TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Elektrotechnik und Informationstechnik

Institut für Nachrichtentechnik

Actual Topics Projekt

zum Thema

Simulation des Helfer-Knotens im drahtlosen Maschennetz

vorgelegt von Xiang, Zuo
Matrikel-Nr 4117235
im Studiengang Informationstechnik, Jg. 2014

Betreuer: Dipl.-Inf. Frank Wilhelm
Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. Dr. h.c. Frank H.P. Fitzek
Tag der Einreichung: 24. Januar 2016

1 Einleitung

Wegen der Eigenschaft vom drahtlosen Maschennetz(auf Englisch Wireless Mesh Network - WMN) könnten Knoten in der Nähe vom Sender(als Helfer-Knoten betrachtet) Übertragungen der Hauptroute mithören. In Anbetracht dessen, dass drahtlose Kanäle normalerweise unter signifikanten Paketverlusten leiden, könnte dieses Mithören des Helfer-Knotens eine potentielle Möglichkeit dafür erzeugen, dass sie die Gesamtleistung des Netzwerks durch opportunistische Sendungen der empfangenen Paketen verbessern.

Andererseits wegen der Konkurrenz des Zugriffs auf drahtlose Kanäle dürfen Sender und Helfer nicht zur gleichen Zeit Pakete transportieren. Deswegen könnte der Helfer nur die Sendezeit des Senders besetzen, wenn er keine neue Informationen dem Empfänger anbietet. [1]

Deshalb könnte die Untersuchung nach unterschiedlichen Sendestrategien des Helfers ein wichtiges Thema sein. Im Rahmen dieser Arbeit sollte ein Simulationsprogramm mit Kodo-Python aufgebaut werden, damit zwei unterschiedliche Strategien des Helfers verglichen werden könnten. Nämlich hier zuerst ohne Helfer und dann mit dem Helfer-Knoten, das bei der Übertragung den PlayNCool-Algorithmus verwendet.

Dieses Protokoll wird in fünf Teilen gegliedert. Im Kapitel 2 werden zuerst das Szenario, Arbeit sowie Werkzeuge der Simulation genauer vorgestellt. Danach werden Simulationsergebnisse unterschiedlicher Parameter im Kapitel 3 durch Grafiken veranschaulicht und analysiert. Anschießend werden im Kapitel 4 eine Zusammenfassung und einige Ausblicke in Bezug auf Weiterentwicklungen dieser Arbeit sowie Verbesserungen der Simulation geliefert. Am Ende stehen einige Tabelle und zusätzliche Abbildungen der Simulationsergebnisse im Anhang zur Verfügung.

2 Methode

2.1 Aufgabenbeschreibung

2.1.1 Szenario der Simulation

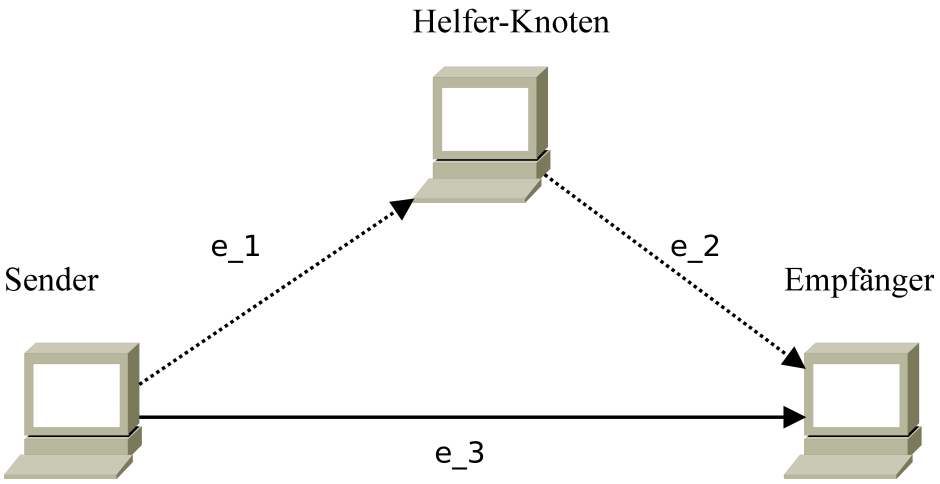


Abbildung 2.1: Szenario der Simulation

Weil in der Abbildung 2.1 beschriebenes Szenario der häufigste Fall in jeder Verbindung des drahtlos vermaschten Netzwerks ist, sollte es in dieser Aufgabe detailliert analysiert und simuliert.

Die Paketverlustraten zwischen Sender, Helfer-Knoten sowie Empfänger werden jeweils durch e_1, e_2 sowie e_3 charakterisiert. Um die Analysen sowie Simulationen zu vereinfachen, wird es in dieser Aufgabe angenommen, dass Verbindungen des Helfer-Knotens symmetrisch sind($e_1 = e_2$).

2.1.2 Simulationsarbeit

Wie in der Einleitung erwähnt, könnte die heuristische Weitersendung des Helfer-Knotens zur Verbesserung der Netzwerkleistung beitragen. Dehalb werden zwei Situationen in dieser Aufgabe mit Benutzung vom Kodo-Python simuliert und auch verglichen.

(1) Datenübertragung mit RLNC ohne Helfer-Knoten

(2) Datenübertragung mit RLNC mit Helfer-Knoten

Bei der zweiten Situation gibt es viele unterschiedliche Strategien dafür, wann und wie der Helfer Pakete weiter senden sollte. Und in dieser Aufgabe wird der PlayNCool Algorithmus[1] implementiert und simuliert.

Weil die zufällige Netzwerkkodierung bei den Simulationen verwendet wird, können Sender, Helfer-Knoten und Empfänger auch als Encoder, Recoder sowie Decoder genannt werden.

Wenn das Helfer-Knoten Pakete zu senden beginnt, liegen auch verschiedene Varianten für die Disposition zwischen Sender und Helfer vor. In dieser Simulation wird eine vereinfachte Annäherung von TDMA(Time Division Multiple Access) verwendet, nämlich Sender oder Helfer besitzt das Kanal alternativ.

Bei jeder Simulation werden Paketverlustraten(nämlich e_1, e_2, e_3) als Variablen eingestellt. Um einen Vergleich durchzuführen, wird die Anzahl von Paketen bekommen, die insgesamt und auch nur von dem Helfer gesendet werden. Danach werden Daten mehrerer Simulationen verarbeitet und durch Grafiken veranschaulicht. In dieser Arbeit wird der Test für bestimmten Parameter jeweils 50 mal durchgeführt und die Student-Verteilung bei der Schätzung verwendet.

2.2 Algorithmus

2.2.1 RLNC

RLNC(Random Linear Networkcoding) ist ein einfaches aber leistungsfähiges Kodierungsschema für Netzwerkkodierung. Zusammenfassend senden Knoten im Netzwerk zufällige lineare Kombinationen von empfangenen Paketen, um z.B. den Durchsatz und Sicherheiten zu erhöhen. Laut der Forschung bietet RLNC wegen der Fähigkeit von Rekodierung große Vorteile bei Multi-Hop-Netzwerks an. [2]

2.2.2 PlayNCool

PlayNCool ist ein opportunistisches Protokoll basiert auf Netzwerkkodierung für lokale Optimierung vom Rounting im drahtlos Maschennetz. [1]

Ein wichtiges Leistungsmerkmal dieses Algorithmus ist, dass das Helfer-Knoten nicht sofort sondern nach gewisser Zeit Pakete transportieren wird,

um mehr nützliche Informationen dem Empfänger anzubieten. Wegen der Vermeidung der vergeblichen Konkurrenz um das drahtlose Kanal in der Anfangsphase, sollte die Leistung des Netzwerks deutlich verbessert werden.

Für die Erzeugung der kodierten Pakte mit der Generationsgröße g wird dabei RLNC verwendet. Die Anzahl der Pakete am Helfer-Knoten, die vom Sender transportiert werden, wird als r definiert. Dieser Wert sollte als der Schwellenwert für den Anfang der Paketübertragung zwischen Helfer und Empfänger benutzt werden.

Laut der PlayNCool-Politik wird r durch folgende Formel berechnet. [1]

$$r = -g \cdot \frac{C}{D - (1 - e_3) \cdot C} \tag{2.1}$$

mit

$$C = -1 + e_2 + e_3 - e_1 \cdot e_3 \tag{2.2}$$

$$D = (2 - e_3 - e_2) \cdot (e_3 - e_1 \cdot e_3) \tag{2.3}$$

2.3 Vorstellung der Simulationswerkzeuge

Die Simulationscodes sowie Dokumentationen stehen im Repository auf Github(`https://github.com/stevelorenz/acto15-helper`) zur Verfügung.

2.3.1 Kodo

Kodo ist eine kommerzielle Bibliothek für Netzwerkkodierung, die entwickelt vom Steinwurf wird. Kodo ermöglicht die Verwendung von RLNC mit hoher Leistung in einer Vielzahl von Produkten und Applikationen. Kodo ist mit C++ geschrieben und stellt viele Anbindungen für andere Programmiersprachen zur Verfügung, wie Python, Java, Javascript usw. Mehr Informationen sind auf der Webseite(`http://steinwurf.com/kodo/`) verfügbar[3]

In diesen Simulationen wird binäre RLNC-Kodierung mit Hilfe vom Kodo-Python implementiert. Ausführliche Informationen vom Kodo-Python stehen auf der Website(`https://github.com/steinwurf/kodo-python`) zur Verfügung. [4]

2.3.2 Matplotlib

Für die Erstellung der Grafiken wird eine Python-Bibliothek: Matplotlib verwendet. Damit können schöne graphische Darstellungen von Daten schnell

erstellt werden. Eine Vielzahl von Beispielen steht auf der Galerie zur Verfügung. In dieser Arbeit wird z.B Errorbar für Grafiken mit Konfidenzintervallen verwendet.(`http://matplotlib.org/1.2.1/examples/pylab_examples/errorbar_demo.html`) [(5)]

3 Ergebnisse

3.1 Simulationsparameter

In Simulationen könnten $e_2(e_1 = e_2)$ und auch e_3 als Variablen betrachtet werden. Um einen Vergleich zu machen, nimmt e_2 jeweils einen bestimmten Wert(hier 30%, 60% und 90%) und variiert e_3 von 0% bis 95% mit dem Abstand von 5%.

Als Kriterien für die Durchführung der Übertragung werden hier zwei Kenngrößen betrachtet, nämlich gesamte gesandte Pakte und Pakete, die nur vom Helfer-Knoten gesandt werden. Durch Simulationen des Umfang von 50-Malen werden Mittelwerte, empirische Standardabweichungen bekommen(Tabellen stehen im Anhang zur Verfügung) und bildhaft dargestellt. Dabei werden die Konfidenzintervalle nach der relativen Größe als Errorbar oder vernachlässigt.

Um die Verbesserung durch Helfer anschaulicher zu zeigen, wird auch ein Gewinn dazwischen berechnet und in Grafiken dargestellt. Der Gewinn wird durch folgende Formel definiert.

$$Gewinn = (\frac{A}{B} - 1) \cdot 100\% \tag{3.1}$$

A: Die Anzahl insgesamt gesandter Pakete ohne Helfer

B: Die entsprechende Anzahl mit Helfer

3.2 Simulationsergebnisse

In folgenden Grafiken werden Simulationsergebnisse dargestellt und analysiert.

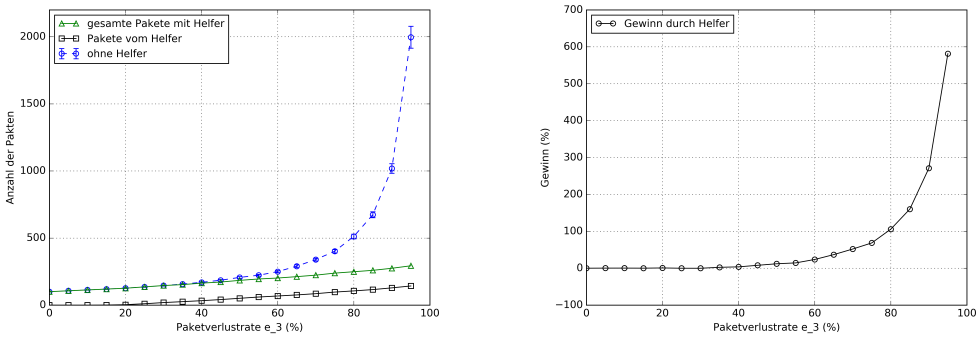
Zuerst wird e_2 auf 30% eingestellt und Ergebnisse werden in der Abbildung 3.1 gezeigt. Entsprechende Daten kann man aus Tabellen A.1 sowie A.2 lesen.

Aus 3.1 (a) zeigt es sich zuerst, dass die Anzahl von Paketen der Situation ohne Helfer eine offensichtliche steigende Tendenz aufweist. Insbesondere nach 85%, wächst der Wert von 673.96 auf 1996.58 über 1018.02 relativ schnell. Im Vergleich dazu nimmt der Mittelwert der Situation mit Helfer mit einer niedrigen und stabilen Geschwindigkeit zu. Beim schlechtesten

Fall($e_3 = 95\%$), müsste der Sender ohne Helfer ungefähr siebenfach so viele Pakte transportieren als mit dem Helfer. Aus die Linie vom Gewinn kann man erfahren, dass der Gewinn nach $e_3 = 60\%$ sich schnell vergrößert, was eine relative gute Verbesserung durch Einstellung des Helfer kennzeichnet.

Wenn es um die empirische Standardabweichung zu kommen, stellen Werte eine gleiche Tendenz wie Mittelwert dar. Nämlich während die Situation ohne Helfer einen deutlichen Wachstum zeigt, bleibt sie mit Helfer beinahe unverändert bei schlechten Fällen. Deshalb werden Konfidenzintervalle zweiter Situation vernachlässigt.

Die Anzahl der gesandten Paketen(schwarze Linie) vom Helfer vergrößert sich mit der ansteigenden Paketverlustrate. Damit könnte es vermutet werden, dass der Helfer immer mehr zur der Übertragung beiträgt. Außerdem wird es in der Abbildung 3.1(a) gezeigt, dass es einen Schnittpunkt(hier ungefähr um 30%) zwischen grünen und blauen Linien(ohne Helfer und gesamte Anzahl mit Helfer) gibt. Danach hat die Übertragung mit dem Helfer immer eine bessere Leistung. Deshalb könnte dieser Wert als einen Schwellenwert betrachtet werden, ob ein Helfer für die Übertragung eingestellt werden sollte.

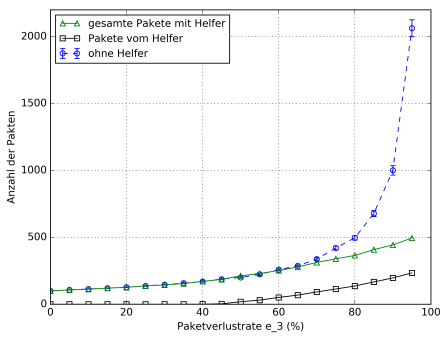


(a) Mittelwerte **(b)** Gewinn

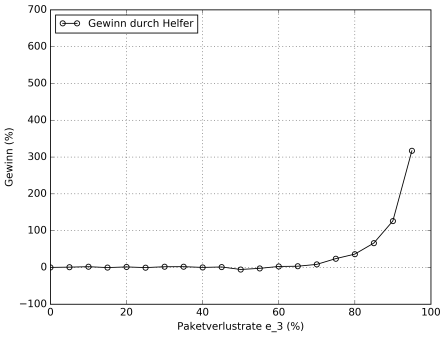
Abbildung 3.1: Simulationsergebnisse mit $e_2 = 30\%$

Jetzt wird e_2 auf 60% vergrößert und Ergebnisse werden in der Abbildung 3.2 gezeigt. Entsprechende Daten kann man aus Tabellen A.3 sowie A.4 lesen.

Daraus kann man erfahren, dass im Vergleich zu besseren Verbindungen des Helfers, fängt der Helfer nur nach größerem Wert von e_3(hier ungefähr nach 40%)an, Paket zu transportieren. Und der Schnittpunkt wird auch nach ungefähr 60% verschoben. Außerdem verringert sich der Gewinn beim schlechtesten Fall auf ungefähr 300%.



(a) Mittelwerte

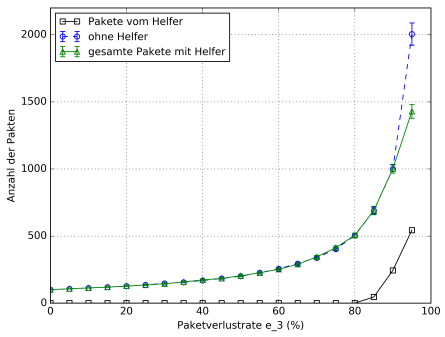


(b) Gewinn

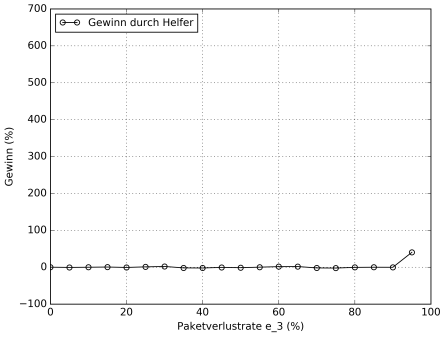
Abbildung 3.2: Simulationsergebnisse mit $e_2 = 60\%$

Schließlich wird e_2 einen Extremwert 90% nimmt, nämlich die Verbindungen des Helfer-Knotens deutlich schleicht sind. Die daraus bekommende Ergebnisse werden in Abbildung 3.3 veranschaulicht. Entsprechende Daten kann man aus Tabellen A.5 sowie A.6 lesen.

Daraus zeigt sich, dass der Helfer nur beim schlechtesten Fall zur Übertragung beiträgt. Der Gewinn durch Einstellung des Helfers ist offensichtlich niedriger im Vergleich zu anderen Situationen, nämlich meistens um 0% und kleiner als 50% beim schlechten Fall.



(a) Mittelwerte



(b) Gewinn

Abbildung 3.3: Simulationsergebnisse mit $e_2 = 90\%$

Deshalb kann Schlussfolgerungen daraus gezogen, dass in diesem Szenario(Symmetrie von e_1 und e_2) die Übertragungsleistung offensichtlich verbes-

sert durch die Einstellung des Helfer-Knotens wird, nur wenn die Paketverlustrate zwischen Sender und Empfänger(e_3) relativ schlechter als die zwischen Helfer und Sender sowie Helfer und Empfänger($e_1 = e_2$), nämlich der Helfer sollte gute Verbindungen besitzen.

4 Zusammenfassung und Aussicht

4.1 Zusammenfassung

Das Ziel dieser Arbeit ist die Implementierung der Simulationen für das in Kapitel 2.1.1 beschriebene Szenario mit Hilfe vom Kodo-Python sowie Simulationsergebnisse durch Tabellen und Grafiken darzustellen.

Bei der Situation mit dem Helfer-Knoten liegen unterschiedliche Strategien vor und in dieser Aufgabe wird ein auf RLNC basierter Algorithmus: PlayNCool verwendet.

Aus Simulationsergebnissen könnte eine Schlussfolgerung gezogen werden:

In diesem Szenario(Symmetrie von e_1 und e_2) wird die Übertragungsleistung offensichtlich verbessert durch die Einstellung des Helfer-Knotens, nur wenn die Paketverlustrate zwischen Sender und Empfänger(e_3) relativ schlechter als die zwischen Sender und Helfer sowie Helfer und Empfänger($e_1 = e_2$).

Die Simulationscodes sowie Dokumentationen stehen im Repository auf Github(`https://github.com/stevelorenz/acto15-helper`) zur Verfügung.

4.2 Aussichten

Um die Simulation zu vereinfachen, werden einige nicht allgemeine Voraussetzungen in dieser Arbeit angenommen. Diese könnten als weitere Arbeiten betrachtet werden.

4.2.1 Unsymmetrische Paketverlustraten des Helfer-Knotens

In dieser Simulation nimmt e_1 den gleichen Wert als e_2, deshalb wird die Anzahl der empfangenen Paketen am Helfer als der Schwellenwert für den Anfang der Weiterübertragung benutzt. Aber wenn diese Symmetrie nicht vorliegt, muss man überlegen und testen, ob dieser Kriterium weiter gilt. Es gibt z.B. Ausnahmezustände: Obwohl die Verbindung zwischen Helfer und Empfänger relativ gut ist, aber wegen der schlechten Übertragung zwischen Sender und Helfer sind Pakete vom Helfer linear abhängig und haben damit keinen Beitrag zur Dekodierung.

4.2.2 Übertragung mehrere Generationen von Paketen

In dieser Simulation wird nur eine Generation von Paketen gesandt. Wenn die transportierte Informationen relativ groß ist, sollte die Übertragung durch mehr Generationen notwendig sein. Damit entstehen viele komplexere Probleme, die bei der Simulation berücksichtigt werden sollten. Zum Beispiel, wie sollte sich der Sender verhalten, wenn er auf die Empfangsbestätigung der letzten Generation vom Empfänger warten.

Literaturverzeichnis

[1] Peyman Pahlevani, Daniel E Lucani, Morten V Pedersen und Frank HP Fitzek. „Playncool: opportunistic network coding for local optimization of routing in wireless mesh networks“. In: *Globecom Workshops (GC Wkshps), 2013 IEEE*. IEEE. 2013, S. 812–817 (siehe S. 1, 4, 5).

[2] Tracey Ho, Muriel Médard, Ralf Koetter, David R Karger, Michelle Effros, Jun Shi und Ben Leong. „A random linear network coding approach to multicast“. In: *Information Theory, IEEE Transactions on* 52.10 (2006), S. 4413–4430 (siehe S. 4).

[3] *Introduction to Kodo*. Steinwurf. URL: `http://steinwurf.com/kodo/` (siehe S. 5).

[4] *Github Repository of Kodo-Python*. Steinwurf. URL: `https://github.com/steinwurf/kodo-python` (siehe S. 5).

[5] *Homepage of matplotlib*. URL: `http://matplotlib.org/` (siehe S. 6).

A Anhang

Tabelle A.1: Mittelwerte sowie Standardabweichungen mit $e_2 = 30\%$

	Mittelwert			Empirische Standardabweichung		
$e_3(\%)$	Ohne Helfer	gesamt mit Helfer	nur vom Helfer	Ohne Helfer	gesamt mit Helfer	nur vom Helfer
0	100.00	100.00	0.00	0.00	0.00	0.00
5	106.90	106.66	0.00	2.19	2.48	0.00
10	113.34	113.04	0.00	4.12	4.52	0.00
15	119.60	119.50	0.58	5.07	5.51	1.42
20	126.68	125.82	3.16	5.59	6.43	3.93
25	135.62	135.76	10.12	5.99	5.50	3.97
30	145.44	145.68	18.84	8.87	7.16	4.17
35	156.54	152.84	25.74	10.28	8.97	5.59
40	169.72	163.76	32.88	11.28	8.54	4.96
45	185.40	171.88	41.24	14.00	10.62	5.62
50	206.42	184.30	50.70	16.69	13.76	7.09
55	222.70	194.72	60.22	17.48	12.83	7.03
60	249.40	201.76	67.90	18.45	12.58	6.81
65	290.10	212.10	75.82	27.48	11.63	5.93
70	339.36	223.26	85.22	26.16	15.63	7.54
75	401.66	238.36	97.06	33.12	13.04	6.95
80	511.84	248.72	105.90	46.27	13.36	7.05
85	673.96	259.08	115.60	58.16	14.81	7.75
90	1018.02	274.52	128.50	94.41	12.33	5.80
95	1996.58	292.98	142.64	214.42	16.38	8.11

Tabelle A.2: Gewinn mit $e_2 = 30\%$

$e_3(\%)$	0	5	10	15	20	25	30	35	40	45
$Gewinn(\%)$	0.00	0.23	0.27	0.08	0.68	-0.10	-0.16	2.42	3.64	7.87
$e_3(\%)$	50	55	60	65	70	75	80	85	90	95
$Gewinn(\%)$	12.00	14.37	23.61	36.78	52.00	68.51	105.79	160.14	270.84	581.47

Tabelle A.3: Mittelwerte sowie Standardabweichungen mit $e_2 = 60\%$

	Mittelwert			Empirische Standardabweichung		
$e_3(\%)$	Ohne Helfer	gesamt mit Helfer	nur vom Helfer	Ohne Helfer	gesamt mit Helfer	nur vom Helfer
0	100.00	100.00	0.00	0.00	0.00	0.00
5	107.02	106.28	0.00	2.25	3.17	0.00
10	113.78	111.62	0.00	3.90	3.70	0.00
15	119.76	120.36	0.00	5.14	4.57	0.00
20	127.50	125.88	0.00	6.71	5.76	0.00
25	136.18	136.92	0.00	7.25	6.70	0.00
30	146.60	143.96	0.00	7.90	8.40	0.00
35	157.34	154.26	0.00	9.62	8.66	0.00
40	170.00	170.08	0.46	11.20	10.73	1.83
45	187.38	185.24	2.92	11.25	13.41	4.34
50	199.96	211.88	18.16	13.05	13.37	11.13
55	223.50	229.36	30.64	19.61	17.99	12.13
60	258.08	251.74	50.68	18.13	19.00	10.67
65	286.86	277.98	67.84	18.79	18.44	11.94
70	337.72	312.10	91.64	25.56	27.87	16.52
75	419.86	339.06	113.42	39.94	22.21	12.52
80	495.84	364.46	135.96	43.39	31.53	14.99
85	677.62	407.92	166.42	59.04	27.76	15.10
90	1000.66	442.54	196.26	92.59	35.28	16.08
95	2062.32	494.68	233.90	167.64	38.40	19.17

Tabelle A.4: Gewinn mit $e_2 = 60\%$

$e_3(\%)$	0	5	10	15	20	25	30	35	40	45
$Gewinn(\%)$	0.00	0.70	1.94	-0.50	1.29	-0.54	1.83	2.00	-0.05	1.16
$e_3(\%)$	50	55	60	65	70	75	80	85	90	95
$Gewinn(\%)$	-5.63	-2.55	2.52	3.19	8.21	23.83	36.05	66.12	126.12	316.90

Tabelle A.5: Mittelwerte sowie Standardabweichungen mit $e_2 = 90\%$

	Mittelwert			Empirische Standardabweichung		
$e_3(\%)$	Ohne Helfer	gesamt mit Helfer	nur vom Helfer	Ohne Helfer	gesamt mit Helfer	nur vom Helfer
0	100.00	100.00	0.00	0.00	0.00	0.00
5	106.22	106.82	0.00	3.20	3.51	0.00
10	112.92	113.04	0.00	3.98	3.43	0.00
15	119.34	118.80	0.00	5.43	4.88	0.00
20	126.14	126.96	0.00	4.85	5.93	0.00
25	136.06	134.84	0.00	8.11	8.04	0.00
30	147.00	144.10	0.00	9.74	8.22	0.00
35	154.98	157.94	0.00	8.46	9.66	0.00
40	168.96	172.70	0.00	10.04	11.54	0.00
45	183.18	184.16	0.00	12.98	13.75	0.00
50	200.86	203.50	0.00	12.09	17.00	0.00
55	226.04	225.82	0.00	15.26	17.42	0.00
60	256.08	252.28	0.00	21.64	19.85	0.00
65	294.68	289.56	0.00	20.01	18.65	0.00
70	337.70	344.44	0.00	31.27	31.61	0.00
75	402.70	412.28	0.00	35.21	39.54	0.00
80	503.48	505.90	0.00	38.52	43.66	0.00
85	689.70	690.38	45.64	66.02	83.02	44.62
90	999.58	1001.90	243.92	80.85	86.45	57.69
95	2005.16	1428.92	543.60	218.66	134.64	73.91

Tabelle A.6: Gewinn mit $e_2 = 90\%$

$e_3(\%)$	0	5	10	15	20	25	30	35	40	45
$Gewinn(\%)$	0.00	-0.56	-0.11	0.45	-0.65	0.90	2.01	-1.87	-2.17	-0.53
$e_3(\%)$	50	55	60	65	70	75	80	85	90	95
$Gewinn(\%)$	-1.30	0.10	1.51	1.77	-1.96	-2.32	-0.48	-0.10	-0.23	40.33