Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Modellierung dynamischer Systeme

Sascha Waltz

Simulation eines Netzwerkes und seiner Auslastung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sascha Waltz

Simulation eines Netzwerkes und seiner Auslastung

Modellierung dynamischer Systemearbeit eingereicht im Rahmen der
Modellierung dynamischer Systemeprüfung
im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Wolfgang Fohl
Zweitgutachter:

Abgegeben am 31.05.2017

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In diesem Abschnitt der Ausarbeitung soll kurz dargestellt werden, worum es geht, welche Randbedingungen aufgestellt wurden und welche Ziele mit welchen Mitteln erreicht werden sollen.

1.1 Motivation

Diese Ausarbeitung entstand als Prüfungsleistung des Fachen *Modellierung dynamischer Systeme* an der **Hochschule für angewandte Wissenschaften Hamburg (HAW)**. In dieser Vorlesung werden mathematische Grundlagen gelehrt, die es den Studierenden ermöglichen sollen, das Zeitverhalten physikalischer, technischer und ökonomischer Prozesse mathematisch zu modellieren. Außerdem sollen die Studierenden in der Lage sein Simulationswerkzeuge und numerische Lösungsverfahren auszuwählen, anzuwenden und zu interpretieren.

1.2 Zielsetzung

Auf der Grundelage dieses Wissens wurden während des Praktikums schon diverse Simulationen durchgeführt. In der Regel wurde hier das Programm *MatLab*[1] genutzt, welches durch diverse Unterprogramme wie SimuLink und Ähnlichem, für die meisten Simulationen bestens geeignet ist.
Ziel dieser Ausarbeitung ist es, ein Netzwerk zu simulieren, in dem Knoten Nachrichten untereinander verschicken. Zu Beginn der Simulation kennen die einzelnen Knoten nur ihre direkten Nachbarknoten und müssen, wie in einem Netzwerk üblich, zunächst die

[1] http://de.mathworks.com/products/matlab/

Routen zu den entsprechenden Zielknoten ausfindig machen. Meistens gibt es mehrere solcher Routen, so dass zunächst die schnellste Route gewählt wird. Fällt hier aber ein Knoten oder eine Verbindung aus, muss das Netzwerk sich neu organisieren. Hierzu sollen Messungen während der Simulation angestellt werden um herauszufinden, wie lange ein Netzwerk braucht, um sich neu zu organisieren, wie viele Nachrichten dabei verloren gehen und wie viele zusätzliche Nachrichten verschickt werden, um die neue Route zu erkunden.

1.3 Gleiderung

Diese Ausarbeitung befasst sich im anschließenden Kapitel mit der Analyse der gegebenen Situation. Hier werden wichtige Faktoren und Größen ermittelt, die in einem Netzwerk Einfluss haben. Ebenso werden Verfahren und Verhalten eines Netzwerkes vorgestellt, die beim Verschicken von Nachrichten angewendet werden.
An das Analysekapitel schließt sich dann der Hauptteil der Ausarbeitung an, die eigentliche Simulation. Hier werden Verfahren erläutert, die für die Simulation genutzt wurden und anhand diverser Grafiken werden die Ergebnisse veranschaulicht.
Den Abschluss der Ausarbeitung bildet ein Fazit, in welchem die Ergebnisse und Erkenntnisse der Simulation zusammengefasst werden.

2 Analyse

Dieser Abschnitt der Ausarbeitung befasst sich mit der Beschaffenheit von Netzwerken, ihren Möglichkeiten und gängigen Verfahren. Es sollen grundlegende Begrifflichkeiten erklärt und die Abläufe näher betrachtet werden. Um den Rahmen dieser Ausarbeitung nicht zu überschreiten, wird nur auf die wichtigsten Aspekte eingegangen.

2.1 Netzwerke

Netzwerke sind streng genommen verteilte Systeme in denen diverse Komponenten Daten austauschen. Hierbei unterscheidet man zwischen *shared memory* und *message passing* Systemen. Letztere kommunizieren durch das Verschicken von Nachrichten miteinander, wie es in einem Netzwerk allgemein üblich ist. Des Weiteren werden hier synchrone und asynchrone Modelle unterschieden. Netzwerke sind in der Regel asynchrone Systeme, da hier jedes System jederzeit Nachrichten verschicken oder empfangen und Berechnungen durchführen kann. Synchrone Systeme arbeiten hier mit einem rundenbasierten System, bei dem jede Komponente pro Runde ebenfalls Nachrichten verschicken oder empfangen und Berechnungen durchführen kann, aber alle Nachrichten, die in einer Runde verschickt werden, werden auch in der selben Runde empfangen. Bei einem asynchronen Netzwerk können die Nachrichten auch nach einer beliebigen, endlichen Zeit empfangen werden. (vgl. Akili, 2015) Für die Simulation gehen wir davon aus, dass die einzelnen Komponenten des Netzwerks über Kanäle mit beliebig vielen anderen Komponenten verbunden sind und mit diesen über die entsprechenden Kanäle Nachrichten austauschen können. Hat eine Komponente keine direkte Route zu einem Ziel, wird über ein bestimmtes Verfahren eine Route zum Ziel ermittelt und die Nachricht dann über mehrere Komponenten verschickt. Die einzelnen Komponenten verhalten sich also wie Knotenpunkte oder Router in diesem Netzwerk.

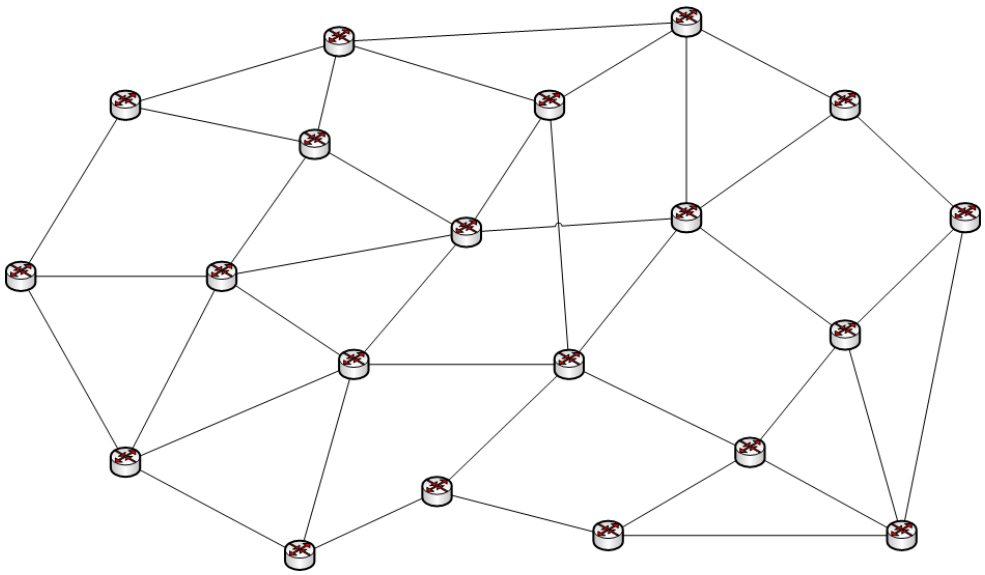

Abbildung 2.1: Beispiel für ein Netzwerk, welches in der Simulation behandelt werden kann

In Abbildung 2.1 ist ein solches Netzwerk dargestellt. Man sieht hier, dass die einzelnen Knoten mit mehreren aber nicht allen anderen Knoten direkt verbunden sind.

2.2 Routing

Das Routing beschreibt die optimale Wegwahl für Datenpakete in einem Netzwerk. Um diese Pakete vom Sender zum Empfänger zu schicken, müssen sie richtig adressiert werden. Die Wahl des Weges fällt hierbei den Routern des Netzwerkes zu. Anhand von Routing-Tabellen und -Algorithmen bestimmen sie, über welches ihrer Interfaces ein Datenpaket am besten weitergeleitet werden muss, um an sein Ziel zu gelangen. Man unterscheidet hier zwischen statischem und dynamischem Routing.
Bei statischem Routing werden die Routen in den Routern fest eingespeichert, so dass Datenpakete immer über die selbe Route geschickt werden. Fällt hier nun ein Router aus oder ist die Route überlastet, wird auf diesen Ausfall nicht automatisch reagiert und die Pakete können nicht mehr zugestellt werden.
Das dynamische Routing dagegen ermöglicht es den Routern für jedes Datenpaket die

zum Zeitpunkt des Versendens beste Route zu ermitteln und das Paket dementsprechend weiterzuleiten. Hierzu gibt es diverse verschiedene Routing-Protokolle und , die es den Routern ermöglichen untereinander Informationen zu Routen auszutauschen. Dadurch kann automatisch auf einen Ausfall oder eine Überlastung reagiert werden und Datenpakete erreichen trotzdem weiterhin ihr Ziel, solange eine Route existiert. Hierzu aktualisieren die Router ihre Routing-Tabelle automatisch, sobald sich an den Gegebenheiten etwas ändert.

2.2.1 Routing-Protokolle und Algorithmen

Es gibt diverse Routing-Protokolle, die dynamischen Routing seit Jahren eingesetzt werden. Um den Rahmen der Ausarbeitung nicht zu sprengen, sollen an dieser Stelle nur die wichtigsten kurz erläutert werden.

EIGRP - Enhanced Interior Gateway Routing Protocol Von der Firma Cisco[1] als Nachfolger des IGRP entwickelt, gehört das Enhanced Interior Gateway Routing Protocol immer noch zu den Standards der Interior Gateway Protocols die weltweit zum Einsatz kommen. EIGRP selbst ist ein Enhanced Distance Vector Protocol, welches den Diffused Update Algorithm (DUAL) verwendet, um die kurzeste Route durch ein Netzwerk zu ermitteln. (vgl. Cisco Engineers, 2017) Der Diffused Update Algorithm sorgt für schnelle und schleifenfreie Berechnung der Routen, indem nur Knoten, die von einer Veränderung der Netzwerkumgebung betroffen sind, agieren und ihre Informationen über die Veränderung in den Routen an ihre Nachbarn übermitteln. (vgl. Savage u. a., 2016)

OSPF - Open Shortest Path First Das OSPF-Protokoll in Version 2 wurde bereits 1998 von der Internet Engineering Task Force (IETF)[2] definiert. Es basiert auf Dijkstra's "shortest-path"-Algorithmus und ist ein sogenanntes Link-State-Routing-Protokoll. Basis der Routenberechnung bildet die Link-State-Datenbank eines Routers, in der alle benachbarten Router enthalten sind. Informationen über den Status der Nachbarn werden per Flooding übermittelt, wobei Nachrichten an sämtliche benachbarte Router geschickt werden. (vgl. Moy, 1998)

[1] Webseite der Firma Cisco - https://www.cisco.com - Letzter Zugriff: 08.09.2017
[2] Webseite der Internet Engineering Task Force - https://www.ietf.org - Letzter Zugriff: 08.09.2017

AODV - Ad-hoc On-demand Distance Vector Als drittes soll hier noch ein Ad Hoc Protokoll erwähnt werden. Das AODV Routing Protokoll ist dafür entwickelt worden, um in Ad Hoc Netzwerken mit mobilen Knoten schnell ein Netzwerk aufzubauen und auf die häufigen Veränderungen direkt und ressourcenschonend reagieren zu können. Bei dieser Art der Routing-Protokolle werden die Routen erst bei Bedarf ermittelt und nicht die komplette Topologie in einer Routing-Tabelle gespeichert. Erst wenn eine Route neu erstellt werden muss, weil ein Knoten dazu gekommen oder weggefallen ist, werden neue Routen angefragt. Die Ermittlung erfolgt per Route Requests (RREQs), Route Replies (RREPs) und Route Errors (RERRs) Nachrichten. (vgl. Perkins u. a., 2003)

2.2.2 Dynamic Source Routing (DSR)

Das DSR-Protokoll gehört zu den Ad Hoc Routing Protokollen, da auch hier Routen erst gesucht werden, sobald sie benötigt werden. In einem neuen Netzwerk kennen die einzelnen Komponenten zunächst nur sich selbst und im besten Fall ihre direkten Nachbarn mit denen sie direkt verbunden sind. Wenn nun eine Komponente eine Nachricht an einen unbekannten Empfänger verschicken will, muss zunächst herausgefunden werden, über welchen Weg der Empfänger erreicht werden kann. Hierzu verschicken die Komponenten sogenannte Route-Request Nachrichten, die nur den Zweck erfüllen, eben solche Routen herauszufinden. Die Nachricht merkt sich hierzu, welche Knoten sie passiert und beim Erreichen des Ziels wird die herausgefundene Route an den Urheber der Nachricht zurück gesendet und dieser kann seine Nachrichten an die Ziel-Komponente senden. Um das Ziel überhaupt zu erreichen, wird das Netzwerk mit den Route-Request Nachrichten geflutet, sprich diese werden als Broadcast-Nachrichten an alle Nachbarn gesendet. Hat ein Knoten diese Nachricht bereits erhalten und weitergeleitet, reagiert er nicht, sollte er die Nachricht ein weiteres Mal erhalten. Dieses Verfahren nennt sich Dynamic Source Routing und wurde bereits 1996 für Ad Hoc Wireless Networks entwickelt. (vgl. Johnson und Maltz, 1996) Es wurde für diese Simulation ausgewählt, da es ein recht simples, gut nachvollziehbares und schnell zu implementierendes Verfahren ist.

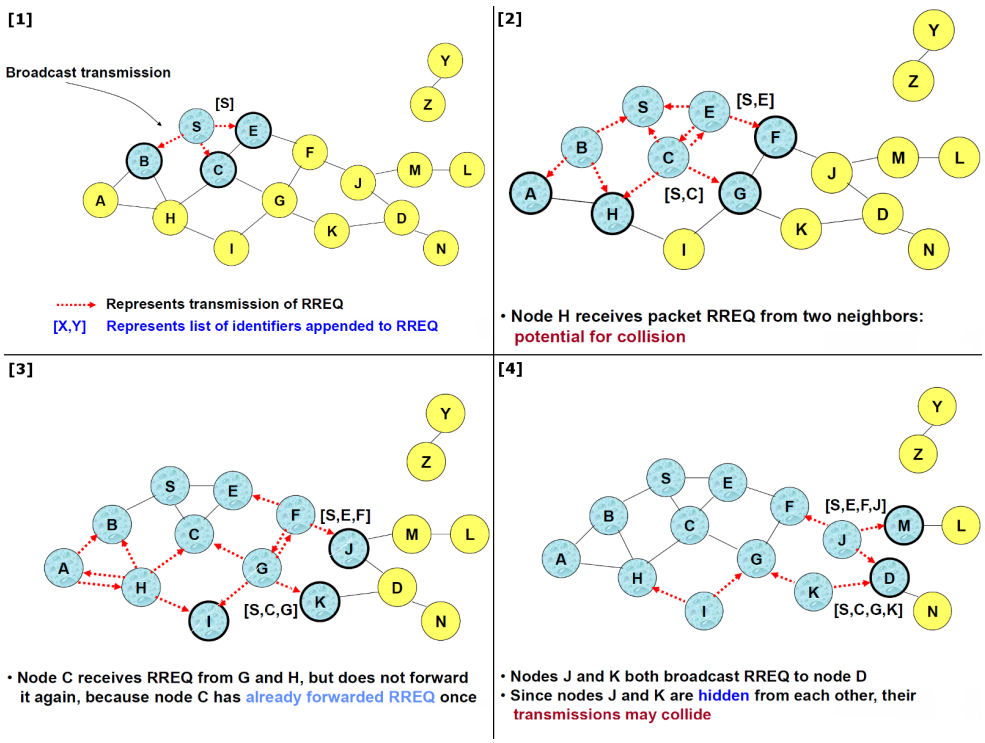

Abbildung 2.2: Route Discovery Nachrichten im Dynamic Source Routing Schmidt

In Abbildung 2.2 dargestellt ist eine Route Request von Knoten S nach D. Im ersten Abschnitt [1] sendet S seine Route Request Nachricht an alle seine Nachbarn mit seinem eigenen Namen als Inhalt. Im zweiten Abschnitt [2] senden diese Nachbarn wiederum Broadcast Nachrichten an alle ihre Nachbarn und hängen ihren eigenen Namen an den Inhalt dran. Erhält ein Knoten Route Request Nachrichten von 2 seiner Nachbarknoten kollidieren diese, da ein Knoten eine Route Request Nachricht nur einmal weiterleitet, wie in Abschnitt [3] zu sehen. Da die Nachrichten nacheinander abgearbeitet werden, wird die zuerst eingegangene Nachricht mit den bis dahin eingetragenen vorangegangenen Knoten weitergeleitet. In Abschnitt [4] erreichen die Route Requests das Ziel D, hier wird dann ein Route Reply über den im Route Request ermittelten Pfad an den Absender zurück geschickt, womit dieser dann die entsprechende Route kennt.

2.3 Anforderungen

Die aus dieser Analyse abgeleiteten optimalen Anforderungen für die Simulation stellen sich wie folgt dar:

- Zu Beginn unbekanntes Netzwerk
- Nur direkte Nachbarn sind bekannt
- Herausfinden der Routen
- Broadcast-Nachrichten um Routen herauszufinden
- Rückmeldungen an den Sender bei erfolgreicher Routenfindung
- Reaktion auf Veränderungen

Diese Anforderungen stellen die wichtigsten Punkte dar, die sich für eine Simulation eines Netzwerks ergeben. Für die Simulation wird das Dynamic Source Routing genutzt, da es gut zu implementieren ist und es die genannten Punkte größtenteils gut erfüllt.

2.3.1 Minimal-Entwurf

Die Simulation, welche für diese Ausarbeitung erstellt wurde, beinhaltet ein Modell eines Netzwerkes, anhand dessen die Simulation durchgeführt wird. Dieses Modell ist ein reduziertes Abbild der Realität und wurde soweit vereinfacht, dass es in einem angemessenen Rahmen umgesetzt werden konnte und für die Simulation ausreichende Ergebnisse liefert.
Anhand dieser Simulation sollen Aussagen über das Verhalten eines realen System getroffen werden können. Je nach Simulation kann die Größe und der Grad des Modells weiter angepasst werden. In Abbildung 2.3 ist das in der Simulation verwendete Modell eines Netzwerks abgebildet. Es besteht aus 9 Knoten und 13 Kanten. Jeder Knoten ist mit mindestens einem Nachbarknoten verbunden, es gibt also keine unerreichbaren Ziele.
Die Umsetzung der Simulation erfolgt in Java, eine Benutzeroberfläche zum Visualisierung und Steuerung wird ebenfalls implementiert. Im Gegensatz zu einem echten

Netzwerk, wird die Anzahl der Nachrichten reduziert um eine gute Visualisierung zu ermöglichen.

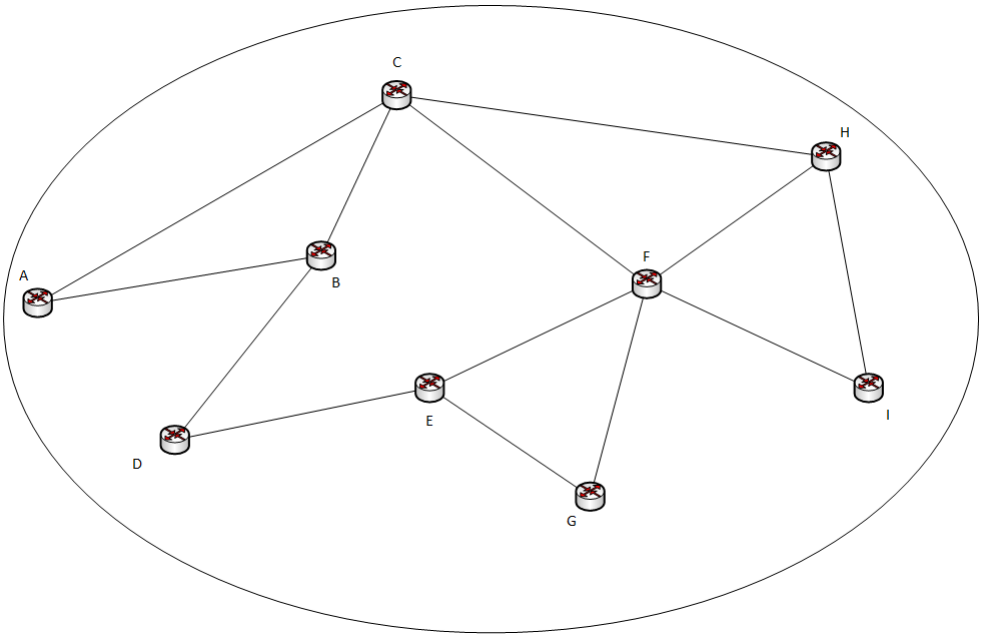

Abbildung 2.3: Netzwerk für die Simulation

3 Entwurf

Dieser Abschnitt der Ausarbeitung behandelt den Entwurf, Aufbau, Ablauf und die Auswertung der Simulation. Hier werden außerdem die Ergebnisse dokumentiert und erläutert.

3.1 Architektur

Die Architektur der Netzwerk-Simulation ist recht simpel: Es gibt eine bestimmte Anzahl an Knoten-Objekten, die alle über einen Netzwerk-Kanal des Multi-Agenten-Systems verbunden sind und miteinander Nachrichten austauschen können.
Ein Knoten kann sich die Routen zu anderen Knoten in einer Map speichern, quasi als Routing-Tabelle, und muss so nicht jedes Mal erneut nach der Route fragen. Die Knoten sind in der Lage, neben den normalen Nachrichten und Antworten außerdem noch Route-Requests und -Replies zu verschicken.
Eine Benutzeroberfläche stellt die Vorgänge graphisch dar und ermöglicht einem User die Steuerung der Knoten.

3.1.1 Multi-Agenten-System

Um die verschiedenen Knoten des Netzwerks zu simulieren, wird ein Multi-Agenten-System verwendet. Diese sind dafür ausgelegt, ein System zu realisieren, in dem mehrere Agenten untereinander Nachrichten austauschen und so miteinander kommunizieren und zusammenarbeiten. Multi-Agenten-Systeme sind sehr vielseitig einsetzbar und kommen unter anderem in verteilten Systemen oder für verteilte künstliche Intelligenz zum Einsatz. Dank der weiten Verbreitung sind diverse Frameworks verfügbar,

mit denen die Entwicklung vereinfacht wird. In der Simulation kommt das Akka-Framework[1] zum Einsatz. Dieses bietet einige Vorteile, wie zum Beispiel: einfache Erstellung von Aktoren (Agenten) und JavaScript Object Notation (JSON) als Nachrichtensprache.
Um Nachrichten in Akka zu verschicken, werden Kanäle erstellt, die von Agenten abonniert werden. Hat ein Agent einen Kanal abonniert, kann er hier Nachrichten hinein schreiben und auslesen. Nachrichten in Kanälen können immer von allen gelesen werden, die den entsprechenden Kanal abonniert haben.
Um den Aufbau der Simulation zu vereinfachen, werden alle Nachrichten des Netzes über einen Kanal gesendet, das bedeutet, jeder Knoten kann jede Nachricht sehen, reagiert aber nur auf die für Ihn Relevanten. Somit ist das Netz besser skalier- und erweiterbar als wenn jeder Knoten einen eigenen Kanal mit seinen Nachbarknoten aufbauen würde.

3.1.2 Benutzeroberfläche

Wie schon angesprochen, erhält das Programm zur Simulation eine Benutzeroberfläche. Diese wird in JavaFX umgesetzt und ist somit direkt im Programm integriert.

[1] Webseite des Akka-Projekts - http://akka.io; letzter Zugriff: 08.09.2017

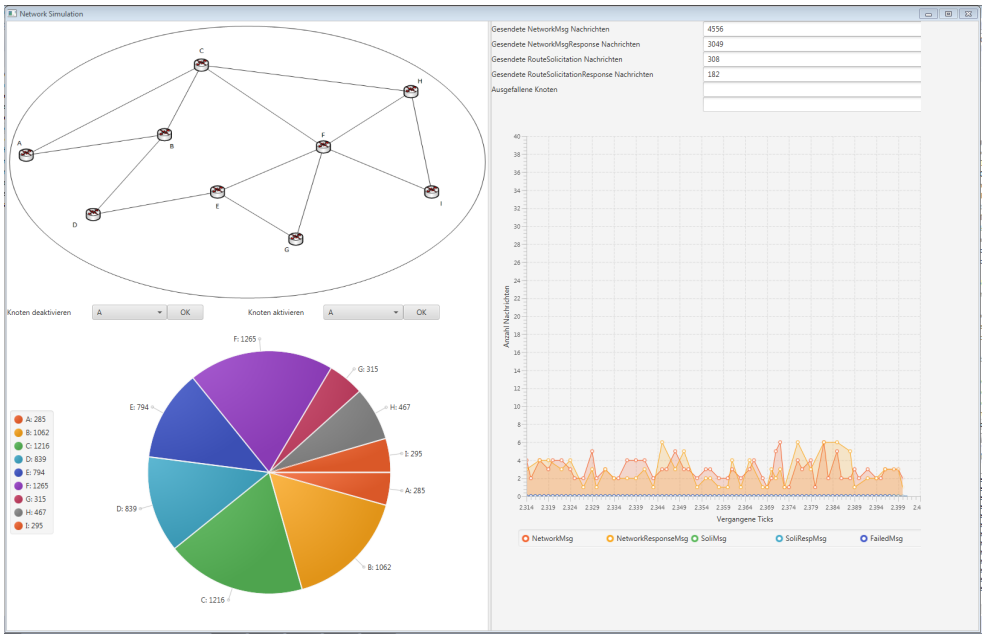

Abbildung 3.1: Benutzeroberfläche

Die Benutzeroberfläche ist, wie in Abbildung 3.1 zu sehen, in mehrere Informationsbereiche aufgeteilt. Zum einen sieht man links oben eine Übersicht über das Netzwerk, welches simuliert wird. Direkt darunter lassen sich Knoten deaktivieren und wieder aktivieren. Das PieChart zeigt an, welcher Knoten wie viele Nachrichten verschickt hat. Hierzu wird es mit jeder Zeiteinheit (Tick) aktualisiert.
Der rechte Bereich zeigt an, wie viele Nachrichten von welcher Art verschickt wurden. Das LineChart-Diagramm stellt die aktuell verschickten Nachrichten grafisch dar, so dass leicht erkennbar ist, wie der Nachrichtenverkehr sich aktuell gestaltet. Die Zeitangabe ist hierbei in Ticks. Ein Tick ist in der Simulation definiert als 100 Millisekunden.

4 Evaluation

In diesem Abschnitt wird die Simulation und ihre Ergebnisse evaluiert. Es soll gezeigt werden, wie sich das Netzwerk bei seiner Initialisierung verhält und wie die Nachrichten sich mit der Zeit verlagern, sobald die Routen bekannt werden. Außerdem soll gezeigt werden, was mit dem Netzwerk passiert, wenn ein oder mehrere Knoten ausfallen.
Die Simulation steht unter folgendem Link zur Verfügung: https://github.com/squas82/md-simulation

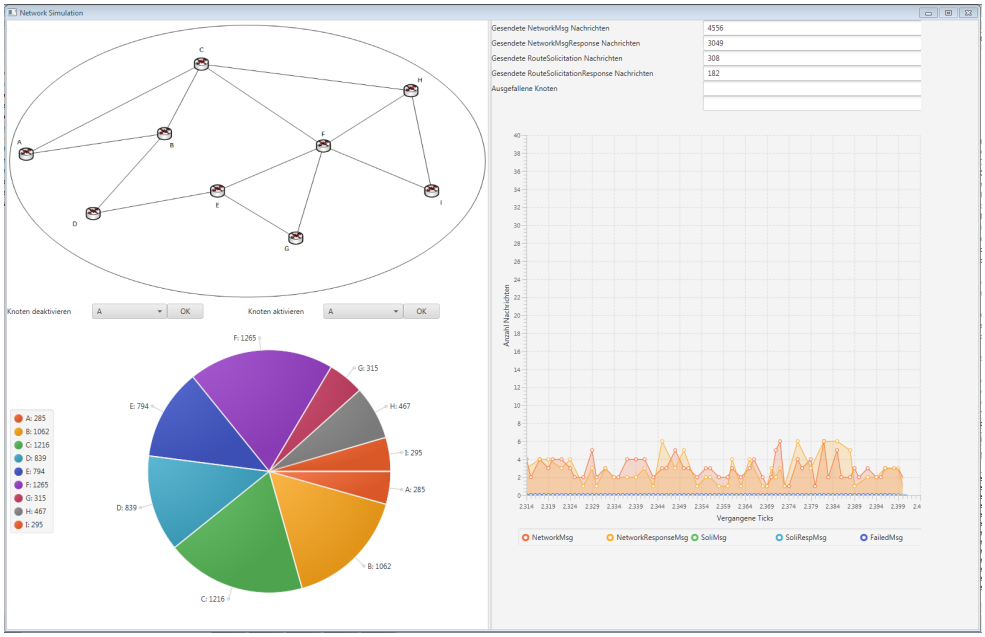

Abbildung 4.1: Normaler Ablauf im stabilen Netzwerk

4.1 Ablauf der Simulation

Zu Beginn der Simulation erkennt man in Abbildung 4.2, das noch sehr viele Nachrichten zur Routen-Erkennung verschickt werden, da das Netz noch weitestgehend unbekannt ist. Der grüne Verlauf im Diagramm stellt die Anfragen, der Türkise die Antworten dar. Wie schon in Abschnitt 2.2.2 beschrieben, müssen zu Beginn, um das Netzwerk kennen zu lernen, viele Route-Request Nachrichten verschickt werden. Diese und die dazugehörigen Antworten überwiegen das Diagramm am Anfang deutlich, nehmen aber mit der Zeit immer weiter ab. Hat sich das Netz normalisiert, werden etwa gleich viele Nachrichten wie Responses verschickt. Die Route-Request und -Response Nachrichten kommen nahezu zum erliegen, da die Routen mit der Zeit alle erkannt wurden. Da wir ein ideales Netz angenommen haben, gehen auch keine Nachrichten verloren (blauer Graph).

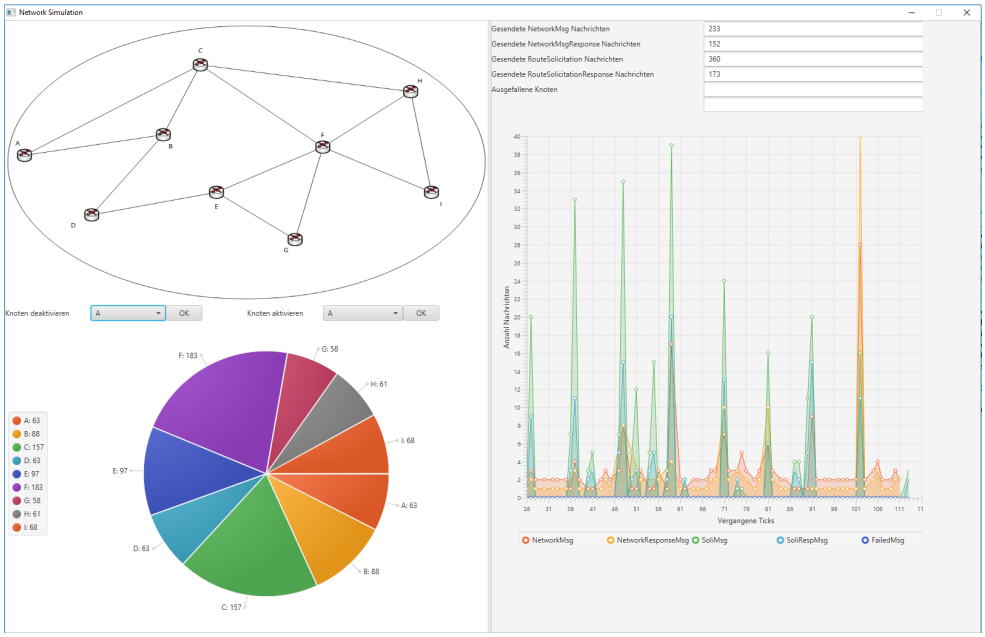

Abbildung 4.2: Start der Simulation

4.2 Auswertung

Die Auswertung der Simulation bezieht sich hier auf das Verhalten des Netzwerks, hier lassen sich Erwartungen und Beobachtungen vergleichen. Eventuell lassen sich Rückschlüsse durch das Verhalten ziehen, die nicht direkt ableitbar sind.

4.2.1 Anzahl Nachrichten

Das PieChart in der Benutzeroberfläche zeigt die versendeten Nachrichten pro Knoten im Verhältnis zueinander an. In Abbildung 4.4 kann man erkennen, dass es einige Knoten gibt, die mehr Nachrichten verschicken und einige, die wesentlich weniger aktiv sind.

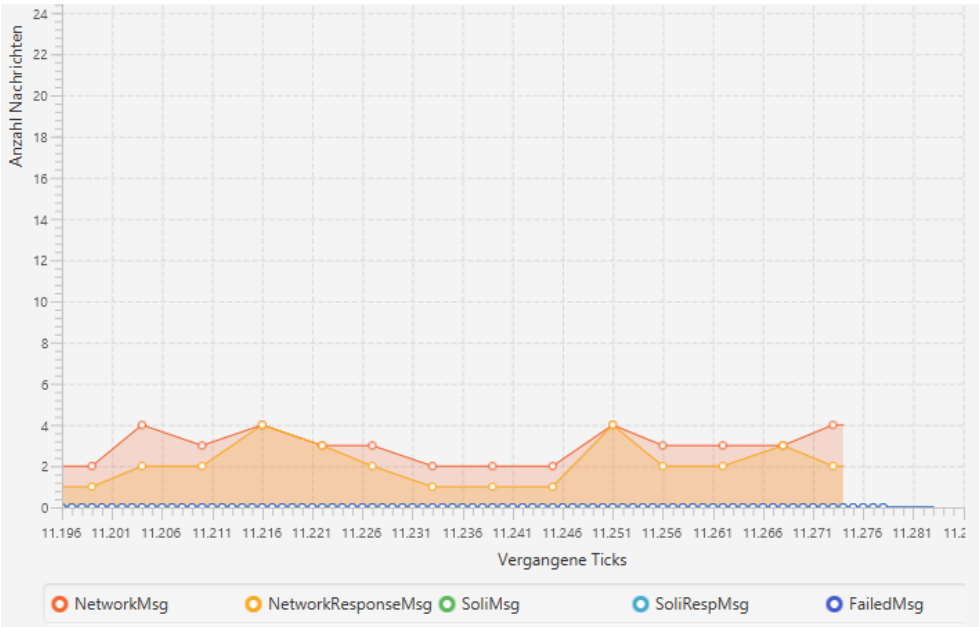

Abbildung 4.3: Nachrichtenversand nach etwa 11.000 Ticks

Nach einem Durchlauf mit etwas mehr als 10.000 Ticks lief das Netzwerk durchgehend stabil, die Nachrichten hatten sich bei etwa 10 pro Tick eingependelt. In Abbildung 4.4 ist die Aufteilung der Nachrichten pro Knoten zu sehen. Auffällig hierbei ist, dass

die Knoten C und F die meisten Nachrichten verschickt haben. Dies ließe sich damit erklären, dass diese beiden Knoten die meisten Verbindungen zu Nachbarn aufweisen, Knoten C nämlich vier und Knoten F sogar 5. Trotzdem hat Knoten C noch etwa 100 Nachrichten mehr versendet als Knoten F. Dies wiederum lässt vermuten, dass auch die Aktivität und die ausgewählten Routen der Nachbar-Knoten darauf Einfluss haben.

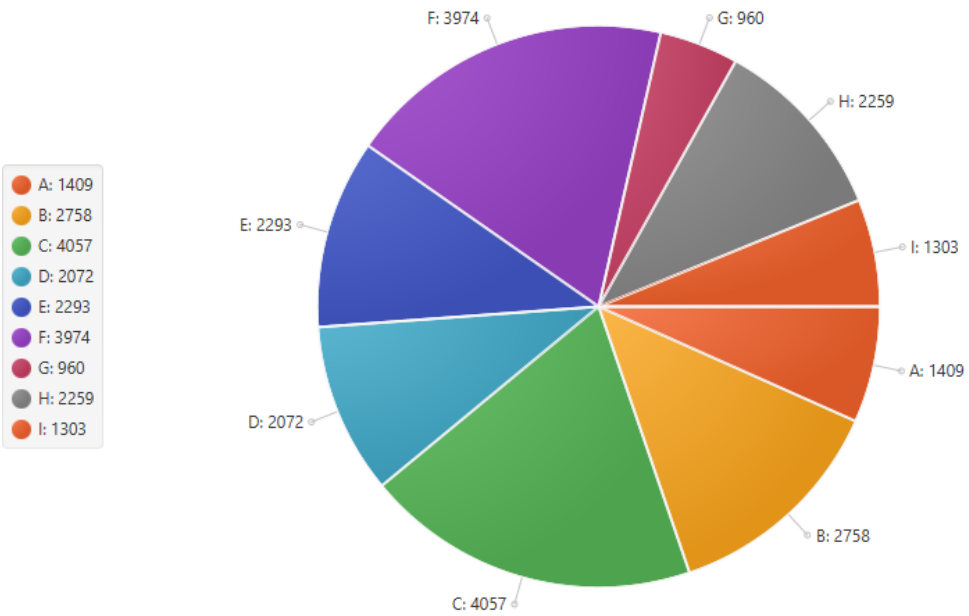

Abbildung 4.4: Nachrichtenversand pro Knoten nach etwa 11.000 Ticks

In Tabelle 4.1 sind die versendeten Nachrichten noch einmal übersichtlich ins Verhältnis gesetzt um darzustellen, dass es nicht alleine an der Anzahl der Nachbarn liegt, wie hoch der Anteil versendeter Nachrichten ist, auch die Routenauswahl der Nachbarn trägt dazu bei, welcher Knoten besonders viele Nachrichten übermitteln muss.

Tabelle 4.1: Vergleich der Nachrichten pro Knoten und Nachbarn

Knoten	**Nachrichten**	**Anteil**	**Nachbarn**	**Nachrichten / Nachbarn**
A	1409	7%	2	705
B	2758	13%	3	919
C	4057	19%	4	1014
D	2072	10%	2	1036
E	2293	11%	3	764
F	3974	19%	5	795
G	960	5%	2	480
H	2259	11%	3	753
I	1303	6%	2	652
Gesamt	21085			

4.2.2 Verhalten bei Ausfall eines Knotens

Bei einem Ausfall eines Knotens dürfen über diesen keine Nachrichten mehr gesendet werden. Dies kann man während der Simulation sehr gut an dem PieChart sehen, da hier die Anzahl der Nachrichten beim deaktivierten Knoten gleich bleibt. Da Nachrichten, die über diesen Knoten verschickt werden sollen, nun aber nicht mehr ankommen, müssen die verschickenden Knoten erneut nach einer Route suchen. Wie in Abbildung 4.5 zu sehen, sorgt dies wieder für einen großen Anstieg von Route Request Nachrichten. Dazu kommt, dass einige Nachrichten nun auch verloren gehen, bis eine neue Route gefunden wurde. Die maximalen Versuche eine Nachricht zuzustellen, wurden im Programm auf 10 gesetzt, danach wird ein neuer Route-Request gestartet.

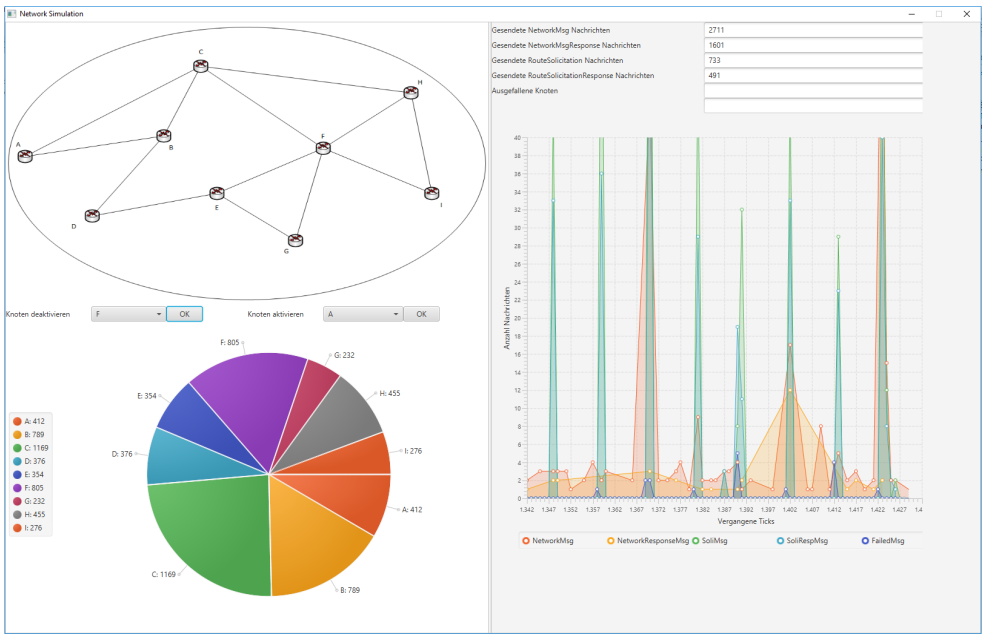

Abbildung 4.5: Verhalten der Simulation bei Ausfall eines Knoten

Ein Problem besteht an dieser Stelle noch, scheinbar im Algorithmus zur Erkennung wieder aktivierter Knoten, nachdem diese ausgefallen waren. Das Netz kommt oftmals nicht wieder in einen stabilen Zustand zurück, leider bliebt zum Ende dieser Ausarbeitung nicht genug Zeit um diesen Fehler noch auszumerzen.

4.3 Fazit

Der Start und der normale Ablauf der Simulation sind wie in den Anforderungen in Abschnitt 2.3 beschrieben umgesetzt worden. Die Auswertungen und die Darstellung der laufenden Simulation über die Benutzeroberfläche funktionieren gut und sind übersichtlich strukturiert.
Einzig der Ausfall und die wieder Aktivierung von Knoten macht oftmals Probleme. Die Simulation geht mit einem Ausfall korrekt um, kommt aber trotzdem nicht wieder in einen stabilen Zustand durch das Suchen neuer Routen.

5 Fazit

5.1 Zusammenfassung

Im Rahmen der Ausarbeitung wurde eine Simulation erstellt, die eine Netzwerkumgebung simuliert. Um eine möglichst reale Simulation erstellen zu können, wurden zunächst der Aufbau, die Eigenarten, Protokolle und Algorithmen analysiert. In Kapitel 2 wurden diese Ergebnisse zusammengefasst dargestellt.
Mit diesen gewonnenen Erkenntnissen wurde dann in einen Entwurf erstellt, der die Anforderungen an die Simulation darstellte und in Abschnitt 2.3.1 dargestellt ist. In diesem Entwurf wurden Annahmen formuliert, die in der Entwicklung vorausgesetzt wurden. Die eigentliche Entwicklung und der Aufbau der Simulation wurde ich Abschnitt 3 erläutert. Die Simulation wurde mit Java, dem Akka-Framework und JavaFX umgesetzt.
Die Ergebnisse der Simulation wurden in Kapitel 4 dargestellt und evaluiert. Des Weiteren wurde ein Fazit gezogen.

5.2 Ausblick

Zukünftige Arbeiten könnten an diese Arbeit anknüpfen, indem sie den implementierten Routing-Algorithmus verfeinern und perfektionieren. Das Verhalten bei Aufall und wieder Eingliederung von Knoten und das hinzufügen weiterer Knoten während der Simulation wären ebenso weiter zu führende Themen. Die Kanäle zwischen den Knoten sollten eingerichtet werden, damit jede Kommunikation zwischen den einzelnen Knoten direkt abläuft.

Literaturverzeichnis

[Akili 2015] Akili, Samira: *Simulation zertifizierender Netzwerkalgorithmen*. 2015. – URL https://www2.informatik.hu-berlin.de/top/_media/www/mitarbeiter/simulation_zertifizierender_netzwerkalgorithmen_samira_akili.pdf

[Cisco Engineers 2017] Cisco Engineers: *Enhanced Interior Gateway Routing Protocol*. September 2017. – URL https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html. – Zugriffsdatum: 2017-09-07

[Johnson und Maltz 1996] Johnson, David B. ; Maltz, David A.: *Dynamic Source Routing in Ad Hoc Wireless Networks*. S. 153–181. In: Imielinski, Tomasz (Hrsg.) ; Korth, Henry F. (Hrsg.): *Mobile Computing*. Boston, MA : Springer US, 1996. – URL https://doi.org/10.1007/978-0-585-29603-6_5. – ISBN 978-0-585-29603-6

[Moy 1998] Moy, John: OSPF Version 2 / Internet Engineering Task Force. IETF, April 1998 (54). – STD. – URL https://www.ietf.org/rfc/rfc2328.txt

[Perkins u. a. 2003] Perkins, C. ; Belding-Royer, E. ; Das, S.: Ad hoc On-Demand Distance Vector (AODV) Routing / Internet Engineering Task Force. IETF, July 2003 (3561). – RFC. – URL https://www.ietf.org/rfc/rfc3561.txt. – ISSN 2070-1721

[Savage u. a. 2016] Savage, D. ; Ng, J. ; Moore, S. ; Slice, D. ; Paluch, P. ; White, R.: Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP) / Internet Engineering Task Force. IETF, May 2016 (7868). – RFC. – URL https://www.ietf.org/rfc/rfc7868.txt. – ISSN 2070-1721

[Schmidt] Schmidt, Prof. Dr. T.: *MANET Routing*. Vorlesungsskript. – URL https://inet.haw-hamburg.de/teaching/ws-2016-17/technik-und-technologie-vernetzter-systeme/manet-routing. – Zugriffsdatum: 03.06.2017