.

Da zugleich der Nutzen, also die Erfolgsrate, nicht viel größer ist als bei GSD, verringert sich die Effizienz drastisch. Hier werden viel mehr Nachrichten gesendet, ohne einen dazu im Verhältnis stehenden Nutzen. Deshalb ist die *10%* höhere Erfolgsrate nicht überzubewerten.
Im Gegensatz dazu schneidet mGSD-TTL erheblich besser ab. Es sendet nicht mehr Angebote und erreicht fast die gleiche Erfolgsrate wie GSD. Dafür braucht es jedoch deutlich weniger Nachrichten und weist so eine um *42%* höhere Effizienz auf als GSD. Auch bleibt die Effizienz von mGSD-TTL mit steigender Geschwindigkeit quasi gleich.

5.3.4 Gesamteindruck

Nach der detaillierten Auswertung der Simulationsergebnisse werden die zentralen Ergebnisse abschließend noch einmal kurz zusamengetragen.

In den Simulationen hat sich gezeigt, dass der adaptive Konfigurationsmechanismus die Effizienz des Protokolls deutlich verbessert. So sendet mGSD mit Adaptiver Verteilung und Adaptiver Lebenzeit im Mittel über *10%* weniger Nachrichten pro

erfolgreichem Dienstaufruf als GSD. Dazu erreicht mGSD auch eine leicht höhere Erfolgsrate als GSD.
Wenn man jedoch nur die Effizienz betrachtet, erreicht der alleinige Einsatz der Adaptiven Lebenszeit die besten Effizienzwerte. So werden im Mittel pro erfolgreichem Dienstaufruf etwa *20%* weniger Nachrichten gesendet. Dabei wird jedoch immer noch die gleiche Erfolgsrate erreicht wie bei GSD.

Bei steigender Geschwindigkeit in den Simulationen hat sich mGSD hingegen nicht bewährt. So wurde im Vergleich zu GSD zwar eine höhere Erfolgsrate erreicht, dazu aber deutlich mehr Nachrichten gesendet. Es bleibt also noch herauszufinden, in welchen Szenarien mGSD hier eine Effizienzverbesserung erreicht.
Demgegenüber hat mGSD bei alleinigem Einsatz der Adaptiven Lebenszeit jedoch überzeugt. So wurden bei steigender Geschwindigkeit *42%* weniger Nachrichten pro erfolgreichem Dienstaufruf gesendet als bei GSD. Insofern hat sich der Vorsprung zu GSD, im Vergleich zu den Versuchen mit niedrigerer Geschwindigkeit, nochmal erhöht.

Zusammenfassend kann man sagen, dass die Einbeziehung der Positions- und Bewegungsinformationen durch die Adaptive Konfiguration einen wirklichen Mehrwert bedeutet. Besonders die Anpassung der Lebenszeit der Angebote führt zu einer deutlich höheren Aktualität der Caches und verbessert die Effizienz der Dienstfindung.

5.4 Evaluation des Parallelen Routings

Der parallele Einsatz zweier Routingverfahren ist im GSD Protokoll nicht möglich. Deshalb kann hier kein direkter Vergleich zwischen mGSD und GSD erfolgen. Ebenso ist es nicht das Thema dieser Arbeit verschiedene Routingverfahren zu vergleichen. Vielmehr soll überprüft werden, inwieweit der Ansatz, in einer Simulation verschiedene Routingverfahren einzusetzen, sinnvoll ist. Dazu werden die zwei Einsatzfälle des Parallelen Routings separat untersucht.

5.4.1 Entfernte Dienste

Die Erweiterung der entfernten Dienste ist als zusätzliches Feature des mGSD Protokolls vorgesehen. So soll hier an einer Versuchsreihe überprüft werden, wie gut die Erweiterung funktioniert. Von einem entfernten Dienst spricht man bei Dienstanfragen, die nicht auf die eigene Umgebung abzielen, sondern auf ein entferntes Gebiet. Ohne geographisches Routing sind solche Anfragen nicht möglich. Deshalb liegt der Nutzen der Positions- und Bewegungsinformationen schon allein in dem Punkt, dass solche Anfragen im erweiterten Protokoll möglich sind.

Zu Evaluation werden folgende Kriterien bewertet:

Such-Erfolg Wie viele Treffer werden auf entfernte Suchanfragen erzielt? Die Versuche werden unter der Annahme durchgeführt, dass immer ein Service Provider mit einem passenden Dienst im Suchgebiet vorhanden ist.

Antwort-Erfolg Wie viele Antworten kommen beim Client an?

Aufruf-Erfolg Wie oft ist ein Aufruf erfolgreich?

Entfernung Client-Zielgebiet Bei welchen Entfernungen zum Zielgebiet kommt die Anfrage im Zielgebiet an?

Entfernung Client-Antwortender Knoten Wie unterscheiden sich die Entfernung vom Client zum Zielgebiet und die Entfernung vom Client zum antwortenden Knoten? Stammt die Antwort wirklich aus dem entfernten Suchgebiet bzw. wie weit ist sie davon entfernt?

Versuchsaufbau Das Versuchsszenario besteht aus 50 Knoten $N_1 ... N_{50}$, die sich auf einem Feld von *200m x 200m* Größe nach dem Random-Waypoint-Model bewegen. Die Pausenlänge beträgt *5s* und die Geschwindigkeit liegt zwischen *0m/s und 2m/s*. Zusätzlich werden dem Szenario *10* Knoten $SP_1 ... SP_{10}$ hinzugefügt, die als Service Provider fungieren und sich nicht bewegen. Diese Knoten sind auf einer Diagonale in gleichmäßigen Abständen über das Feld verteilt und bieten alle den gleichen Dienst *S0* an. Andere Knoten bieten selber keine Dienste an, kennen jedoch die Positionen der designierten Service Provider und wissen welcher Dienst von den Service Providern angeboten wird. Regelmäßig startet ein zufällig gewählter

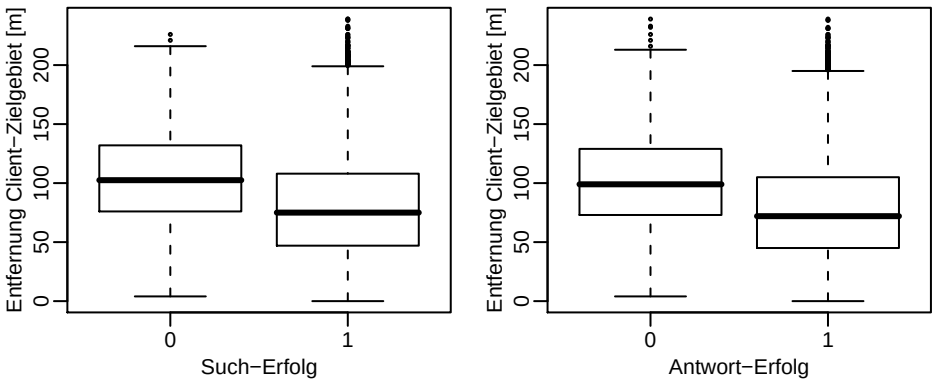

Abbildung 5.24: Enfernte Dienste: Zusammenhang von Such- bzw. Antwort-Erfolg und Entfernung zum Zielgebiet

Knoten aus $N_1 ... N_{50}$ eine entfernte Suchanfrage nach dem Dienst *S0*. Dazu wählt er zufällig einen der designierten Service Provider aus und sendet die Anfrage an dessen Position. Der Radius des Suchgebiets wird auf *15m* konfiguriert. Alle Nachrichten während der Simulation werden mit geographischem Routing gesendet. Nur eine eventuelle lokale Weiterleitung der Suchanfrage geschieht mittels herkömmlichem Routing. Insgesamt werden etwa *28.000* entfernte Suchanfragen ausgewertet. Die mittlere absolute Abweichung vom Median liegt dabei in der Regel zwischen *3%* und *10%*.

Auswertung Das Feature der entfernten Suchanfragen funktioniert in dem verwendeten Szenario recht gut. So kann bei *65%* der Suchanfragen ein Dienstaufruf gesendet werden: Ausgehend von *238* Suchanfragen werden durchschnittlich *154* Dienstaufrufe gesendet. Dabei gehen etwa *15%* der Antworten auf dem Weg zum Client verloren. *91%* der Dienstaufrufe erreichen den Service Provider. Die Bestätigung des Service Providers erreicht in *94%* der Fälle den Client. Die Erfolgsrate liegt damit bei etwa *55%*.

Abbildung 5.24 zeigt den Zusammenhang von Such- bzw. Antwort-Erfolg und der Entfernung vom Client zum Zielgebiet. Wie zu sehen ist, besteht zwischen der Entfernung von Client zum Zielgebiet und dem Erfolg der Suchanfrage ein deutlicher

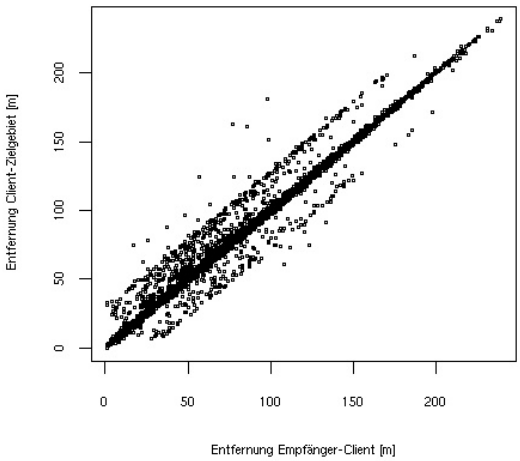

Abbildung 5.25: Enfernte Dienste: Gegenüberstellung der Entfernungen vom Client zum Zielgebiet und vom Sender der Antwort zum Client

Zusammenhang. So liegt der Median der Entfernung vom Client zum Zielgebiet für ankommende Anfragen bei *75m*, für Anfragen, die nicht ankommen, jedoch bei *102m*. Je weiter das Suchgebiet entfernt ist, desto wahrscheinlicher ist es also, dass die Anfrage nicht beantwortet wird. Auch wird es unwahrscheinlicher, dass im Fall eines Suchtreffers die Antwort den Client erreicht. In den Fällen, bei denen die Antwort den Client erreicht, liegt die Entfernung von Client zum Zielgebiet ungefähr bei *72m*. In den Fällen, bei denen der Client keine Antwort bekommt, liegt der Median der Entfernung vom Client zum Zielgebiet bei *99m*.

Die Antwort auf eine Suchanfrage stammt in den meisten Fällen wirklich aus dem Zielgebiet. Aus der Gegenüberstellung der Entfernung vom Client zum Zielgebiet und der Entfernung vom Sender der Antwort zum Client ergibt sich Abbildung 5.25. Aus der deutlichen Annäherung an eine Diagonale lässt sich folgern, dass der antwortende Knoten nahezu die gleiche Entfernung zum Client hat wie das Zielgebiet.

Zusammenfassung Das Feature der entfernten Suchanfragen funktioniert also wie erwartet. Die Paketverluste halten sich dabei in Grenzen, so dass geographisches

Routing im verwendeten Szenario recht gut funktioniert. Der Erfolg einer solchen Anfrage ist jedoch auch deutlich von der Entfernung zum Zielgebiet abhängig. Im verwendeten Szenario liegt diese bei ca. *2* bis *3* Funkreichweiten. Da in Einzelfällen aber auch Anfragen über deutlich größere Entfernungen erfolgreich sind, scheint dieser Effekt im Routingprotokoll begründet zu sein.

5.4.2 Dynamische Routing Entscheidung

Die Erweiterung *Dynamische Routing Entscheidung* verwendet die Positions- und Bewegungsinformationen der Knoten um zu entscheiden, welches Routingprotokoll für eine bestimmte Situation geeignet ist. Bei der Evaluation dieser Erweiterung konzentriert sich die Untersuchung darauf, ob der Einsatz von geographischem Routing innerhalb des GSD Protokolls Sinn hat. Geht man davon aus, dass GPSR im Gegensatz zu AODV keine Routen berechnen muss, kann man erwarten, dass in den Fällen, bei denen AODV erst eine Route berechnen muss, GPSR Nachrichten schneller sendet. So sollten sich die Antwortzeiten verringern. Die Erweiterung sieht vor, in solchen Fällen geographisch zu routen, in denen ein Knoten weiß, dass sich der Empfänger um mehr als T_{Geo} von seiner früheren Position entfernt hat. Dabei

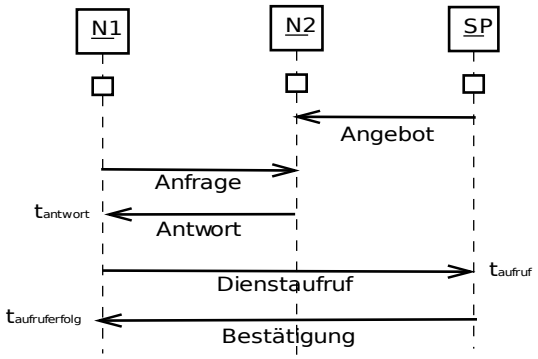

Abbildung 5.26: Nachrichtenüberblick und Zeiten bei einer Suchanfrage

werden pro Suchanfrage drei Zeiten betrachtet. Zur Erklärung sind diese auch in Abbildung 5.26 eingezeichnet:

$t_{antwort}$ Die Zeit $t_{antwort}$, die vom Senden der Suchanfrage vergeht, bis der Client die erste Antwort erhält.

t_{aufruf} Die Zeit t_{aufruf}, die vom Senden der Suchanfrage vergeht, bis der Service Provider den Dienstaufruf des Clients erhält.

$t_{aufruferfolg}$ Die Zeit $t_{aufruferfolg}$, die vom Senden der Suchanfrage vergeht, bis der Client die Bestätigung für den Dienstaufruf erhält.

Zusätzlich wird auch der Paketverlust betrachtet.

Versuchsaufbau Das Szenario für die Versuche besteht wieder aus einem rechteckigen Feld mit *200m x 200m* Größe. Auf ihm bewegen sich 50 Knoten nach dem Random-Waypoint-Model mit Geschwindigkeiten zwischen *1m/s* und *2m/s* und *5s* Pausenlänge. Zusätzlich gibt es einen designierten Client, der sich mit *2m/s* diagonal über das Feld hin- und herbewegt und dabei regelmäßig Suchanfragen sendet. Die übrigen Knoten fungieren alle als Service Provider und bieten jeweils *10%* der verfügbaren Dienste an. Durch die stärkere Bewegung des Clients sollen gerade solche Fälle erzwungen werden, in denen geographisches Routing eingesetzt wird. Der Grenzwert T_{Geo} wird auf eine halbe Funkreichweite, also *15m* festgesetzt.

Für die Auswertung werden die drei oben erwähnten Zeiten gemessen. Dabei werden diese nur für solche Nachrichten gemessen, die geographisch geroutet werden bzw. geographisch geroutet würden, falls das parallele Routing aktiviert wäre. Insgesamt werden *1.000* Simulationen durchgeführt, davon jeweils *500* mit parallelem Routing und *500* nur mit AODV als Routingprotokoll.

Auswertung Während einer Simulation werden im Mittel etwa *6%* der in Frage kommenden Nachrichten geographisch geroutet. Davon sind etwa *80%* Dienstaufrufe, die an den Service Provider gesendet werden. Etwa *16%* sind Antworten auf Suchanfragen. Der Rest sind Bestätigungen für den Dienstaufruf an den Client. Nicht verwunderlich ist dabei, dass Dienstaufrufe am häufigsten geographisch geroutet werden. Denn hier ist die letzte verfügbare Positionsinformation des Zielknotens am ältesten.

Generell ist zu beobachten, dass der Paketverlust beim geographischen Routing sehr hoch ist. So kommen von den betreffenden Nachrichten mit geographischem Routing nur etwa *12%* bis *20%* an, im Vergleich zu den Simulationen ohne geographi-

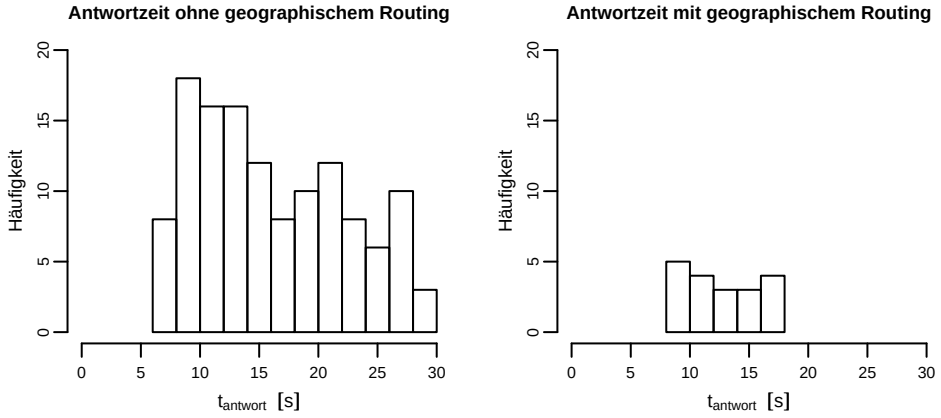

Abbildung 5.27: Dynamische Routing Entscheidung: Antwortzeit

sches Routing. Das geographische Routing funktioniert also im verwendeten Setup schlecht.

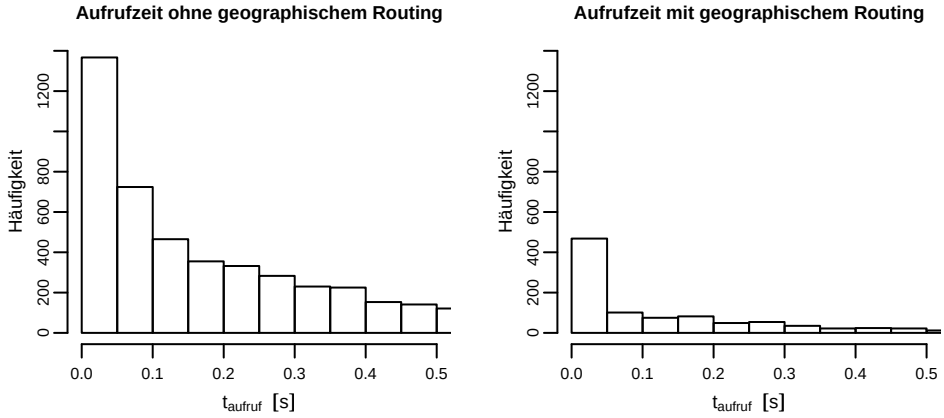

Abbildung 5.28: Dynamische Routing Entscheidung: Aufrufzeit

Wenn man die gemessenen Zeiten betrachtet, fällt auf, dass der Median der Zeit t_{aufruf} viel geringer ist als bei den Zeiten $t_{antwort}$ und $t_{aufruferfolg}$. Dies liegt daran, dass die gesendeten Aufrufe zum größten Teil aus lokalen Suchtreffern hervorgegangen sind. Insgesamt sind die Mediane der Zeiten (siehe Tabelle 5.9) bei aktiviertem parallelen Routing stets kleiner als ohne. Die Abbildungen 5.27, 5.28 und 5.29 zeigen die Häufigkeiten der gemessenen Zeiten. Auch hier ist deutlich zu sehen, dass we-

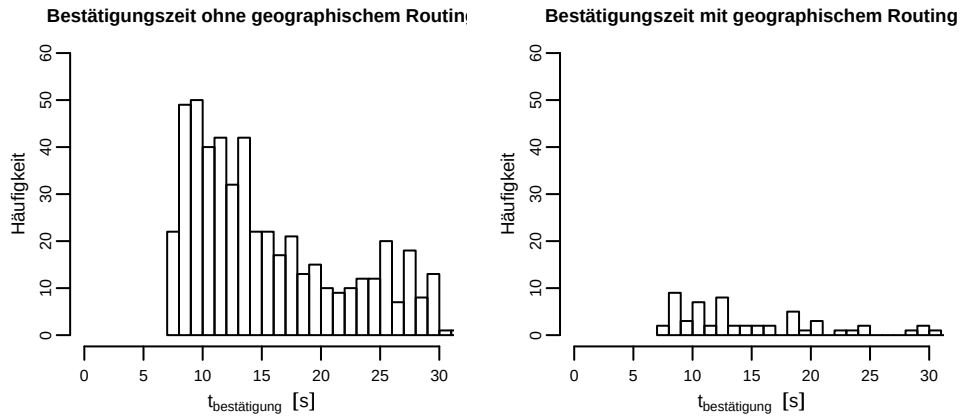

Abbildung 5.29: Dynamische Routing Entscheidung: Bestätigungszeit

sentlich weniger geographisch geroutete Nachrichten ankommen. Zudem treten beim geographischen Routing im Verhältnis mehr kurze Antwortzeiten auf. Das bedeutet, länger laufende Nachrichten gehen beim geographischen Routing mit höherer Wahrscheinlichkeit verloren als kürzer laufende Nachrichten. So ist es erklärlich, dass die Mediane der Zeiten beim geographischen Routing trotz schlechterer Performanz geringer sind. Ein Vergleich der Zeiten ist deshalb nicht aussagekräftig, da die Datenbasis zu klein ist.

	$t_{antwort}$	t_{aufruf}	$t_{bestätigung}$
mit geographischem Routing	*12.06s*	*0.16s*	*12.56s*
ohne geographischem Routing	*14.80s*	*0.28s*	*13.51s*

Tabelle 5.9: Vergleich der gemessenen Zeiten

Zusammenfassung Bei obiger Konfiguration wird geographisches Routing kaum eingesetzt. Nur in *6%* der Fälle kommt es zum Zug. Bei den Versuchen wurde der Grenzwert T_{Geo} auf eine halbe Funkreichweite gesetzt, also auf *15m*. Das Herabsetzen dieses Grenzwertes erscheint wenig sinnvoll, wenn man bedenkt, dass sich dann ein Knoten bei zurückgelegter Grenzentfernung fast nicht von seiner früheren Position entfernt hätte.

Zusammenfassend lässt sich also sagen, dass geographisches Routing in der verwendeten Konfiguration, anders als vermutet, keinerlei Vorteile bringt. Im Gegenteil: Es gehen im Vergleich zum normalen Routingverfahren wesentlich mehr Nachrichten verloren. Eine mögliche Erklärung dafür ist, dass die Fälle, in denen im verwendeten Setup geographisch geroutet wird, per se schon schwierige Fälle sind. Das heißt, die betreffenden Fälle sind Ausnahmen, bei denen die Nachrichten deutlich länger als normal benötigen und damit die Positionsdaten, auf die das Routingprotokoll aufbaut, eventuell veraltet sind.
Es bleibt also herauszufinden, in welchen Szenarien geographisches Routing Vorteile gegenüber herkömmlichem Routing bietet.

5.4.3 Gesamteindruck

In den Simulationen hat sich gezeigt, dass paralleles Routing von Nutzen ist, weil dadurch entfernte Suchanfragen möglich gemacht werden, die im GSD Protokoll nicht möglich waren. Hier nutzen die Positions- und Bewegungsinformationen also durchaus. Das Nutzen des geographischen Routings zur Verringerung der Antwortzeiten in bestimmten Fällen hat sich allerdings in dem verwendeten Szenario als nicht erfolgreich herausgestellt.

6 Zusammenfassung und Bewertung

6.1 Zusammenfassung

Die Dienstfindung ist ein allgemeines Problem in Netzwerken. In MANETs bekommt sie jedoch einen besonderen Stellenwert, weil bekannte Methoden, wie z.B. zentrale Dienstverzeichnisse, für MANETs nicht geeignet sind. In dieser Diplomarbeit sollte untersucht werden, inwiefern die Positions- und Bewegungsinformationen mobiler Knoten hilfreich für die Dienstfindung in MANETs sind. Dazu wurde ein existierendes Service Discovery Protokoll, nämlich das GSD Protokoll, entsprechend erweitert. Das entworfene mGSD Protokoll weist im Vergleich zum GSD Protokoll zwei Erweiterungen auf: Den adaptiven Konfigurationsmechanismus und die Möglichkeit parallel zwei Routingprotokolle einzusetzen.

Der adaptive Konfigurationsmechanismus nutzt die Positions- und Bewegungsinformationen dazu, das Protokoll dynamisch an die jeweilige Umgebung anzupassen. So passt ein Knoten den Zeitabstand zwischen zwei Angeboten sowohl an seine eigene Bewegung als auch an seine Umgebung an. Kommen viele neue Knoten in seine Umgebung, sendet er zusätzliche Angebote, um diesen die eigenen Dienste bekannt zu machen.
Ebenso passt ein Knoten die Lebenszeit seiner Angebote an seine Bewegung an. Zusätzlich kann der Empfänger eines Angebots die Lebenszeit verringern. Anhand seiner Bewegung relativ zum Sender des Angebots, berechnet er, wie lange eine Funkverbindung zum Sender noch möglich ist und passt die Lebenszeit des Angebots entsprechend an.
Weiterhin nutzt das mGSD Protokoll die Mobilitätsinformationen der Knoten, um geographisches Routing zu ermöglichen. Dabei wurde ein Verfahren entwickelt, das

den gleichzeitigen Einsatz eines geographischen und eines herkömmlichen Routingprotokolls ermöglicht. So kann, je nach Situation, das passende Routingprotokoll gewählt werden. Durch diese Option werden Suchanfragen in ein entferntes Gebiet ermöglicht. Dies war im GSD Protokoll nicht möglich.

Nach der Konzeption des mGSD Protokolls wurde dann durch Simulationen im NS-2 evaluiert, ob und inwieweit diese Erweiterungen die Dienstfindung verbessern. Die Ergebnisse der Untersuchung zeigen, dass durch die Einbeziehung der Mobilitätsinformationen vor allem die Effizienz der Dienstfindung verbessert wird. So helfen die Positions- und Bewegungsinformationen in den verwendeten Szenarien dabei, qualitativ minderwertige Suchtreffer zu vermeiden. In den Simulationen konnte so das Verhältnis von Kosten zu Nutzen deutlich verbessert werden. Bei gleichbleibender Anzahl erfolgreicher Dienstaufrufe wurde die Netzwerkbelastung deutlich reduziert. Das Reduzieren der Netzwerkbelastung ist dabei gerade in MANETs sehr wertvoll. Der parallele Einsatz des geographischen Routings konnte hingegen in den verwendeten Szenarien nicht vollständig überzeugen. Im Allgemeinen zeigt der Einsatz des geographischen Routings keine Vorteile, da die Paketverluste zu hoch sind. Jedoch ermöglicht das geographische Routing im mGSD Protokoll entfernte Suchanfragen. Die Untersuchung, in welchen Situationen bzw. Szenarien das Parallele Routing sinnvoll ist, hätte den Rahmen dieser Arbeit jedoch deutlich gesprengt. Die entfernten Suchanfragen sind aber sicherlich eine wertvolle Funktion eines Dienstfindungsprotokolls.

6.2 Ausblick

Neben den in dieser Diplomarbeit gewonnenen Erkenntnissen bleiben jedoch noch einige Punkte offen und bieten Platz für weitere Untersuchungen. So gelten die Ergebnisse dieser Arbeit sicherlich vorerst nur für die verwendeten Szenarien. Ob sich die Ergebnisse auch in andersgearteten Szenarien bestätigen lassen, wäre eine interessante Untersuchung.

Weiterhin bleibt auch eine abschließende Bewertung des Einsatzes zweier Routingprotokolle offen. So ist ungeklärt, woran die schlechte Performanz des geographischen

Routings in den durchgeführten Simulationen liegt und auch wie groß die zusätzliche Netzwerkbelastung durch diese Funktion ist. Hier könnte auch eine tiefergehende Integration beider verwendeter Routingverfahren weitere Untersuchungen wert sein.

Literaturverzeichnis

[1] *Boxplot - Wikipedia.* Version: Mai 2008. http://de.wikipedia.org/wiki/Boxplot

[2] *Implementation of GSD for ns2.* Version: Mai 2008. http://sourceforge.net/projects/gsd4ns2

[3] *Ke Liu's NS2 Code and Q&A.* Version: Mai 2008. http://www.cs.binghamton.edu/~kliu/research/ns2code/index.html#gpsr

[4] *NS-2 Simulator.* Version: Mai 2008. http://nsnam.isi.edu/nsnam/index.php/Main_Page

[5] Böse, Joos-Hendrik ; Bregulla, Frank ; Hahn, Katharina ; Scholz, Manuel: Adaptive Data Dissemination in Mobile ad-hoc Networks. In: Cremers, Armin B. (Hrsg.) ; Manthey, Rainer (Hrsg.) ; Martini, Peter (Hrsg.) ; Steinhage, Volker (Hrsg.): *GI Jahrestagung (2)* Bd. 68, GI, 2005 (LNI). – ISBN 3–88579–397–0, S. 528–532

[6] Broch, Josh ; Maltz, David A. ; Johnson, David B. ; Hu, Yih-Chun ; Jetcheva, Jorjeta G.: A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In: *MOBICOM*, 1998, S. 85–97

[7] Chakraborty, D. ; Perich, F. ; Avancha, S. ; Joshi, A.: DReggie: A smart service discovery technique for E-Commerce applications. In: *20th Symposium on Reliable Distributed Systems*, 2001

[8] Chakraborty, Dipanjan ; Joshi, Anupam ; Finin, Tim ; Yesha, Yelena: GSD: A Novel Group-based Service Discovery Protocol for MANETs. In: *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN)*. Stockholm. Sweden, September 2002

[9] Guttman, Erik: Service Location Protocol: Automatic Discovery of IP Network Services. In: *IEEE Internet Computing* 3 (1999), Nr. 4, S. 71–80. http://dx.doi.org/http://dx.doi.org/10.1109/4236.780963. – DOI http://dx.doi.org/10.1109/4236.780963. – ISSN 1089–7801

[10] Johnson, David B. ; Maltz, David A.: Dynamic Source Routing in Ad Hoc Wireless Networks. In: Imielinski (Hrsg.) ; Korth (Hrsg.): *Mobile Computing* Bd. 353. Kluwer Academic Publishers, 1996

[11] Karp, Brad ; Kung, H. T.: GPSR: Greedy Perimeter Stateless Routing For Wireless Networks. In: *MOBICOM*, 2000, S. 243–254

[12] Klein, M. ; König-Ries, B. ; Obreiter, P.: Lanes - A Lightweight Overlay for Service Discovery in Mobile Ad Hoc Networks. In: *3rd Workshop on Applications and Services in Wireless Networks (ASWN2003)*, 2003

[13] Klein, M. ; König-Ries, B. ; Obreiter, P.: Service Rings - A Semantic Overlay for Service Discovery in Ad hoc Networks. In: *Sixth International Workshop on Network-Based Information Systems (NBIS2003)*, 2003

[14] Kozat, Ulas C. ; Tassiulas, Leandros: Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In: *IEEE INFOCOM* IEEE, 2003

[15] Lee, Choonhwa ; Helal, Abdelsalam ; Desai, N. ; Verma, V. ; Arslan, B.: Konark: a system and protocols for device independent, peer-to-peer discovery and delivery of mobile services. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part A* 33 (2003), Nr. 6, S. 682–696

[16] Mian, Adnan N. ; Beraldi, Roberto ; Baldoni, Roberto: Survey of Service Discovery Protocols in Mobile Ad Hoc Networks / Dipartimento di Informatica e Sistemistica "Antonio Ruberti". Università degli Studi di Roma "La Sapienza", Rome, Italy, April 2006. – Forschungsbericht

[17] Nagler, Jürgen ; Kargl, Frank ; Schlott, Stefan ; Weber, Michael: Ein Framework für MANET Routing Protokolle. In: Weber, Michael (Hrsg.) ; Kargl, Frank (Hrsg.): *WMAN* Bd. 11, GI, 2002 (LNI). – ISBN 3–88579–341–5, S. 153–165

[18] Perkins, Charles E. ; Belding-Royer, Elizabeth M.: Ad-hoc On-Demand Distance Vector Routing. In: *WMCSA*, IEEE Computer Society, 1999. – ISBN 0–7695–0025–0, S. 90–100

[19] Ratsimor, Olga ; Chakraborty, Dipanjan ; Joshi, Anupam ; Finin, Timothy W.: Allia: alliance-based service discovery for ad-hoc environments. In: Viveros, Marisa S. (Hrsg.) ; Lei, Hui (Hrsg.) ; Wolfson, Ouri (Hrsg.): *Workshop Mobile Commerce*, ACM, 2002. – ISBN 1–58113–600–5, S. 1–9

[20] Schiller, Jochen H. (Hrsg.) ; Voisard, Agnès (Hrsg.): *Location-Based Services*. Morgan Kaufmann, 2004. – ISBN 1–55860–929–6

[21] Zhu, Feng ; Mutka, Matt W. ; Ni, Lionel M.: Splendor: A Secure, Private, and Location-Aware Service Discovery Protocol Supporting Mobile Services. In: *PerCom*, IEEE Computer Society, 2003, S. 235–242

A Boxplots

Versuch Adaptive Verteilung Teil I

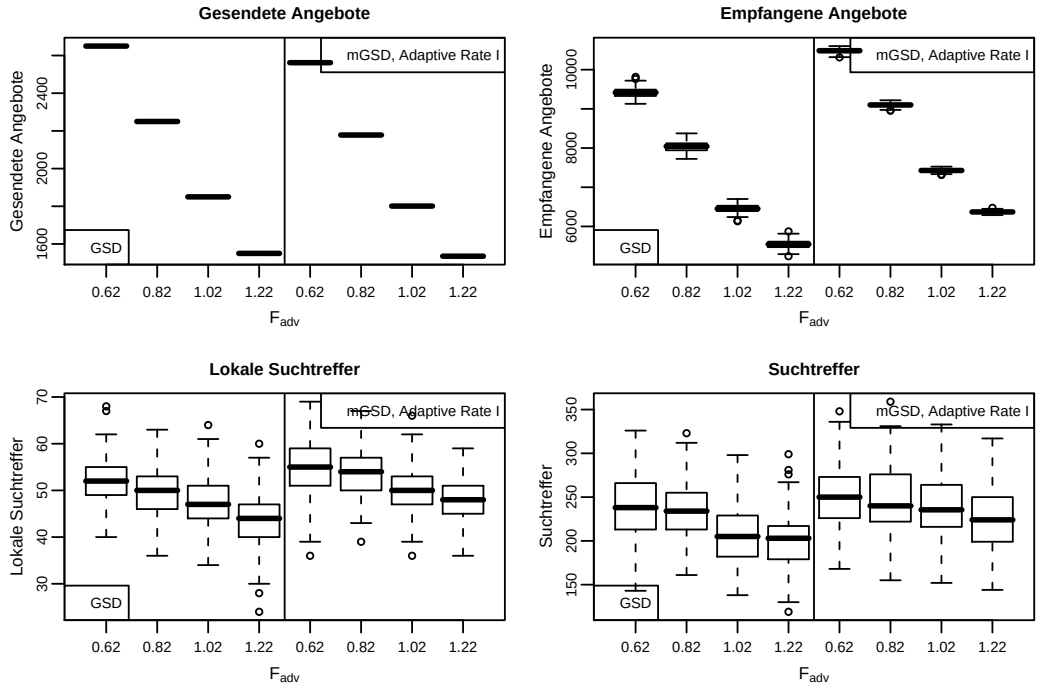

Abbildung A.1: Boxplots: Adaptive Verteilung Teil I (1)

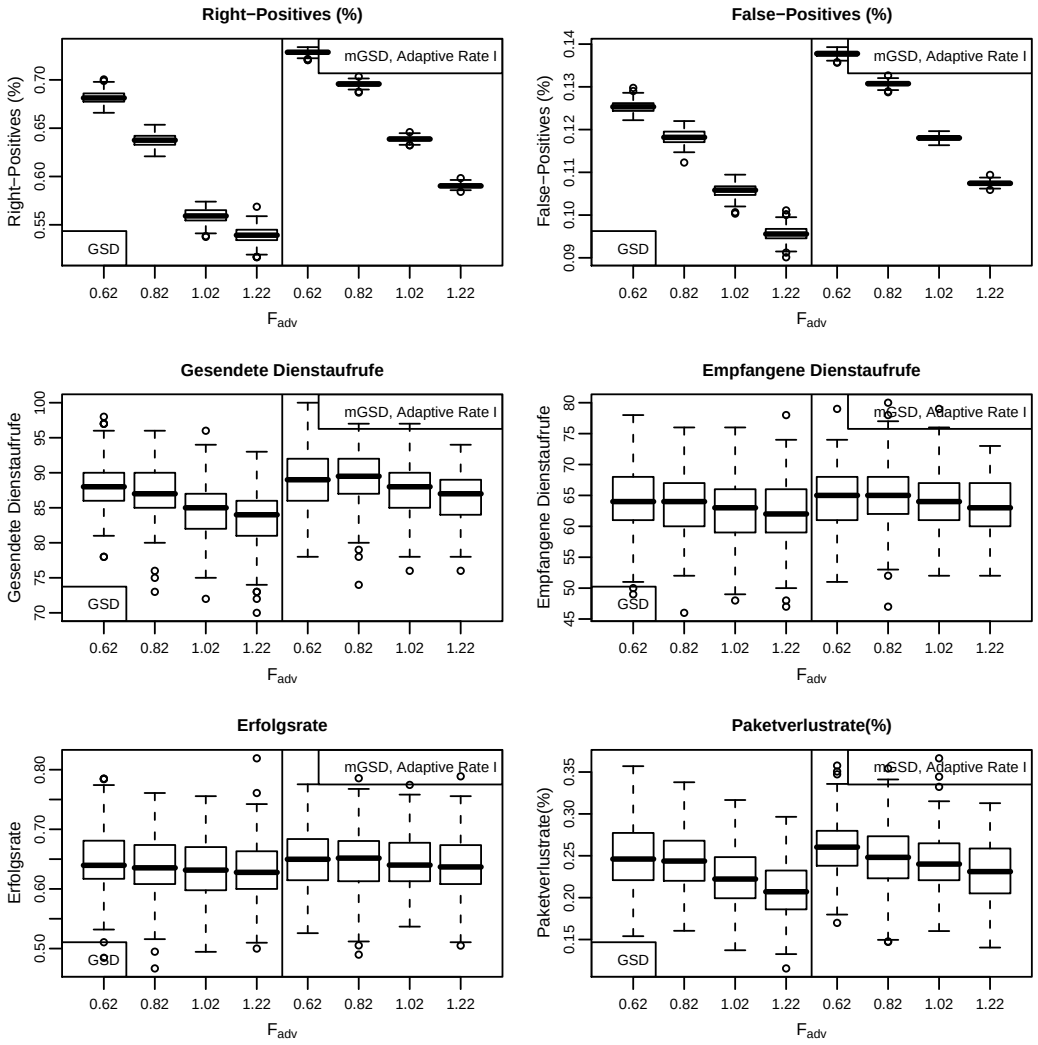

Abbildung A.2: Boxplots: Adaptive Verteilung Teil I (2)

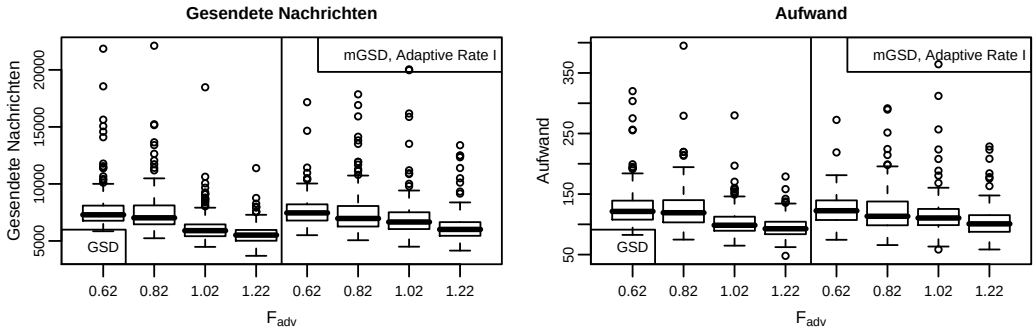

Abbildung A.3: Boxplots: Adaptive Verteilung Teil I (3)

Versuch Adaptive Verteilung Teil II

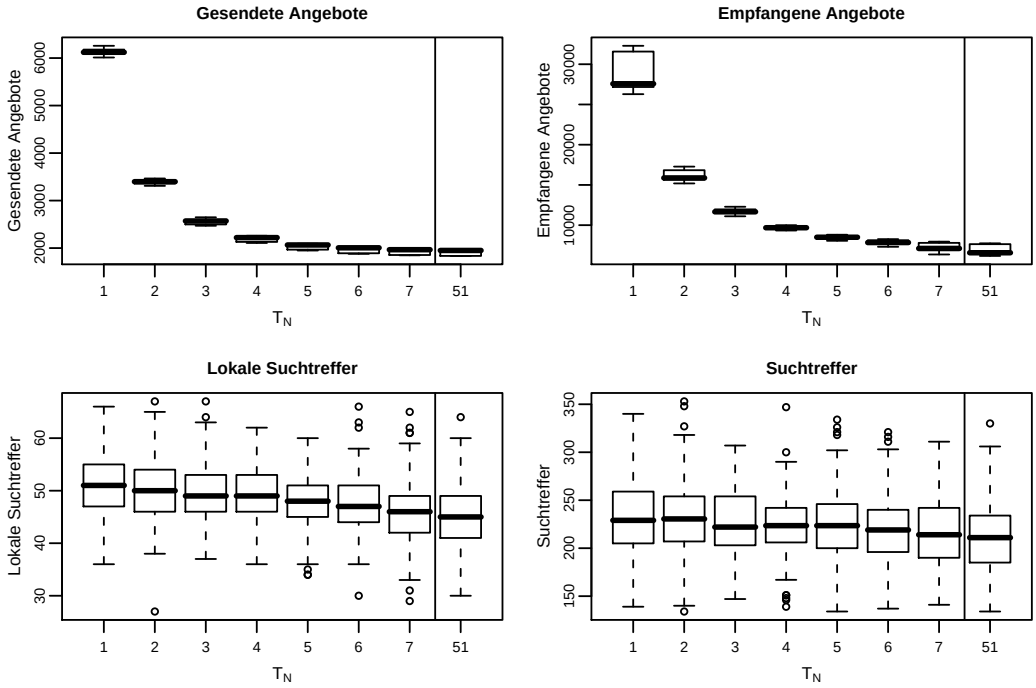

Abbildung A.4: Boxplots: Adaptive Verteilung Teil II (1)

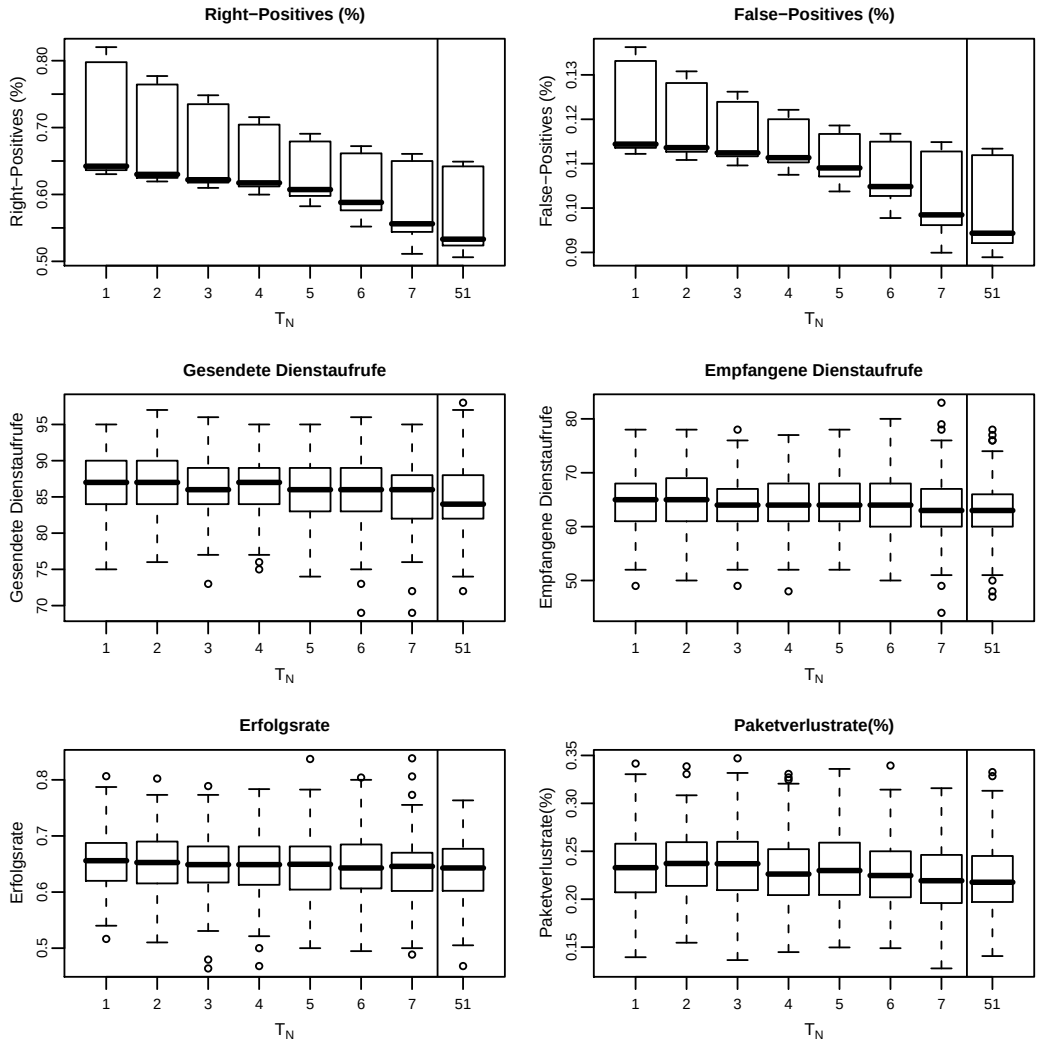

Abbildung A.5: Boxplots: Adaptive Verteilung Teil II (2)

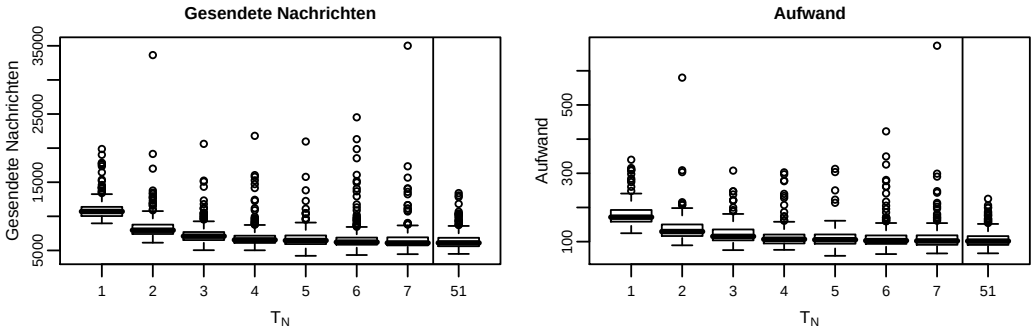

Abbildung A.6: Boxplots: Adaptive Verteilung Teil II (3)

Versuch Adaptive Verteilung Teil I + II

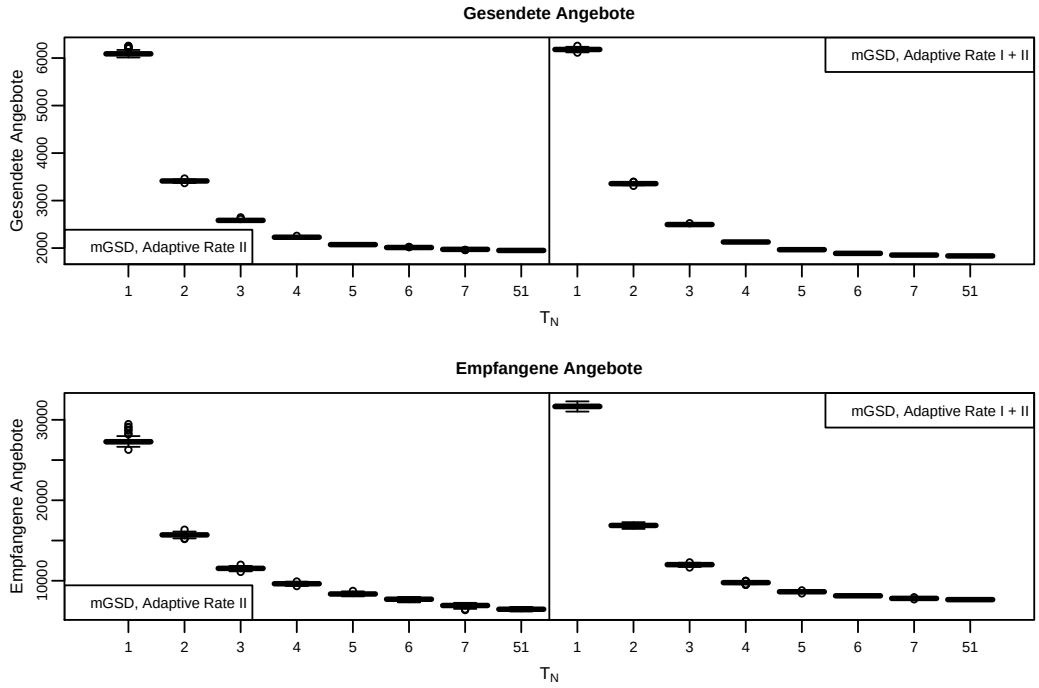

Abbildung A.7: Boxplots: Adaptive Verteilung Teil I + II (1)

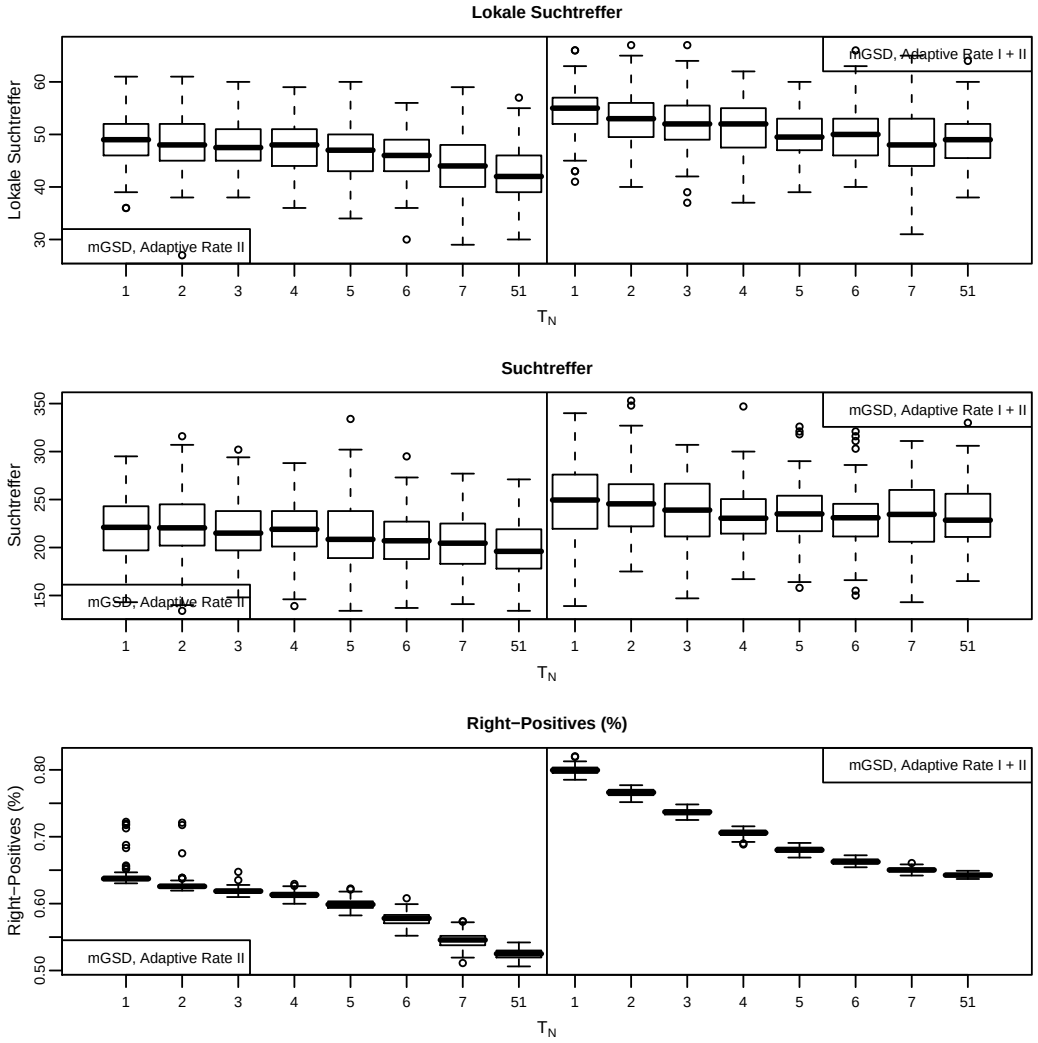

Abbildung A.8: Boxplots: Adaptive Verteilung Teil I + II (2)

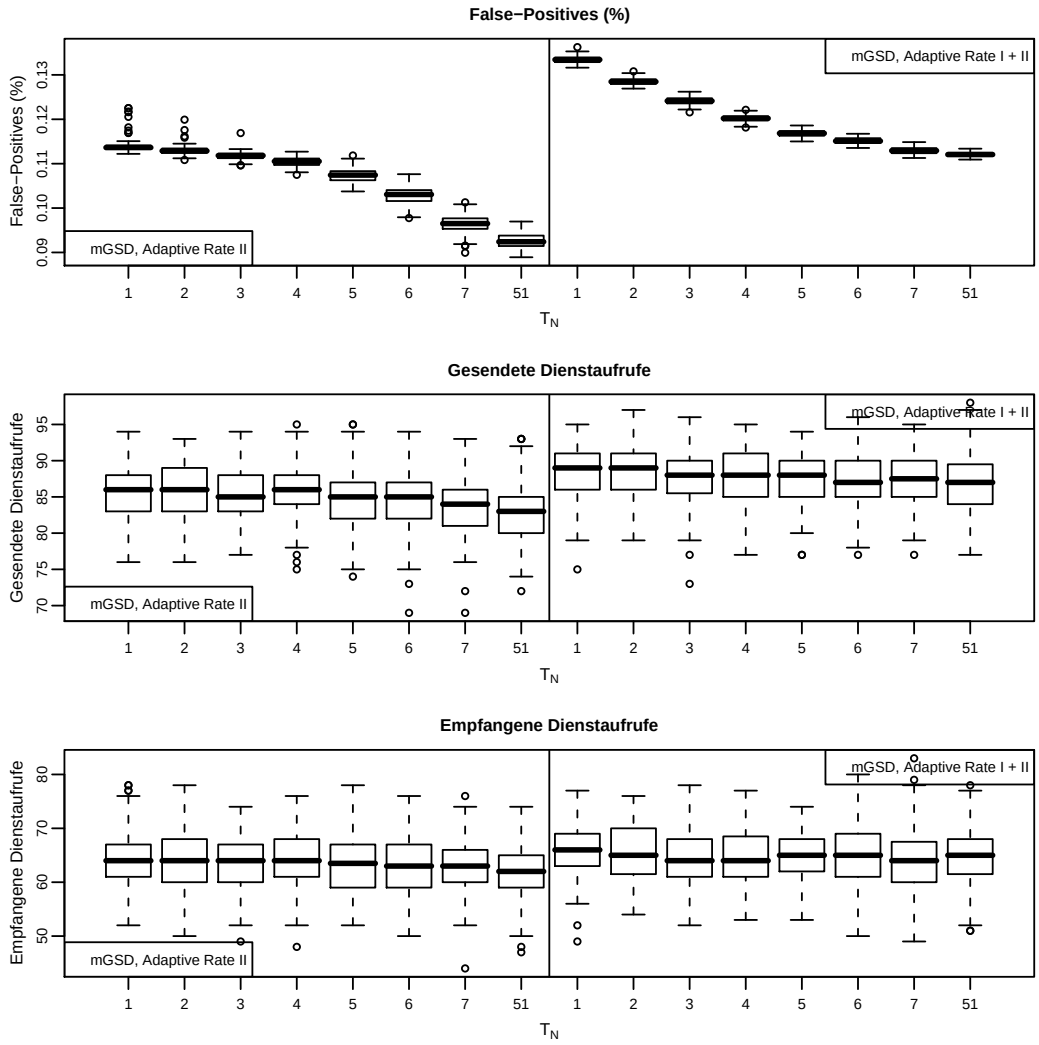

Abbildung A.9: Boxplots: Adaptive Verteilung Teil I + II (3)

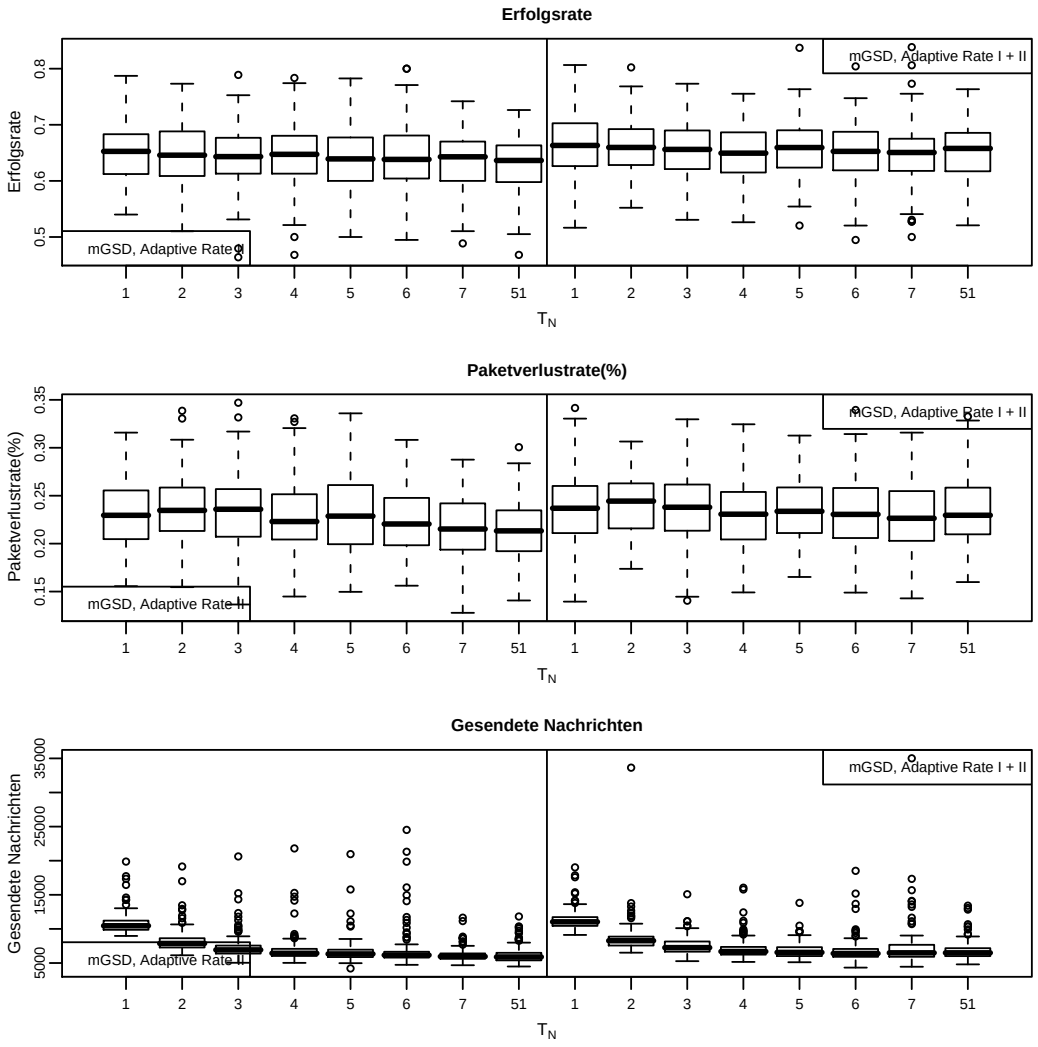

Abbildung A.10: Boxplots: Adaptive Verteilung Teil I + II (4)

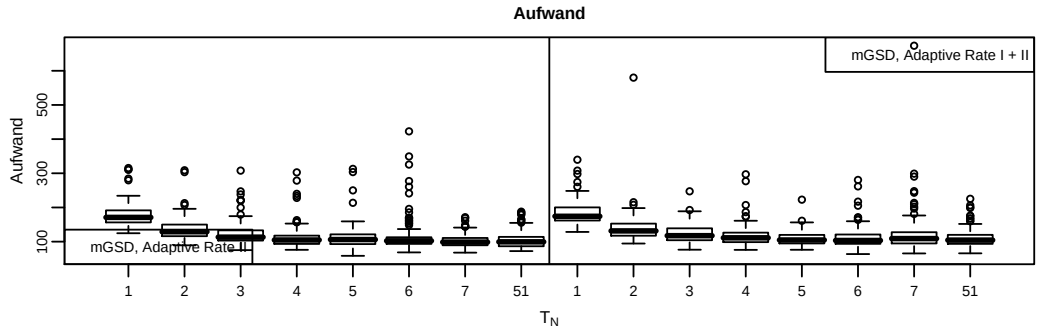

Abbildung A.11: Boxplots: Adaptive Verteilung Teil I + II (5)

Versuch Adaptive Lebenszeit Teil I

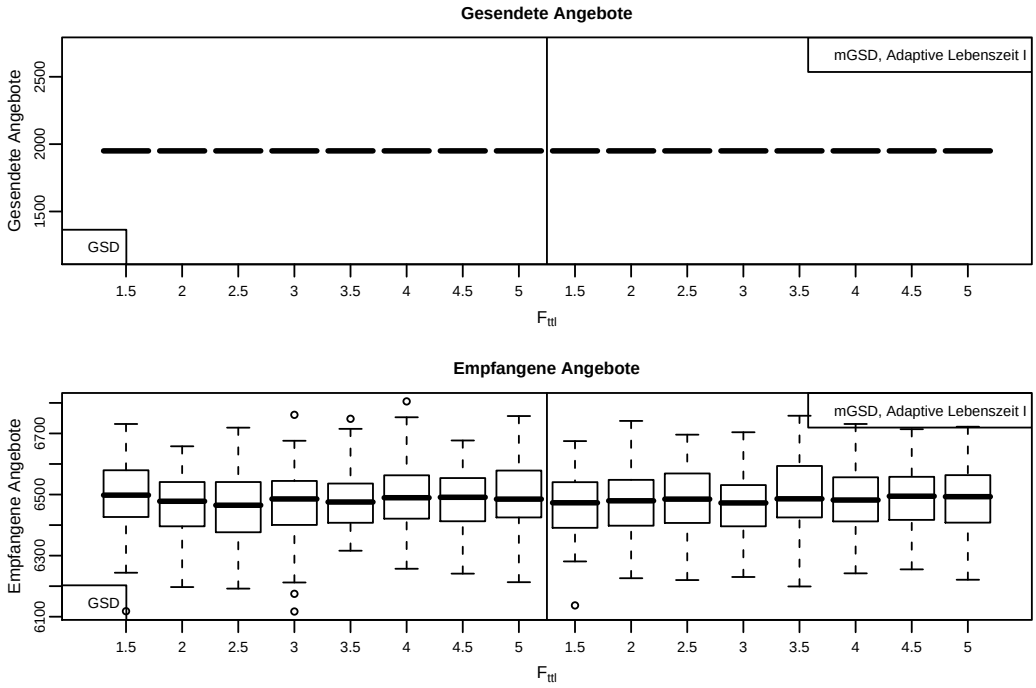

Abbildung A.12: Boxplots: Adaptive Lebenzeit Teil I (1)

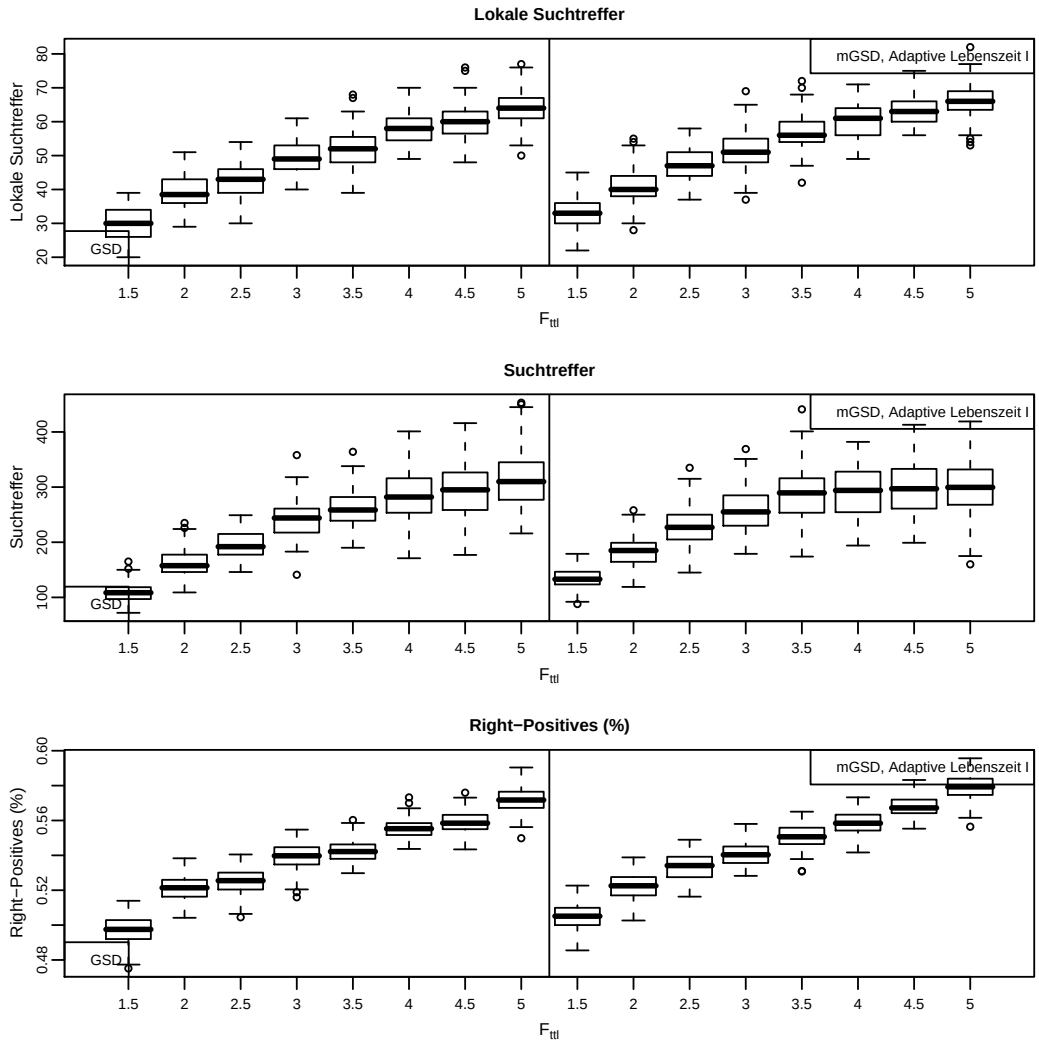

Abbildung A.13: Boxplots: Adaptive Lebenzeit Teil I (2)

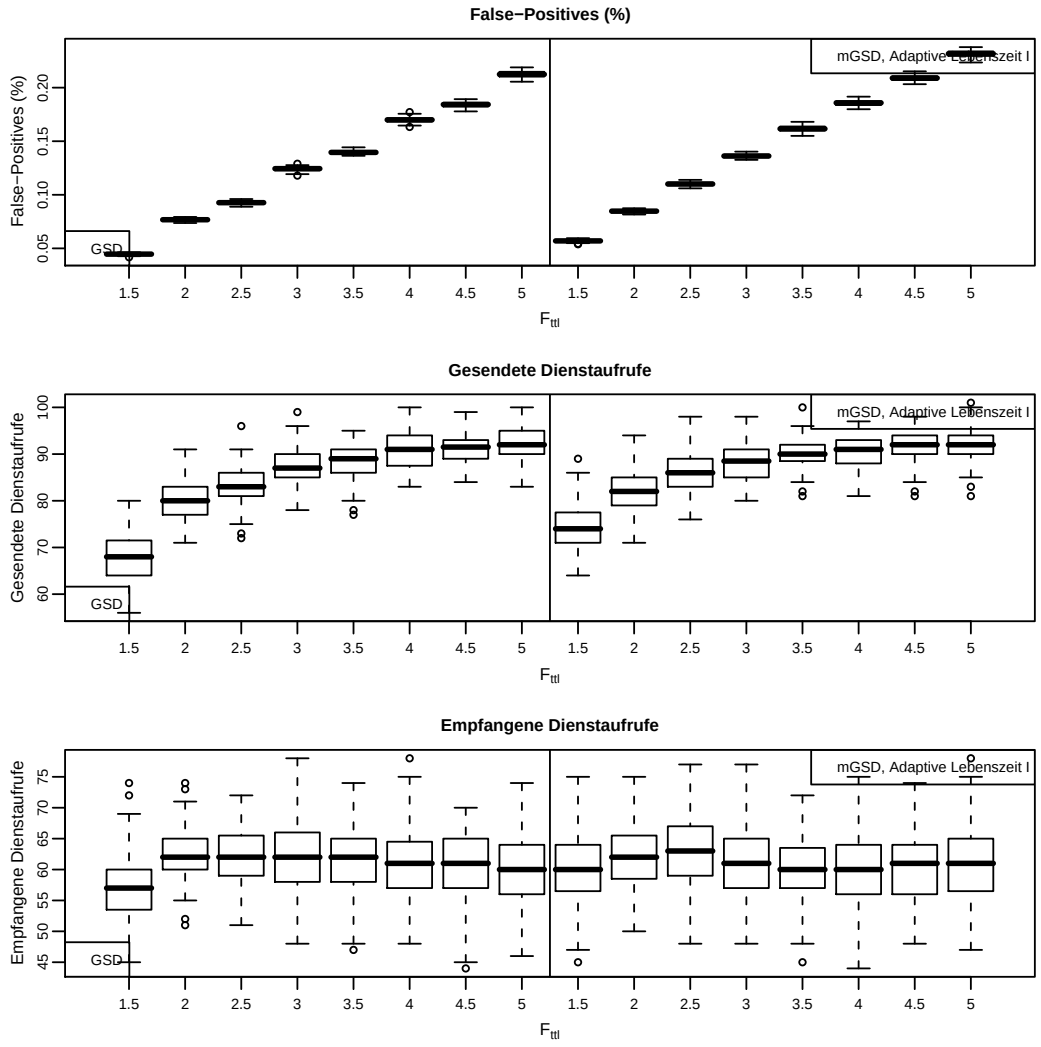

Abbildung A.14: Boxplots: Adaptive Lebenzeit Teil I (3)

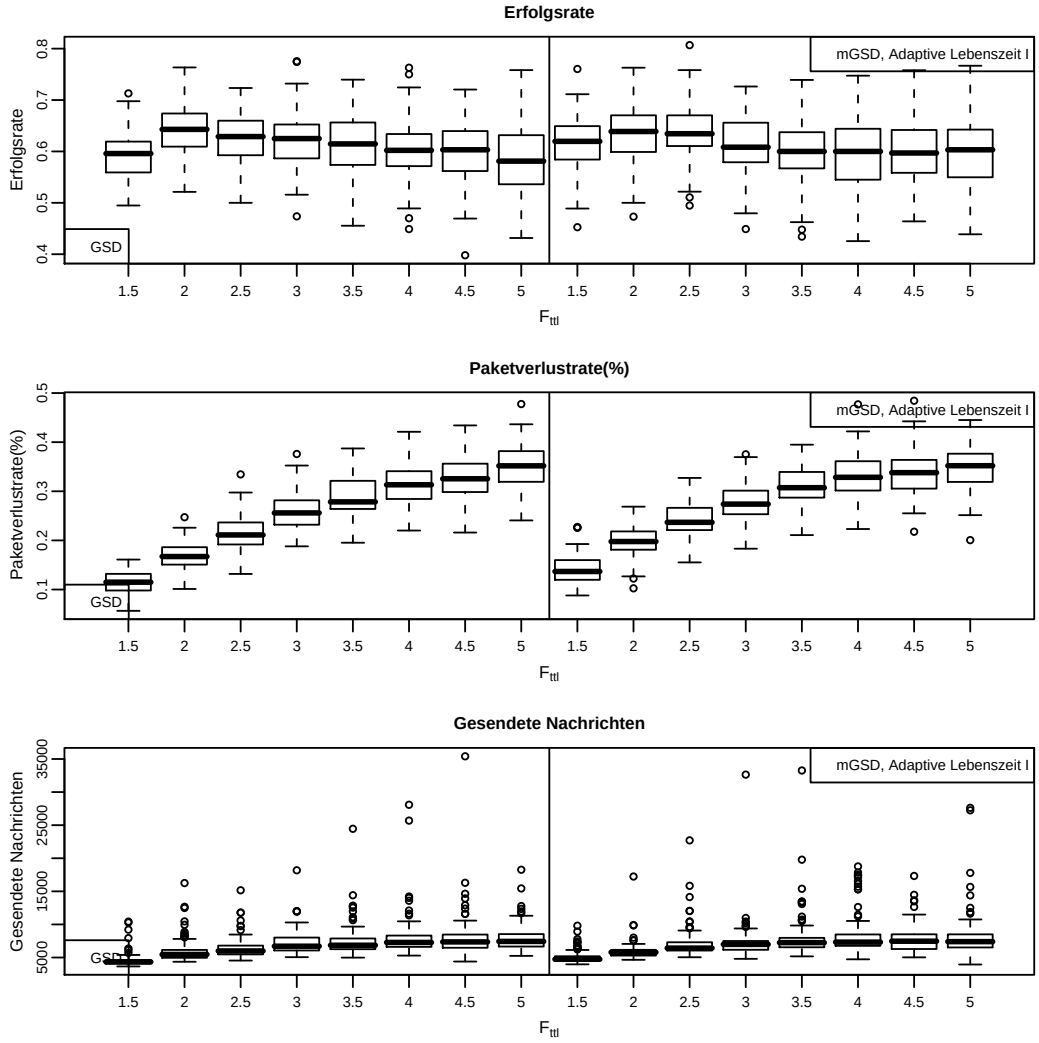

Abbildung A.15: Boxplots: Adaptive Lebenzeit Teil I (4)

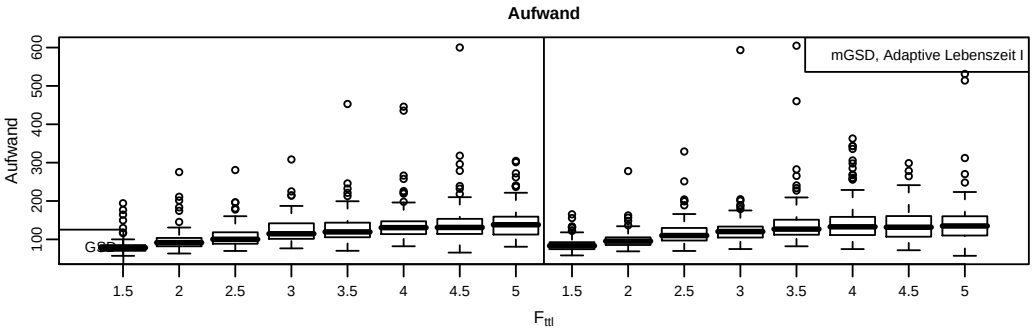

Abbildung A.16: Boxplots: Adaptive Lebenzeit Teil I (5)

Versuch Adaptive Lebenszeit Teil II

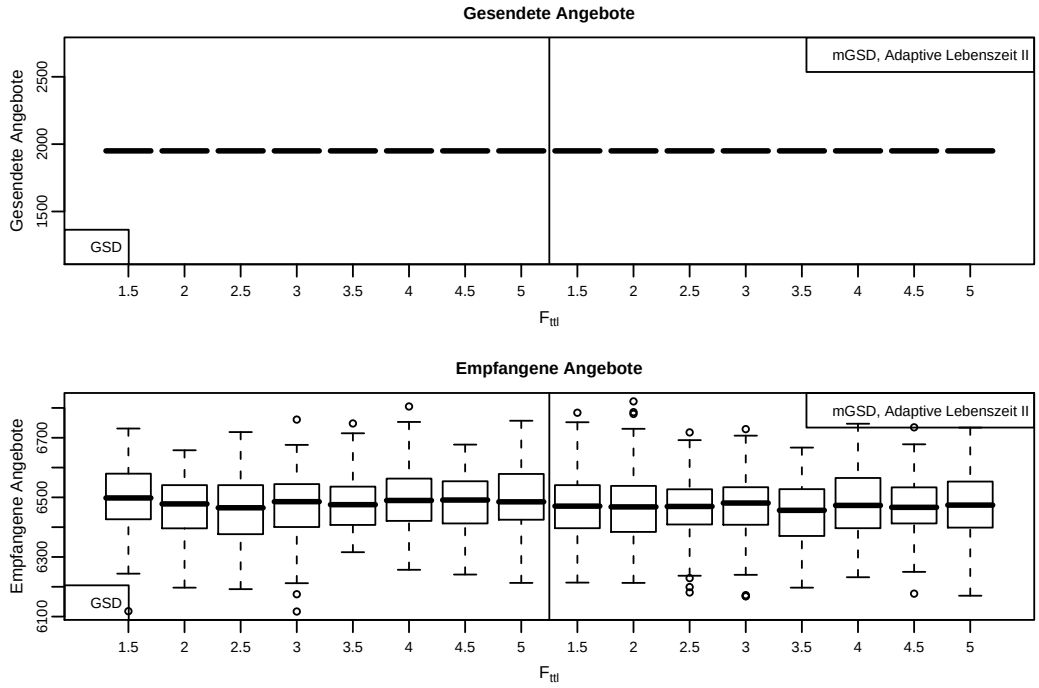

Abbildung A.17: Boxplots: Adaptive Lebenzeit Teil II (1)

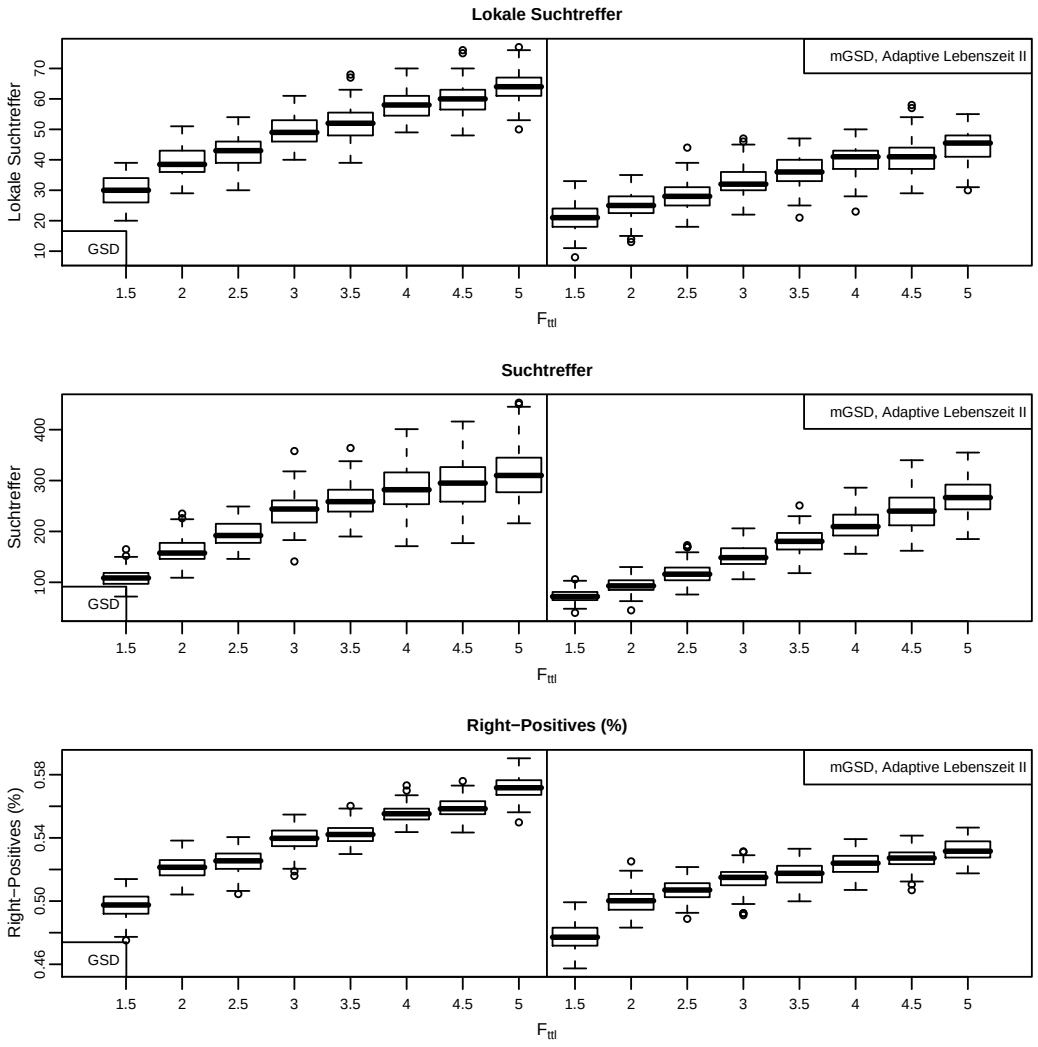

Abbildung A.18: Boxplots: Adaptive Lebenzeit Teil II (2)

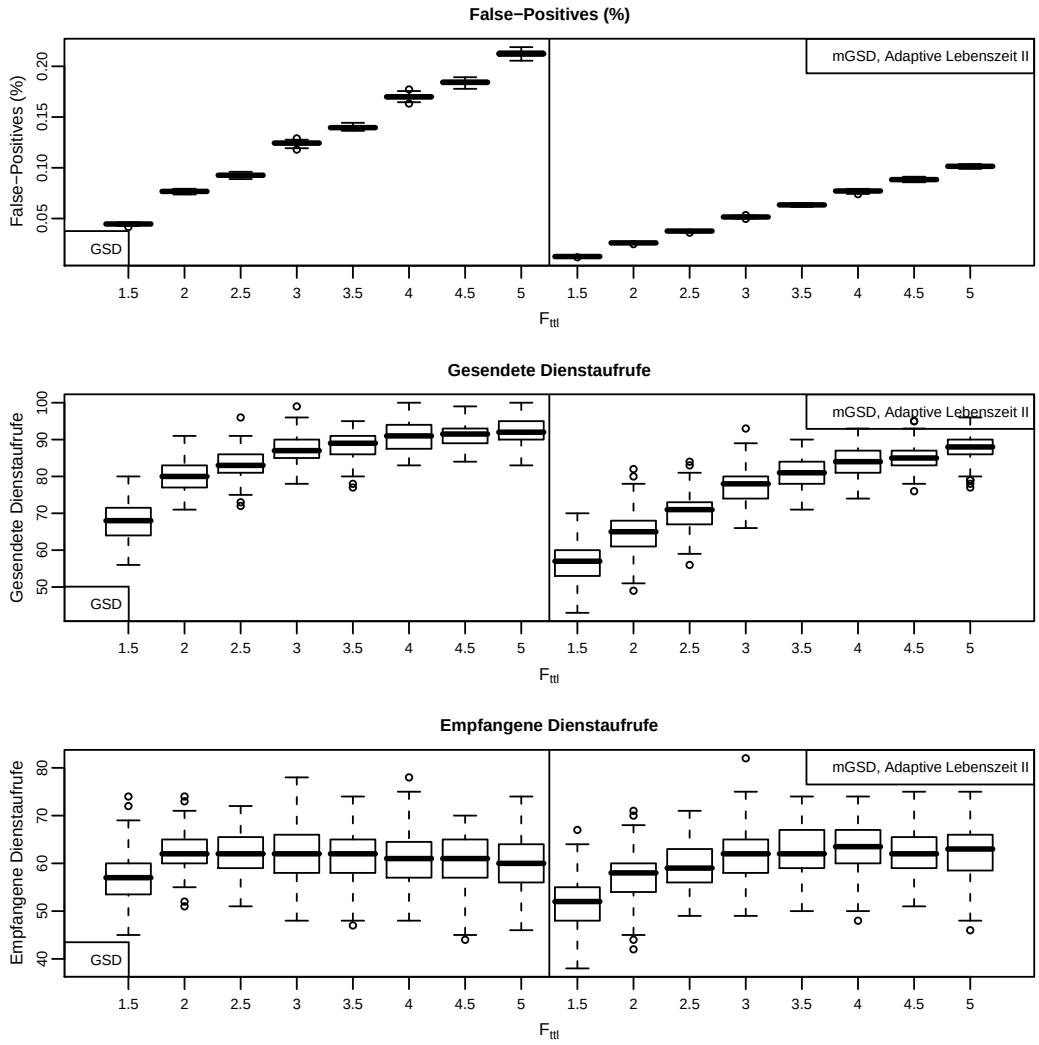

Abbildung A.19: Boxplots: Adaptive Lebenzeit Teil II (3)

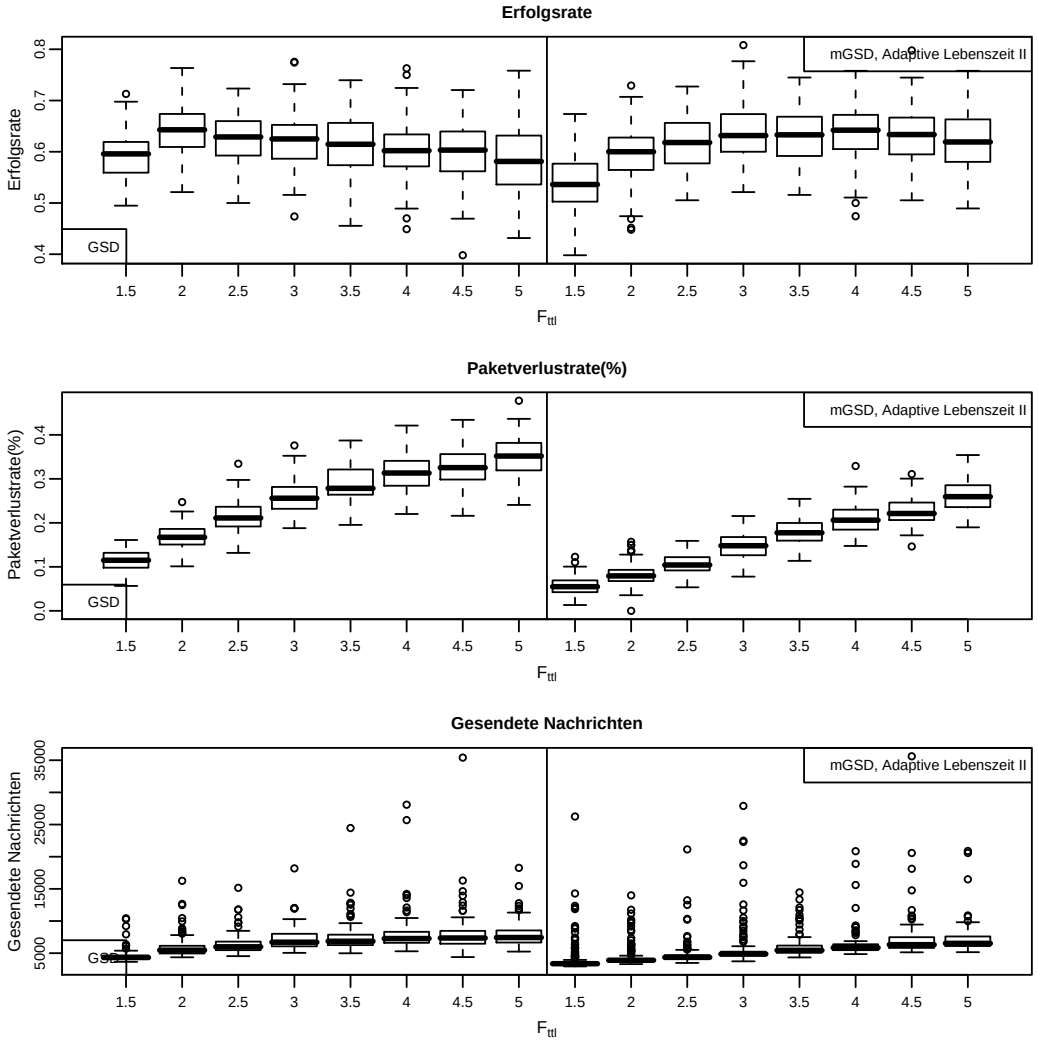

Abbildung A.20: Boxplots: Adaptive Lebenzeit Teil II (4)

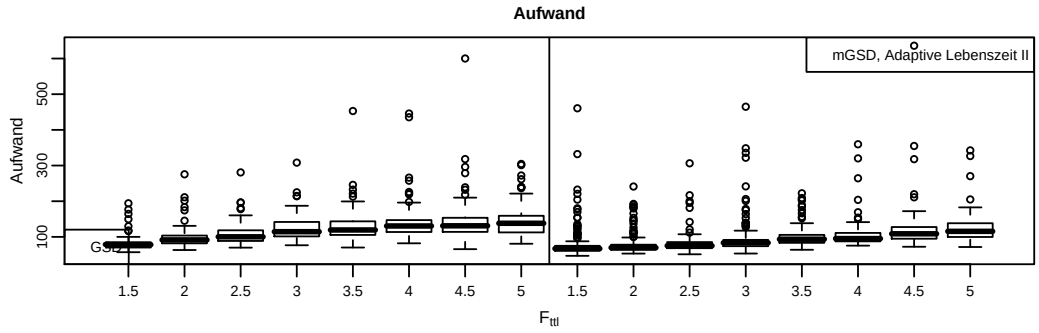

Abbildung A.21: Boxplots: Adaptive Lebenzeit Teil II (5)

Versuch Adaptive Lebenszeit Teil I + II

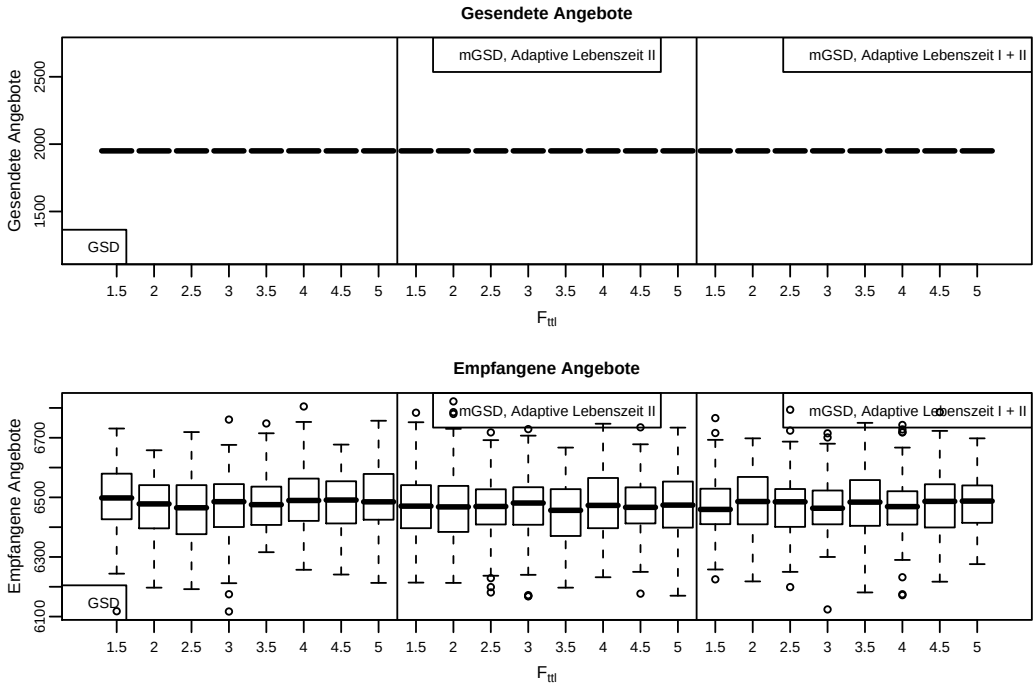

Abbildung A.22: Boxplots: Adaptive Lebenzeit Teil I + II (1)

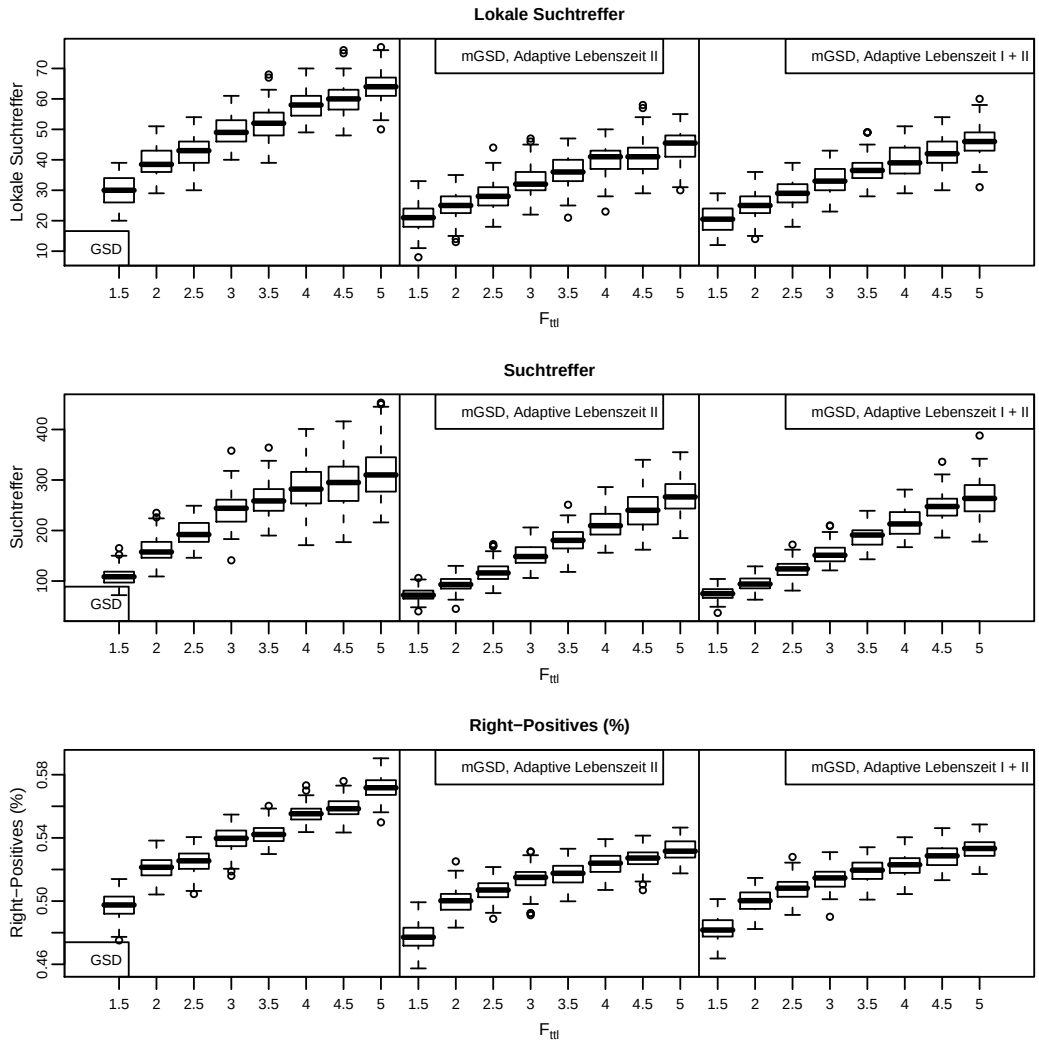

Abbildung A.23: Boxplots: Adaptive Lebenzeit Teil I + II (2)

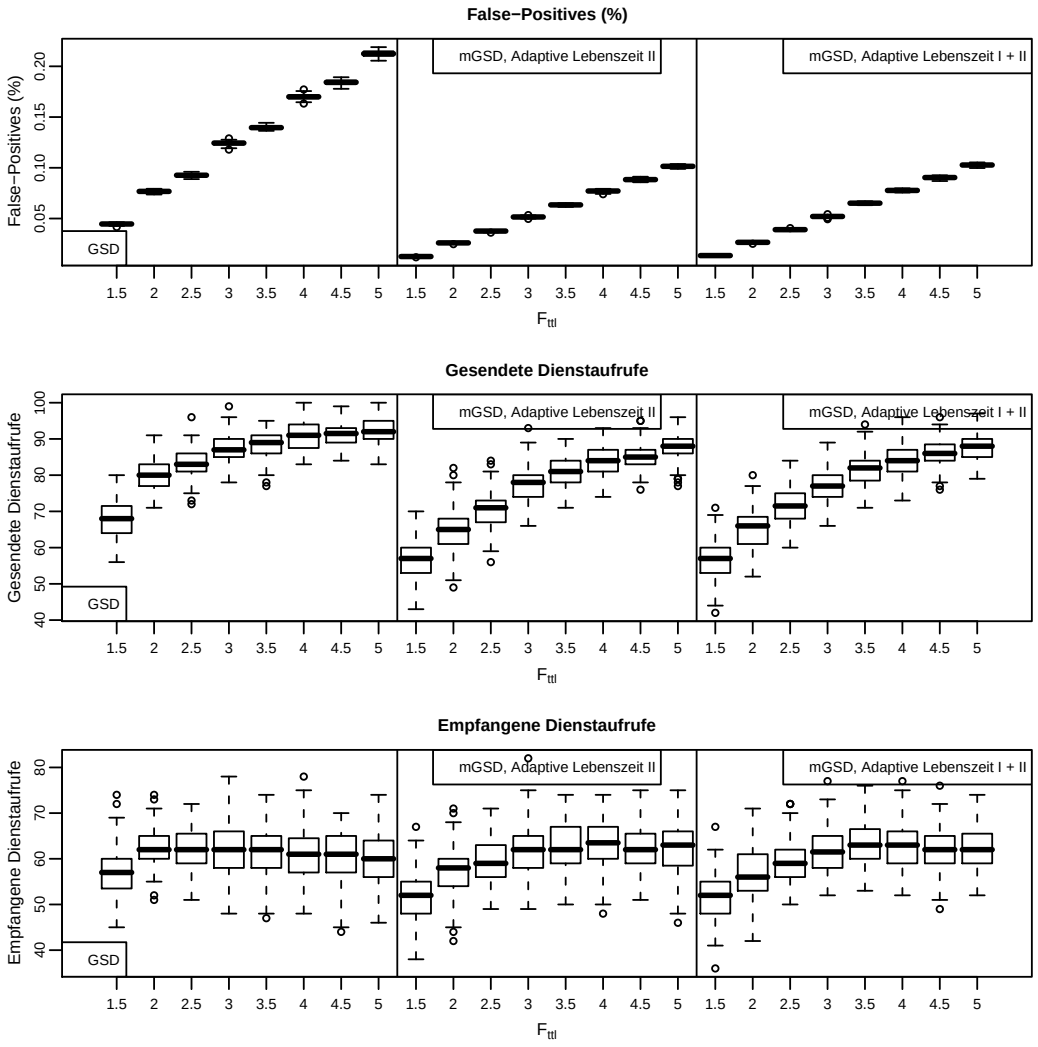

Abbildung A.24: Boxplots: Adaptive Lebenzeit Teil I + II (3)

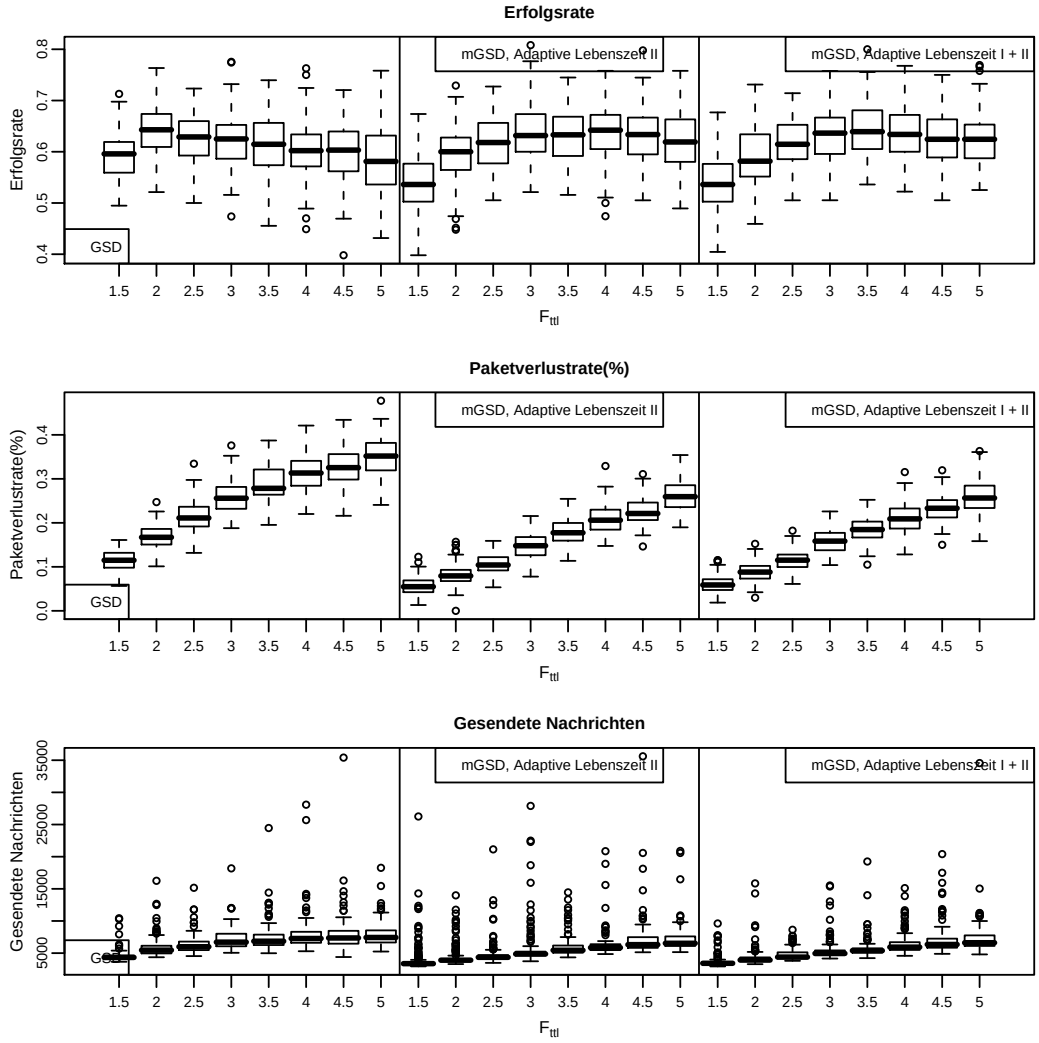

Abbildung A.25: Boxplots: Adaptive Lebenzeit Teil I + II (4)

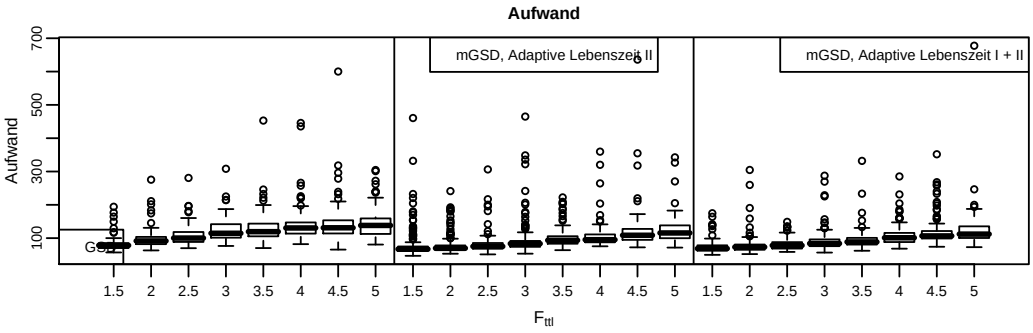

Abbildung A.26: Boxplots: Adaptive Lebenzeit Teil I + II (5)

1. Kombinationsversuch

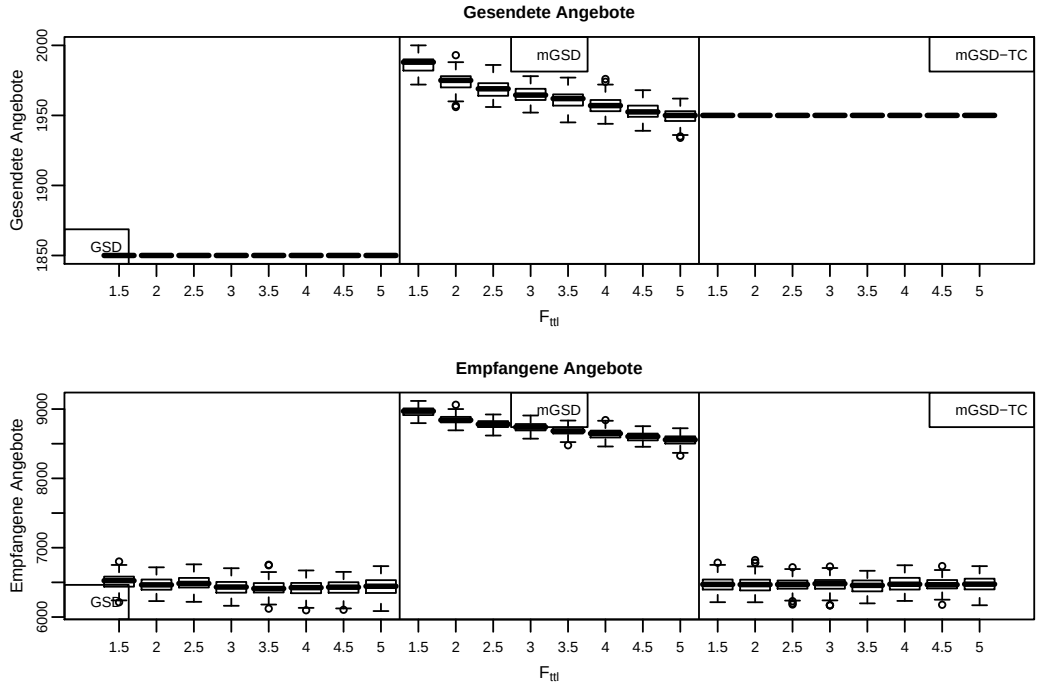

Abbildung A.27: Boxplots: 1. Kombinationsversuch (1)

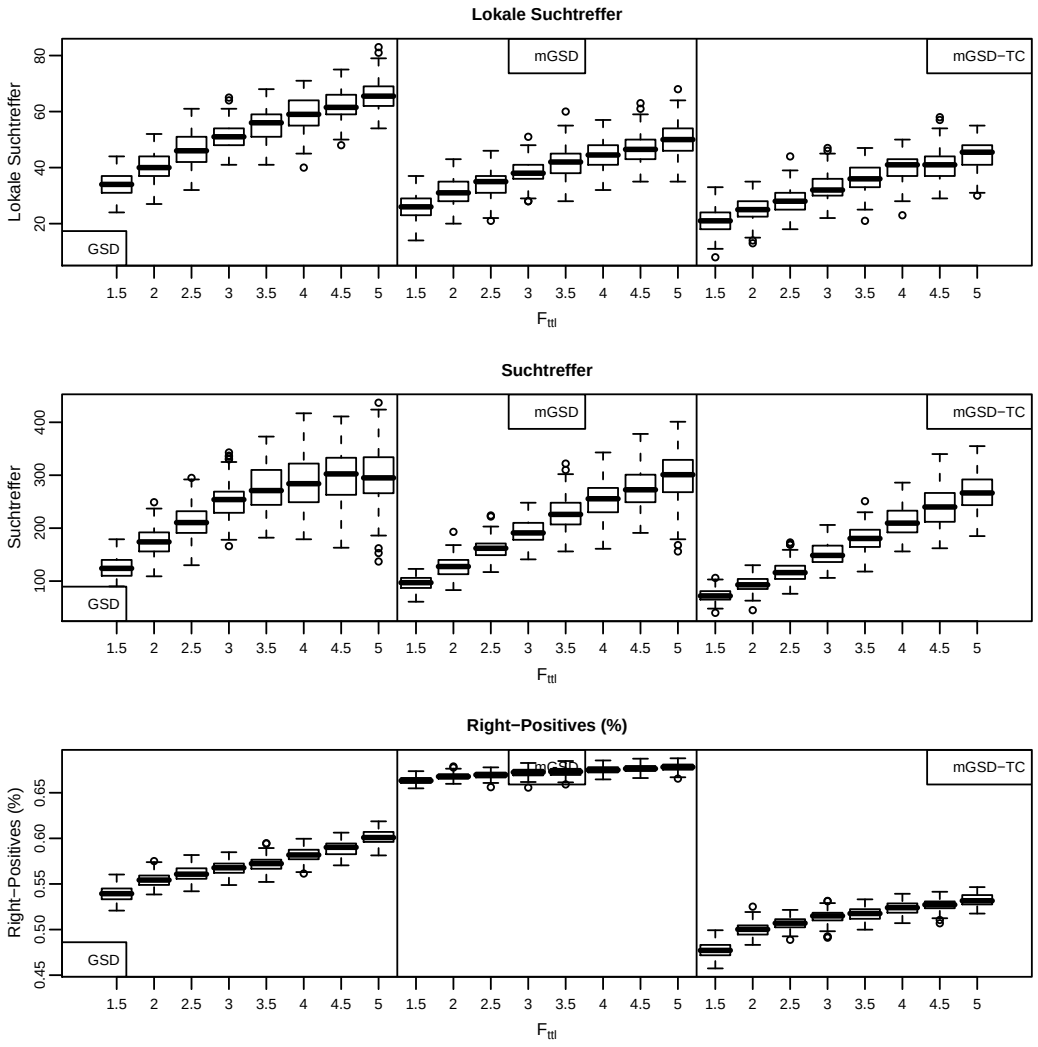

Abbildung A.28: Boxplots: 1. Kombinationsversuch (2)

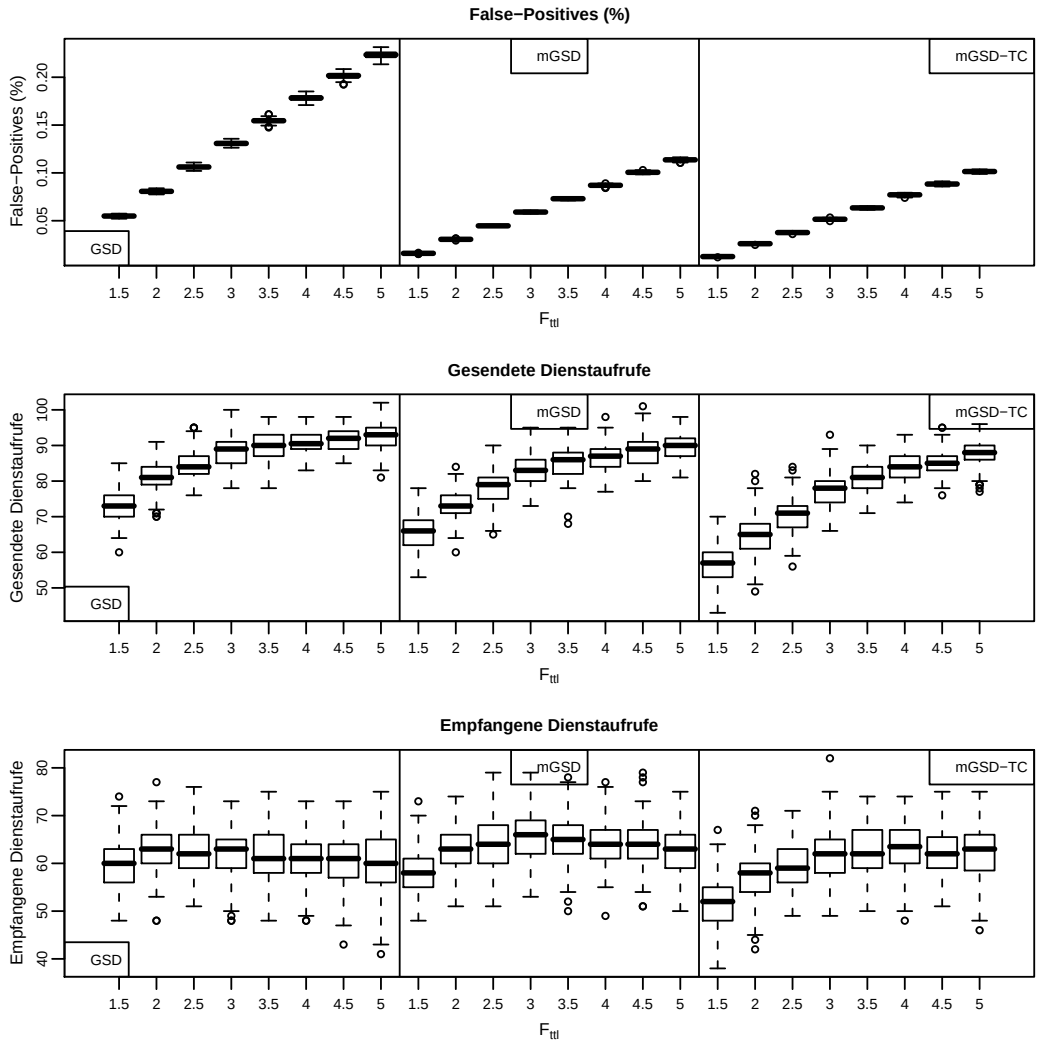

Abbildung A.29: Boxplots: 1. Kombinationsversuch (3)

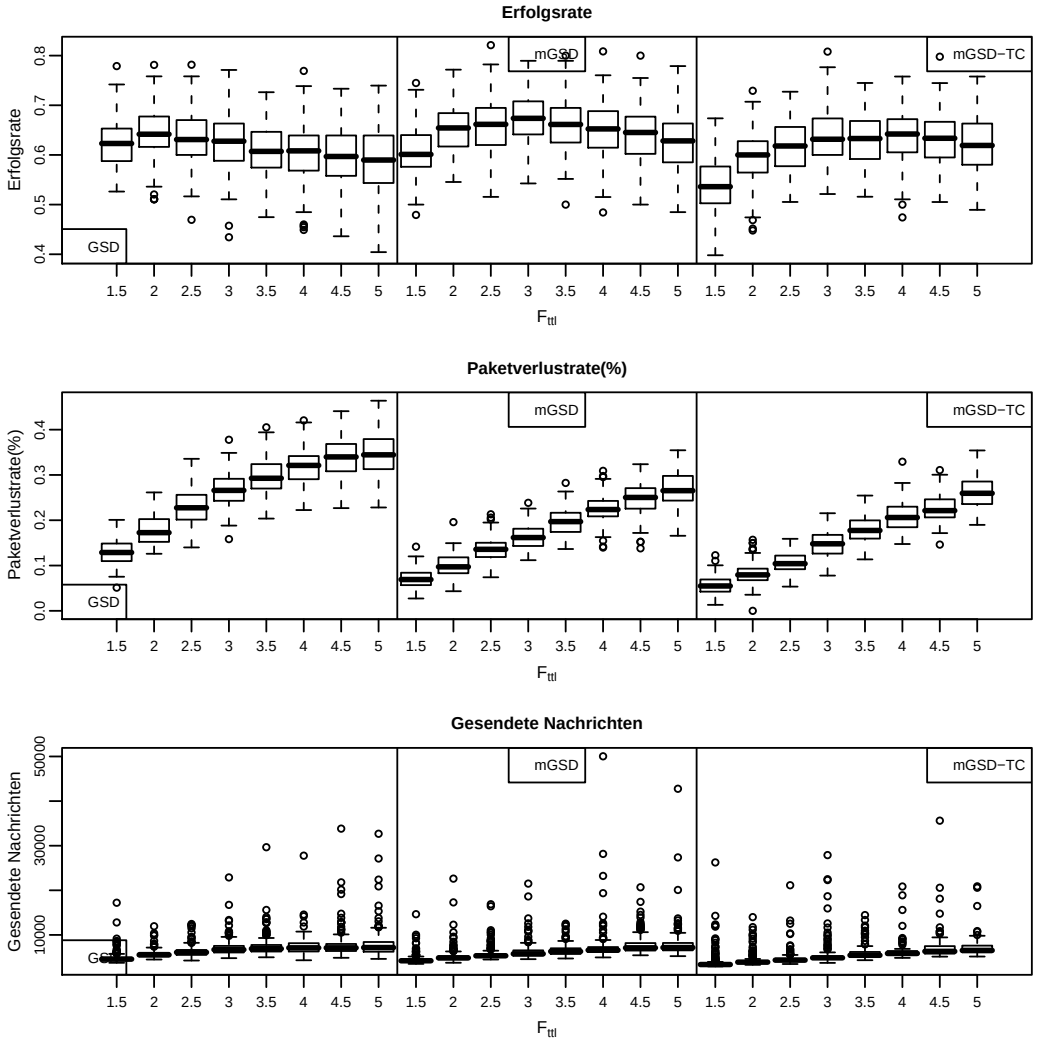

Abbildung A.30: Boxplots: 1. Kombinationsversuch (4)

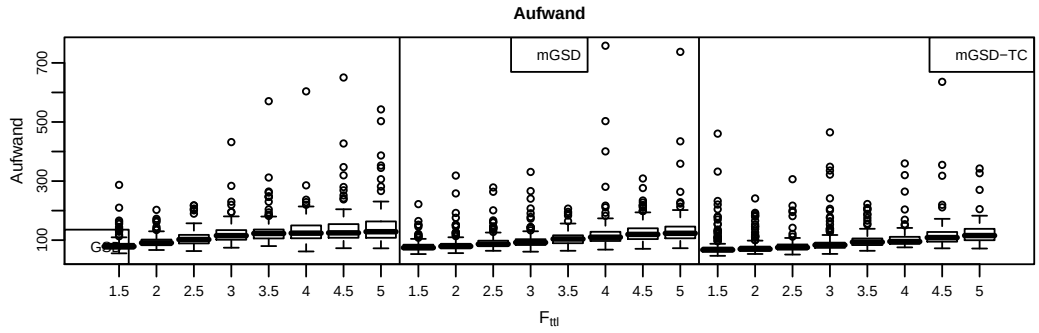

Abbildung A.31: Boxplots: 1. Kombinationsversuch (5)

2. Kombinationsversuch

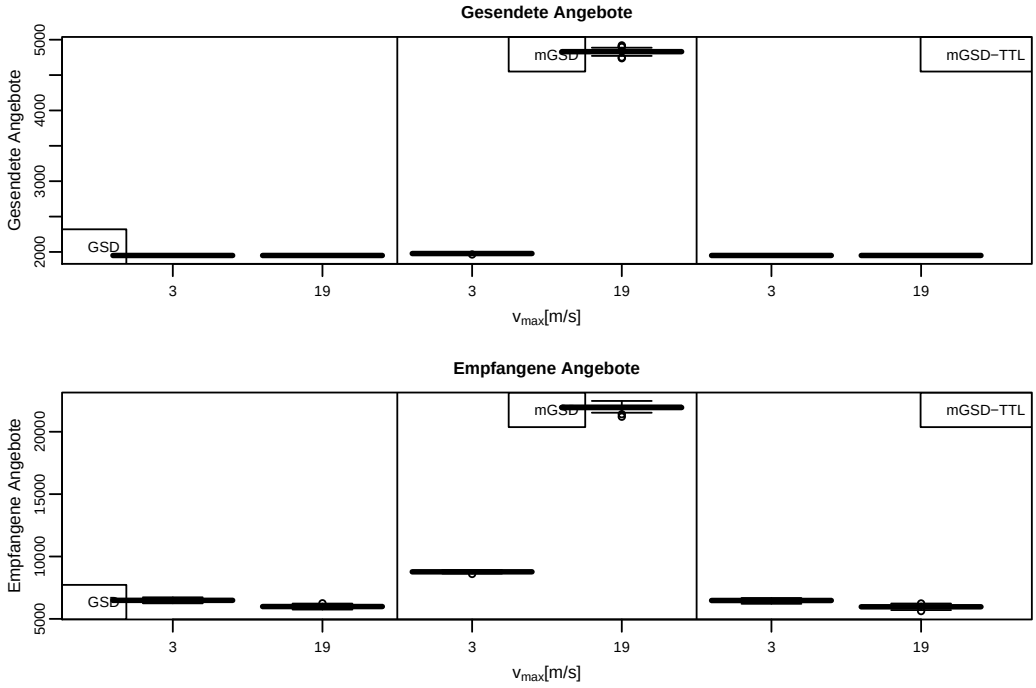

Abbildung A.32: Boxplots: 2. Kombinationsversuch (1)

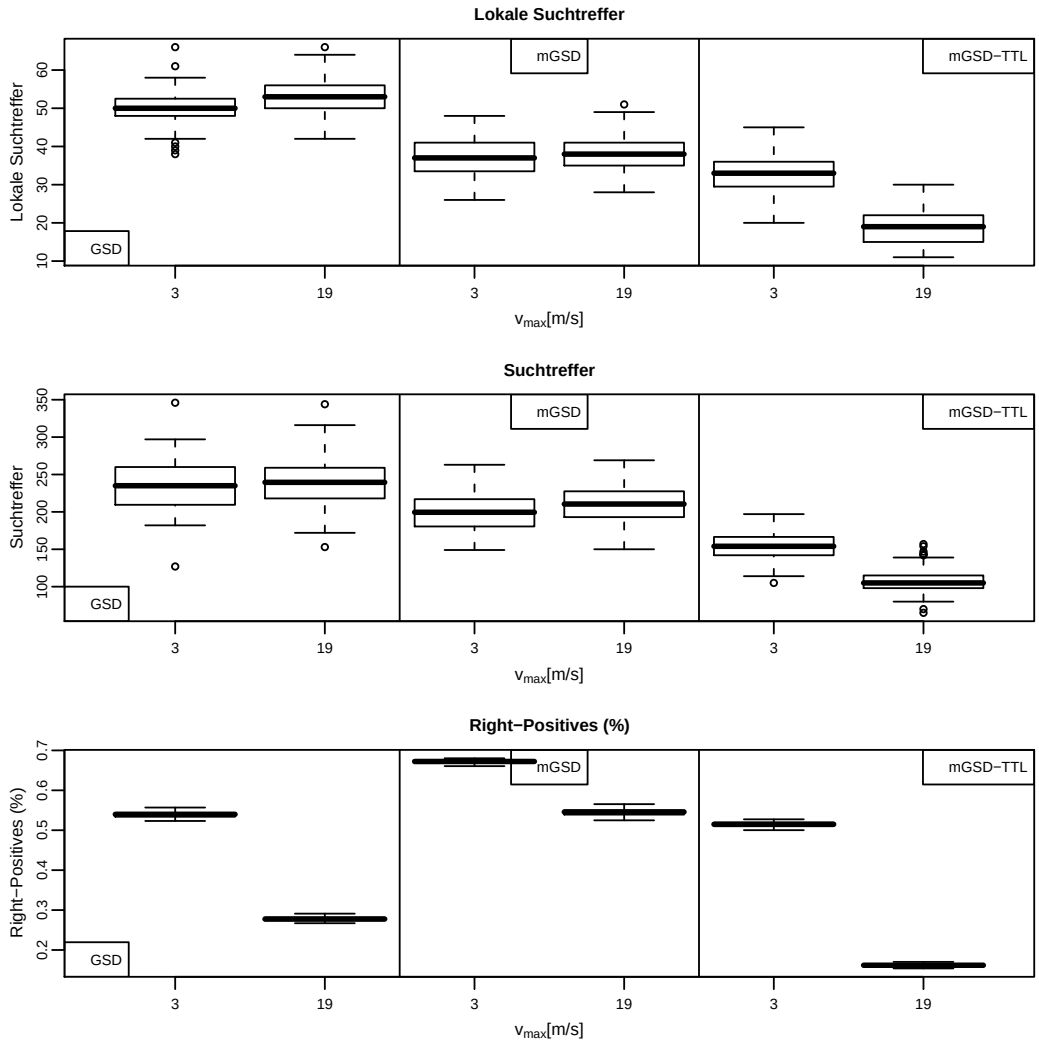

Abbildung A.33: Boxplots: 2. Kombinationsversuch (2)

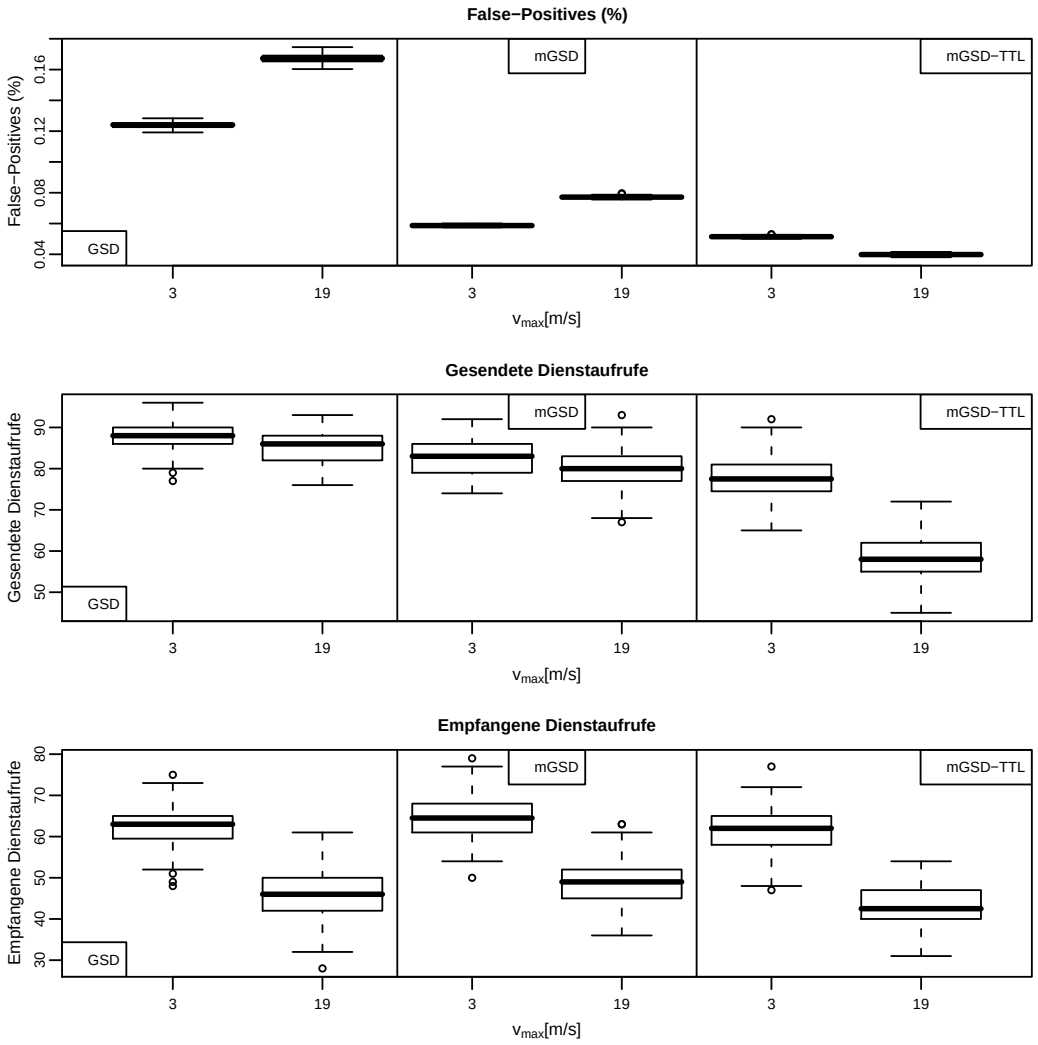

Abbildung A.34: Boxplots: 2. Kombinationsversuch (3)

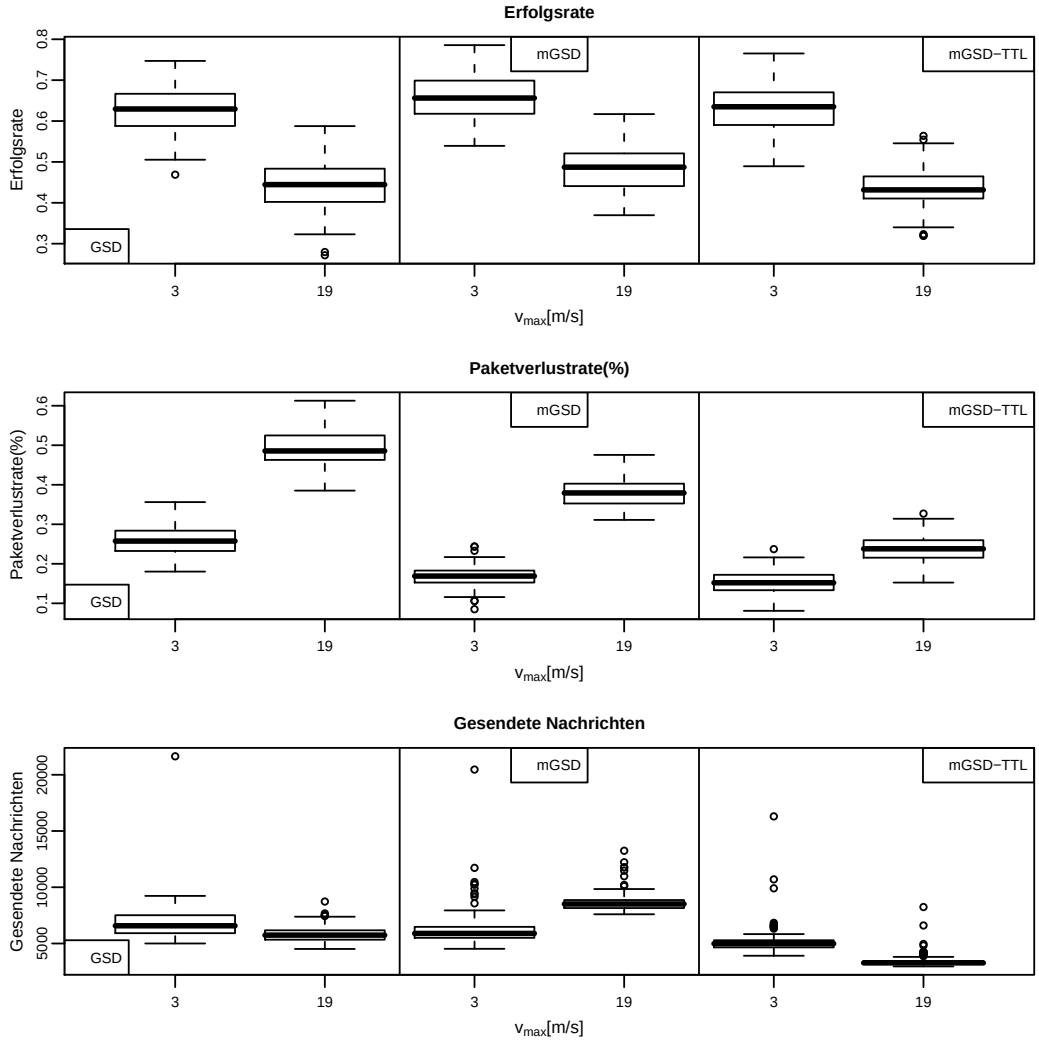

Abbildung A.35: Boxplots: 2. Kombinationsversuch (4)

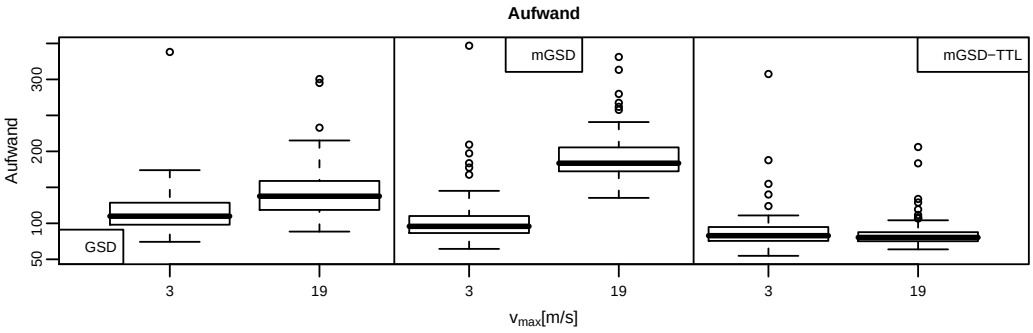

Abbildung A.36: Boxplots: 2. Kombinationsversuch (5)