universitätbonn

Internet Routing Situationen mit GNS3 darstellen

Thomas Trimborn

29.09.2014

Bachelorarbeit

Prüfer: Prof. Dr. Michael Meier
2. Prüfer: Dr. rer. nat. Matthias Frank
Betreuer: Dipl.-Inform. Matthias Wübbeling

Selbstständigkeitserklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Bonn, 29.09.2014

Thomas Trimborn

Inhaltsverzeichnis

1 Einleitung

Diese Bachelorarbeit beschreibt den Entwurf, die Implementierung und die Auswertung einer Software, die aus Internet Routing Daten automatisch eine Simulation mit graphischer Repräsentation in GNS 3 erstellt.

Das Internet ist ein Netz aus kleineren Netzen, den Autonomen Systemen, die miteinander Daten an bestimmten Punkten austauschen. Bei der Nutzung des Internet verlässt man sich darauf, dass die Daten, die man an eine bestimmte Internet Protocol (IP) Adresse sendet, bei dem richtigen Empfänger ankommen. Liegen Sender und Empfänger im gleichen Autonomen System, ist der Weg eines Datenpakets nur innerhalb dieses Systems. Liegt aber der Empfänger in einem anderen Autonomen System als der Sender, ist der Weg um einiges komplizierter als im einfachen Fall. Der Sender kennt nicht den Weg, den sein Datenpaket nehmen muss, sondern das Autonome System sendet das Datenpaket zu seiner Grenze, an der es mit anderen Autonomen Systemen verbunden ist. Dort befinden sich Router, die dank des Border Gateway Protocol (BGP) wissen, an welchen Router eines benachbarten Autonomen Systems sie das Datenpaket senden müssen, um es auf dem schnellsten Weg zu seinem Ziel zu bringen. Über das BGP teilen sich Router gegenseitig mit, welche IP Adressen in ihrem eigenen Autonomen System zu finden sind und welchen Weg sie zu IP Adressen kennen, die nicht in ihrem eigenen Autonomen System liegen. Die Daten werden dann zu dem benachbarten Router gesendet, der angibt die kürzeste Verbindung zu dem Autonomen System zu haben, das die Zieladresse für sich beansprucht. Der empfangende Router sendet die Daten dann zu einem anderen Router im eigenen Autonomen System, der wieder eine Verbindung zu anderen Autonomen Systemen hat. Dort wiederholt sich der Prozess, indem die Daten wieder zu dem Nachbarrouter gesendet werden, der angibt die kürzeste Verbindung zum Ziel zu haben. Dies wiederholt sich so lange, bis die Daten am Ziel angekommen sind. Beim Austausch der BGP Daten vertrauen sich die Nachbarn implizit, sodass jeder senden kann was er möchte, ohne dass Fehler erkannt werden.

In diesem Routing genannten Prozess treten immer wieder Annomalien auf, die sehr unterschiedlicher Natur sind, und nicht korrigiert werden, da ja den Daten der Nachbarn immer vertraut wird. Es gibt viele Fälle, bei denen verschiedene Autonome Systeme die gleichen Adressbereiche für sich beanspruchen. Das kann versehentlich passieren, wenn sich ein Administrator bei der Konfiguration eines Systems vertut, aber auch mit Absicht. Absichtlich kann diese Anomalie entweder einen sinnvollen Ursprung haben, um eine Lastverteilung zu erzeugen, da die Daten immer zu dem Ziel geschickt werden, das näher am Sender liegt. Aber es kann auch Jeder, der Zugang zur Konfiguration eines Autonomen Systems hat, dies tun um Daten zu sich umzuleiten, die er nicht bekommen sollte, so aber bekommt. Eine weitere Anomalie, die auftreten kann, ist wenn eine Route zwischen Autonomen Systemen verschwindet oder ein ganzes Autonomes System ausfällt, weil etwa ein Unterseekabel beschädigt ist. Durch das implizierte Vertrauen zwischen den Autonomen Systemen ist es auch möglich, dass ein möglicher Angreifer zu einem bestimmten Ziel eine kürzere Route annonciert, die nicht existiert. Dazu kann er alle Daten mitlesen, die er auf diesem Weg umleitet. Wenn sein eigenes Autonomes System gut plaziert ist, ist es sogar möglich, die Daten danach zum eigentlichen Empfänger weiterzuleiten und dadurch das Mitlesen komplett zu verschleiern.

All diese Anomalien tauchen in den Routingtabellen der Router der Autonomen Systeme auf und werden von manchen Projekten zur Verfügung gestellt. Ein Beispiel dieser Daten ist in Listing 1 zu sehen.

```
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.194.82|8222|221.134.79.0/24|8222 1299 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.192.133|29686|221.134.79.0/24|29686 3257 9498 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.194.140|25220|221.134.79.0/24|25220 9498 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.192.98|9189|221.134.79.0/24|9189 6695 9498 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.192.14|4589|221.134.79.0/24|4589 9498 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.194.119|34288|221.134.79.0/24|34288 9498 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.192.222|680|221.134.79.0/24|680 174 174 9498 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.192.194|6762|221.134.79.0/24|6762 1299 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.192.175|553|221.134.79.0/24|553 9498 9583|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|80.81.192.30|3257|221.134.79.0/24|3257 9498 9583|IGP
```

Listing 1: Ausschnitt aus Routingdaten

Betrachtet man die aktuellen Daten, die öffentlich zugänglich sind, so enthält ein aktueller Datensatz mit Daten aus dem Internet Exchange Punkt in Frankfurt etwa 9 Millionen Zeilen. Möchte man eine bestimmte Anomalie analysieren, so ist eine große Menge an Daten zu durchforsten, die durch die Darstellungsform sehr unübersichtlich ist. Für die beschriebenen Anomalien ist es wichtig das IP Prefix, den Adressbereich in der dritten Spalte von hinten, sowie die Route zum Ziel in der vorletzten Spalte zu betrachen. Wenn man dazu einen besseren Überblick des globalen Routings haben möchte, kommen viele andere Datensätze von anderen Betrachtungspunkten von vielen Orten auf der ganzen Welt dazu.

Um dies zu vereinfachen wurde im Rahmen dieser Bachelorarbeit die Software **A**utomatische **R**outing **D**aten **A**nalyse (**ARDA**) erstellt. Diese Software parst die Daten, die die Routingdaten beinhalten, analysiert deren Inhalt und erstellt eine Definition von Regionen aus Autonomen Systemen, die sich mit der Software Graphical Network Simulator 3 (GNS3) starten lässt. Eine Simulation in GNS3 hat eine graphische Repräsentation, für die ein Beispiel in Abbildung 1 gezeigt wird. Anhand einer solchen Repräsentation lassen sich Änderungen in den Routen leicht erkennen und auch annoncierte IP Prefixe lassen sich zurückverfolgen. So kann man das Routing vor, während und nach dem Auftreten einer Anomalie simulieren und analysieren, um die Anomalie zu erforschen. Die Software ARDA soll also die von verschiedenene Projekten zur Verfügung gestellten Internet Routing Daten analysieren und daraus deine Simulation erstellen, in der sich Anomalien analysieren lassen.

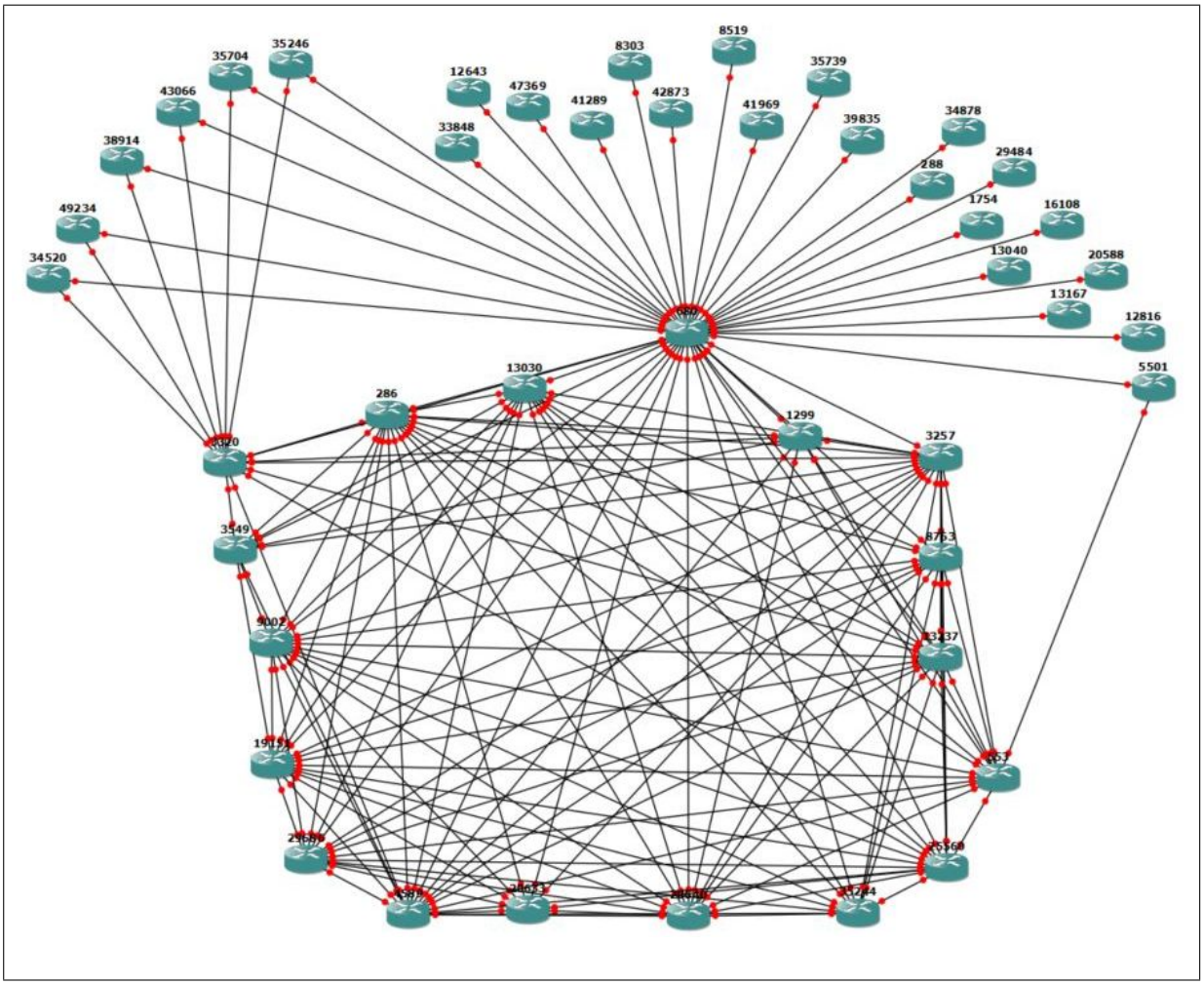

Abbildung 1: Beispiel für eine graphische Darstellung einer Routing Situation

Im Folgenden werden daher im Kapitel 2 die Grundlagen erläutert, die für das Verständnis von BGP, Anomalien und der Simulation nötig sind. Anschließend wird in Kapitel 3 ausführlich auf das Design, den Aufbau und die Implementation der Software ARDA eingegangen. Danach folgt im Kapitel 4 eine Evaluation der Fähigkeiten der Software im Vergleich zu den Anforderungen, die an sie gestellt werden. Ein Überblick über die mögliche Weiterentwiclung der Software wird im Kapitel 5 gegeben. Zum Schluss wird im Kapitel 6 zusammengefasst, was in dieser Bachelorarbeit erreicht wurde.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für die im Rahmen der vorliegenden Bachelorarbeit entstandenen Software beschrieben. Als erstes werden die Daten, die benötigt werden um eine Simulation der Routingsituation im Internet erstellen zu können, welches Format diese haben und woher sie stammen. Weiterhin werden Begriffe und Zustände erläutert, die im Weiteren benötigt werden oder relevant sind für das Verständnis des Themas. Zuerst wird im Abschnitt 2.1.1 beschrieben was BGP ist, wofür dieses da ist, wie die BGP Daten entstehen und wie das Internetrouting unter Verwendung von BGP funktioniert. Anschließend zeigt der Abschnitt 2.1.2 was Routingkonflikte sind, und warum sie häufig auftreten und wie diese von der Software behandelt werden. Der Abschnitt 2.2 über Datenbeschaffung stellt die Quellen dar, aus denen die BGP Routingdaten bezogen werden können, wie sie dort gesammelt werden und in welchem Format sie erhältlich sind. Was GNS3 ist und wie es Funktioniert, welchen Funktionsumfang es bietet und welche Funktionen davon genutzt werden, wird in Abschnitt 2.3 kurz erläutert. Der Aufbau der Konfigurationsdateien für GNS3 und Cisco Router, die von der Software selbstständig anhand der eingelesenen Daten erstellt werden, wird in Abschnitt 2.3.1 erläutert.

2.1 Wie Internet Routing funktioniert

Das Internet besteht aus einer wachsenden Anzahl unabhängiger Netze, den Autonomen Systemen (AS), von denen zur Zeit etwa 110000 bei den dafür zuständigen registrierungs Instanzen registriert, von denen jedes durch eine eindeutige Indentifikationsnummer beschrieben ist [1]. Wie ein Autonomes System intern organisiert ist lässt sich nicht einheitlich beschreiben, da dies bei jedem Internet Service Provider (ISP) oder Unternehmen, das ein AS betreibt unterschiedlich sein kann. So kann ein Autonomes System sich auch aus mehreren kleineren Teilnetzen zusammensetzen die einer administrativen Instanz unterstehen und deshalb mit einer Nummer gemeinsam registriert sind. [16]

Unabhängig davon ob das Ziel innerhalb des Netzes liegt oder die Daten über ein Router an der Außengrenze des Netzes an ein anderes Netz geleitet werden sollen. Untereinander aber verfolgen die meisten AS Betreiber ein ähnliches Ziel, ihre eigenen Systeme möglichst redundant mit dem Internet zu verknüpfen. Kleine Systeme müssen größeren Geld zahlen für ihre hochgeladenen Daten, große Systeme mit Kontinent umspannenden Netzen hingegen zahlen nichts dafür ihre Daten an kleinere weiterzuleiten. Die Systeme werden in der Regel in Tier 1, 2 und 3 eingeteilt, wobei Tier 3 Provider ihren Datenverkehr an keine kleineren Provider weitergeben sondern nur an Endkunden. Tier 1 Provider hingegen bedienen in der Regel keine Endkunden sondern peeren nur mit anderen großen Provider. [1]

Um die kürzesten, schnellsten oder günstigsten Routen zu jeder möglichen angewählten IP Adresse auszuwählen müssen die Router der verschiedenen Autonomen Systeme untereinander Daten über ihre Verbindungen austauschen. [16]

2.1.1 BGP (Border Gateway Protocol)

Das Border Gateway Protocol (BGP) [17] ist der Standard unter den Routing Protokollen mit denen die Router der unterschiedlichen Autonomen Systemen direkt miteinander oder über Austauschpunkte wie DE-CIX in Frankfurt kommunizieren. Dabei bedeutet kommunizieren über einen Austauschpunkt nur, dass die direkte verbindung zweier Router in einem großen Knotenpunkt besteht, trotzdem sind die Router jeweils direkt miteinander Verbunden. Dabei schickt jeder Router jedem benachbarten Informationen darüber welche IP Bereiche zum eigenen Autonomen System gehören um die zugehörigen Daten an sich selbst leiten zu lassen. Außerdem schickt er Daten zu allen bekannten Routen, die jeweils zu spezifischen IP Bereichen führen, an die Nachbarn bei denen er jede Route mindestens einmal um sich selbst erweitert, sich selbst aber auch mehrmals hinzufügen kann um die Wahrscheinlichkeit zu senken, dass die Datan über das eigene Autonome System laufen, da die Route dort länger erscheint. Dadurch bekommt jeder Router eine Tabelle mit kompletten Pfaden von sich selbst zu jeder möglichen Zieladresse und kann die günstigste Route auswählen. Nach dem Austausch der kompletten Routingtabelle werden nur noch Änderungen weitergegeben um das Datenaufkommen zu reduzieren. Zusätzlich muss jeder Router seine Nachbarn regelmäßig darüber informieren, dass er noch funktionstüchtig ist. Sollte diese Information ausbleiben wird der Router der keine Nachrichten mehr empfangen hat den nicht sendenden Router und alle Routen die über ihn führen aus seiner Routingtabelle streichen, und diese Änderung auch an seine weiteren Nachbarn weiterleiten. Diese Kommunikation ist ungesichert, sodass jeder der physischen Zugang zu einer Leitung zwischen Routern oder Zugang zu einem Router hat beliebige BGP Daten einspeisen kann. [16]

Der günstigste Weg für ein bestimmtes Datenpaket ist nicht unbedingt der kürzeste, sondern der der dem Provider die geringsten Kosten verursacht, denn wenn möglich wird er ein weg über ein kostenfreies Peering vorgeben bevor die Daten über ein kostenpflichtigen Uplink zu einem höheren Tier Provider gegeben werden. Ob und wie Provider in das Routing eingreifen ist schwer nachzuvollziehen und unterliegt in der Regel dem Geschäftsgeheimnis der einzelnen Firmen. [1] Es gibt aber die Aufwändige Methode die Routen per Traceroute zu überprüfen, stimmen sie mit den theoretisch kürzesten überein greifen die provider nicht in das Routing ein.

2.1.2 Routingkonflikte

Konflikte im Routing treten häufig auf und sind in vielen Fällen, jedoch nicht immer, beabsichtigt. So ist in der Arbeit *About Prefix Hijacking in the Internet* [2] beschrieben, wie die für ein Ziel bestimmten Daten durch annoncieren einer kürzeren Route zum Ziel einfach umgelenkt werden, da im BGP Routing nicht überprüft wird, ob das entsprechende Annoncement legitim, ein Versehen oder ein Angriff ist. So kann jeder, der einen Router kontrolliert, der an einem Übergang zwischen Autonomen Systemen plaziert ist, und als BGP Router bei den benachbarten Autonomen Systemen registriert ist, einfach beliebige Daten zu sich selbst oder einem anderen nahen Ziel umleiten, solange er eine Route annonciert, die kürzer ist als die ursprüngliche. Wenn der Angreifer anschließend die Daten an den ursprünglichen Empfänger weiterleiten kann, fällt die Man in the middle Attacke in den meisten Fällen nicht auf, wenn nicht explizit danach gesucht wird. Das dies aufgrund von Sicherheitslücken möglich ist, haben auf der Def Con 22 die beiden Sicherheitsforscher Luca Bruno und Mariano Graziano gezeigt wie *heise security* berichtete [13]. Auch durch Fehlkonfigurationen durch Unachtsamkeit oder Zahlendreher bei der manuellen Konfiguration entstehen viele Fehler, bei denen falsche Präfixe oder falsche AS Nummern annonciert werden.

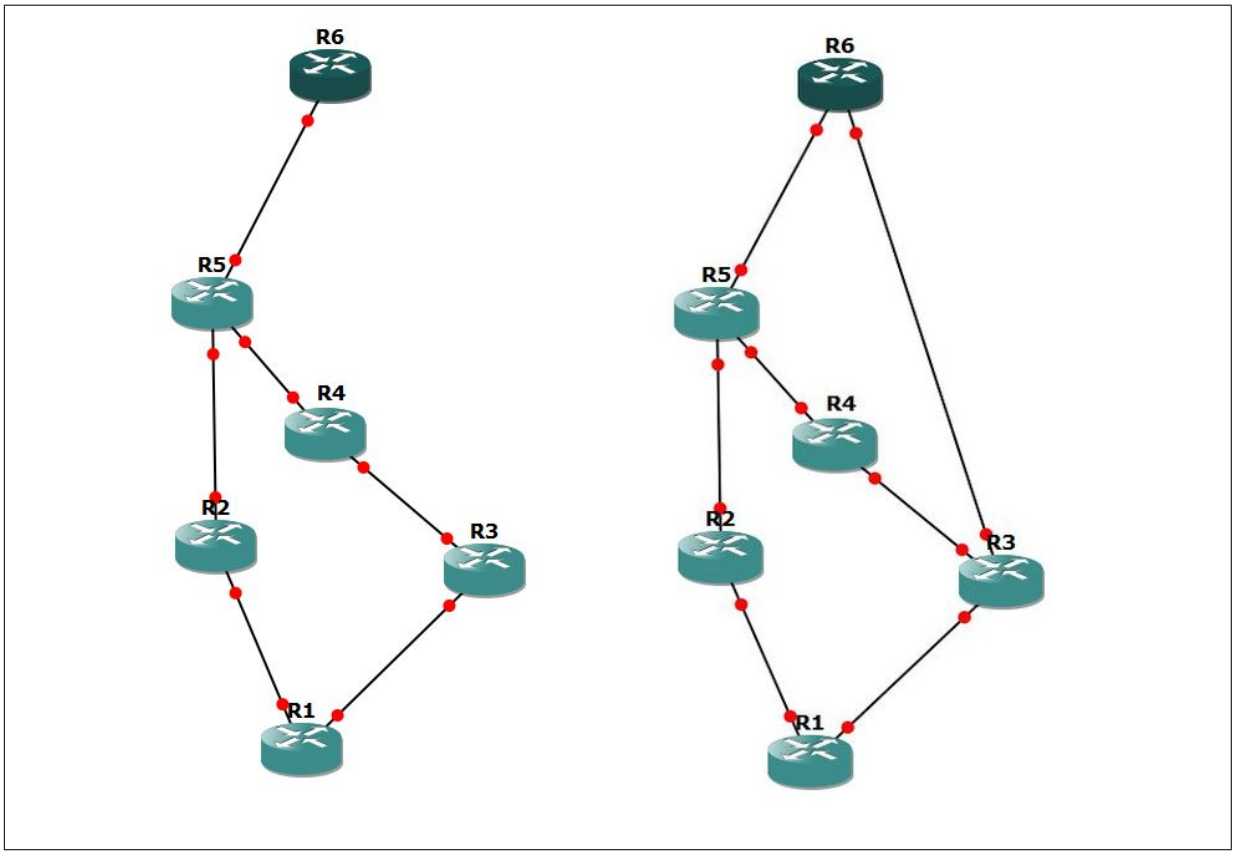

Abbildung 2: Beispiele für Routenänderungen

Außerdem gibt es sehr viele mit Absicht herbeigeführte Fehler im Routing, die meistens *Multiple Origin Autonom System Conflicts* (MOAS Conflicts) sind. Dies bedeutet, dass es zu einem Präfix mehrere Zielsysteme bekannt sind, zu denen die Daten geleitet werden sollen. Oft sind diese legitim, weil sie zum Einen zum load balancing dienen und zum Anderen durch weiterreichen von Subnetzen vom Provider zum Customer entstehen. Solche MOAS Konflikte sind in der Arbeit *Automatic Analysis and Classification of Multiple Origin AS (MOAS) Conflicts* [3] beschrieben und in der zugehörigen Software analysiert worden, wobei sich zeigte, dass ständig etwa 60000 Konflikte stabil vorhanden sind und sich allein im ersten Quartal 2010 etwa 120000 weitere Konflikte für jeweils einen Teilzeitraum zeigte.

Konflikte wie die hier aufgezählten werden häufig beschrieben, etwa in Inter-AS Routing Anomalies: Improved Detection and Classification [15].

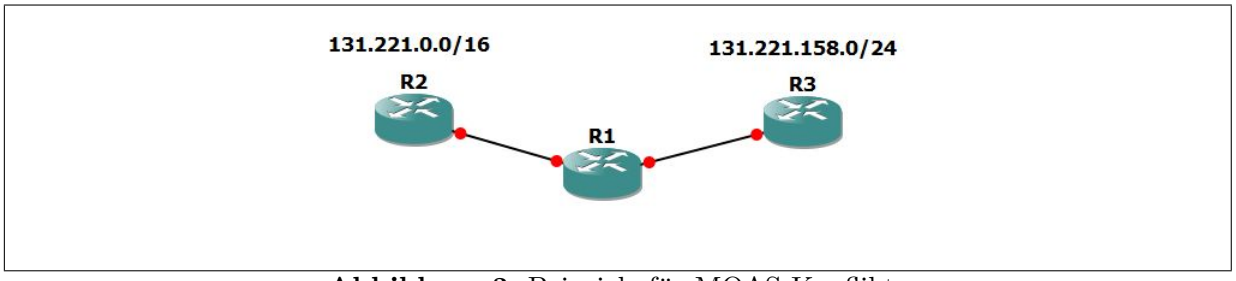

Abbildung 3: Beispiele für MOAS Konflikte

2.2 Datenbeschaffung

Die Daten über das Routing im Internet sind weit verteilt, jeder Router der im globalen Routing teilnimmt hat seine eigene Routingtabelle. Es gibt einige Projekte die diese Daten sammeln, dazu gehört der Routing Information Service (RIS) des Rèseaux IP Européens Network Coordination Centre (RIPE), einer der 5 Regional Internet Registries (RIR) [12]. Dort werden von Route Reflector Clients (RRC) Routingdaten von jeweils mehreren Viewpoints gesammelt, wobei jeder Viewpoint ein Router repräsentiert, die sich alle in einem Internet Exchange Point (IXP) befinden. Bei diesem Projekt wird alle 8 Stunden ein kommplettes Abbild der Routingtabelle aller an dem Projekt beteiligten Router angefertigt, die die Viewpoints in den Daten darstellen. Zusätzlich gibt es alle 5 Minuten eine Datei die alle Updates der letzten 5 Minuten beinhaltet. In ausnahmefällen kann diese Zeitspanne auch länger oder kürzer sein. Diese Dateien, die im Zebra eigenen Format [14] abgelegt und zusätzlich komprimiert werden, sind nach etwa 2 Stunden fertig und auf die Server des Projekts hochgeladen, sodass jeder der eigene Projekte auf diese Daten stützt nie in Echtzeit arbeiten kann sondern immer diese zeit zurückliegt. Die 2 Stunden bis zur verfügbarkeit der Daten basieren auf beobachtungen und sind nur ein grober richtwert auf den man sich nicht verlassen kann. Diese Daten sind umfangreich und beinhalten einen genauen Zeitstempel, Angaben von welchem Router sie stammen, den genauen Pfad zum Ziel, den IP Bereich der mit diesem Datensatz annonciert werden soll und weitere für diese Betrachtung irrelevante.

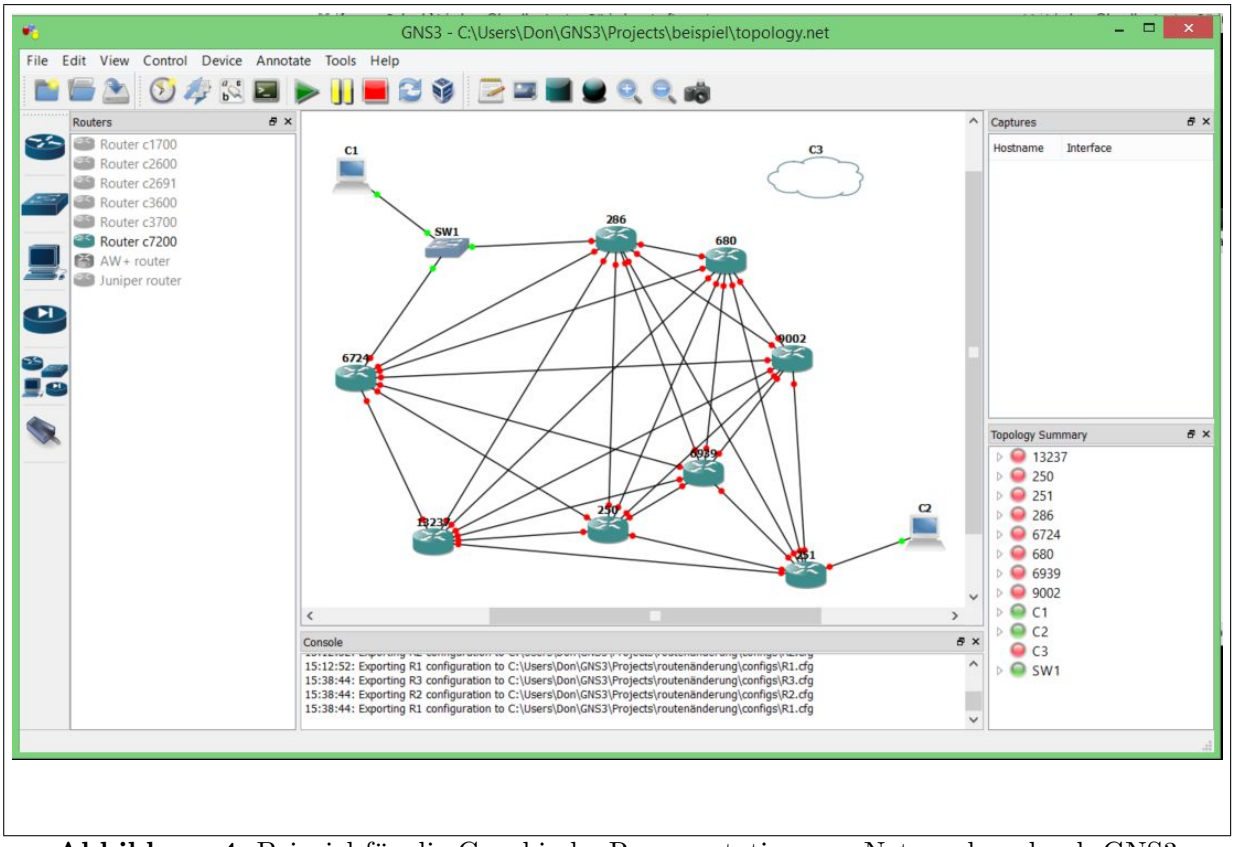

Abbildung 4: Beispiel für die Graphische Representation von Netzwerken durch GNS3

2.3 Graphical Network Simulator 3 (GNS3)

Die Simulation der Routing Situationen wird vom Graphical Network Simulator 3 (GNS3) durchgeführt [8]. Diese Software emuliert die Hardware und stellt diese mithilfe eines leicht verständlichen Graphischen User Interface übersichtlich und strukturiert dar, wie in Abbildung 4 gezeigt. Hierbei kann die Software mittels Dynamps die gängigsten Cisco Router emulieren. Cisco Router werden von Dynamips emuliert, das die meisten gebräuchlichen Cisco Router emulieren kann. Da die Software vom ursprünglichen Entwickler nicht weiter unterstützt wird hat das Team um GNS3 selbst einige Erweiterungen geschrieben. Zur Simulation von Desktop und Server Systemen wird Oracles VirtualBox [10] genutzt. Weitere Geräte können mit Hilfe von Qemu emuliert werden.

2.3.1 Aufbau Konfigurationsdateien

Für das Verständins der Software ist es weiterhin wichtig zu verstehen wie in GNS3 Konfigurationsdateien aussehen. Zum Einen muss der Simulator selbst konfiguriert werden, der die Simulation graphisch darstellt und die Verbindungen zwischen einzelnen emulierten Routern, Knotenpunkten und Rechnern konfiguriert und verarbeitet. Die notwendigen Schritte sowie die Darstellung in den Konfigurationsdateien werden im nächsten Abschnitt 2.3.2 weiter erläutert. Da es keine Dokumentation gibt, die beschreibt wie Konfigurationsdateien für GNS3 aufgebaut sind wurde dies für diese Arbeit aus Beispielen extrahiert. Zum Anderen muss jeder Router, der emuliert wird, beim Start seine Konfiguration übergeben bekommen, die die zu simulierende Routingstruktur wiederspiegelt, wie sie aus den zugrundeliegenden Daten hervorgeht. Die Konfiguration der Router wird im Abschnitt 2.3.3 weiter erklärt.

2.3.2 Konfiguration von GNS3

Alle Angaben im folgenden Abschnitt wurden mangels Dokumentation durch Experimente bestimmt. Die Konfiguration einer Simulation in GNS3 [8] erfolgt in mehreren Teilen. Als erstes liegt im Hauptverzeichnis des Projekts eine Datei, die immer als *topology.net* benannt ist, in der nicht nur Angaben zur Topologie, sondern auch zu jedem Knoten und jeder Kante der Simulation, die Konfiguration oder der relative Dateipfad zur Konfigurationsdatei hinterlegt ist. Diese Konfigurationsdatei beginnt immer mit den Angaben, ob die Simulation beim Laden der Projektdatei automatisch starten soll und mit welcher Version von GNS3 diese angelegt wurde wie es im Listing 2 zu sehen ist.

```
autostart = False
version = 0.8.6
```

Listing 2: Autostart Option

Daraufhin werden alle emulierten Geräte unter einem Punkt zusammengefasst, die über einen Netzwerkport auf dem simulierenden System ansprechbar sind, wie es in Listing 3 in einem Beispiel für nur ein Gerät zu sehen ist. Dies ist immer gefolgt von Angaben zum Arbeitsverzeichnis, Ports und Images von emulierten Geräteklassen. Dabei ist ein Port in diesem Zusammenhang ein virtuelles Äquivalent zu einer RJ45 Buchse, die über eine simulierte Verbindung mit einer anderen Buchse, also einem anderen Port, verbunden werden kann. Ein Image beinhaltet ein

digitales Abbild der Firmware, mit der das jeweilige Gerät betrieben wird. Außerdem werden zu jeder Klasse Angaben gemacht, wie diese emuliert werden, wie lange sie in den idle Zustand gehen sollen, wie ihr Speicher verwaltet wird und ob sie ein vollständiges eigenes Image ihres Systems haben oder nur eine inkrementelle Kopie. Eine Kopie hat für die genutzten Funktionen keine Nachteile und wird deshalb immer von der Software genutzt, um Speicherplatz zu sparen, den man bräuchte, um für jeden emulierten Router die Firmware separat vorzuhalten.

```
[127.0.0.1:7202]
        workingdir = working
        udp = 10200
        [[7200]]
                image = /gns3/images/exampleimage7200.bin
                idlepc = 0x6099bf90
                sparsemem = True
                ghostios = True
```

Listing 3: Firmwareimage für Router definieren

Daraufhin werden,wie in Listing 4 zu sehen ist, die Geräte selbst beschrieben, für die jeweils Parameter angegeben werden, wie sie anzusprechen sind und wo eventuelle Konfigurationsdateien liegen. Dann muss konfiguriert werden, welche Zusatzkarten in den emulierten Slots der Router installiert sind. Bei Routern schließt sich daran die Konfiguration der Netzwerkverbindungen an und mit welchem Netzwerkport eines anderen Geräts diese verbunden sind. Die Ports, die hier konfiguriert werden, sind die oben beschriebenen, die jeweils eine Ethernet Verbindung zu einem anderen Port darstellen. Zuletzt steht dann, an welchen Koordinaten ein Gerät in der graphischen Repräsentation der Simulation dargestellt werden soll.

```
        [[ROUTER R1]]
                console = 2011
                aux = 2511
                cnfg = configs/R1.cfg
                slot1 = PA-2FE-TX
                f1/0 = R2 f1/1
                f1/1 = SW1 3
                x = 5.0
                y = 5.0
                z = 1.0
```

Listing 4: Konfiguration der Knotenpunkte, ihre Verbindungen untereinander sowie der Positionen der graphischen Repräsentationen

Bei Netzwerkswitches kommt die Konfiguration ohne zusätzliche Dateien aus, da die Verbindungen von ihnen direkt von GNS3 übernommen werden und die Daten zwischen den verbundenen Geräten direkt verteilt werden. Deshalb muss nur, wie in Listing 5 gezeigt, angegeben werden, was an welchen Port angeschlossen ist und wo der Switch dargestellt werden soll.

```
[[ETHSW SW 1]]
        1 = access 1 R1 f1/1
        2 = access 1 R3 f2/0
        3 = ...
        x = 10.0
        y = 10.0
        z = 1.0
```

Listing 5: Konfiguration von Switches innerhalb von GNS3

Zuletzt in der Konfigurationsdatei *topology.net* wird noch angegeben, in welchem relativen Verzeichnis GNS3 das Arbeitsverzeichnis sowie die Konfigurationsdateien findet oder ablegen soll, wenn neue erstellt werden, Änderungen gespeichert werden oder diese eingelesen werden sollen, was im Folgenden Listing 6 gezeigt ist.

```
[GNS3-DATA]
        configs = configs
        workdir = working
```

Listing 6: Arbeitsverzeichnisse von GNS3 konfigurieren

2.3.3 Konfiguration von Cisco Routern

Die Konfigurationsdateien für Cisco Router bestehen aus einer Reihe von Befehlen, die auch manuell an der Konsole eingegeben werden könnten, jedoch beim Start aus einer Textdatei eingelesen werden. Dazu wird mit GNS3 eine Basiskonfiguration mitgeliefert, die in Listing 7 zu sehen ist, die dann an erweitert werden muss, um eine Routingumgebung aus realen Daten darstellen zu können. Die Grundeinstellungen sind nicht entscheidend für dieses Projekt und werden daher nicht näher erläutert. Die genauen Bedeutungen können den Handbüchern der Cisco Router der Serie 7200 entnommen werden. [7]. Im Weiteren wird nur auf die Optionen näher eingegangen, die von der Software in Abhängigkeit der eingelesenen Routingdaten gesetzt werden.

```
hostname %h
!
no ip domain lookup
no ip icmp rate-limit unreachable
ip tcp synwait 5
!
line con 0
 exec-timeout 0 0
 logging synchronous
 privilege level 15
 no login
line aux 0
 exec-timeout 0 0
 logging synchronous
 privilege level 15
 no login
!
end
```

Listing 7: Basiskonfiguration eines Cisco 7200 Routers

Die Basiskonfiguration die in Listing 7 zu sehen ist zeigt die Systax die eine Konfigurationsdatei für ein Cisco Router besitzen muss. Dabei wird ab einer Einrückung um ein Leerzeichen die nächste Ebene der Konfiguration angesprochen, die durch den letzten nicht eingerückten Befehl vorher geöffnet wird. Ein Ausrufezeichen springt jeweils eine Ebene zurück, und schließt die vorher geöffnete Ebene wieder.

Jedes Ethernet Interface braucht eine IP Adresse, die es den Nachbarn ermöglicht, es anzusprechen. Die Konfiguration geschieht auch über die jeweiligen Befehle, die zu Beginn aus der Konfigurationsdatei geladen werden. Ein Beispiel für eine solche Konfiguration ist im folgenden Listing 8 dargestellt.

```
interface Ethernet4/0
 ip address 10.42.13.1 255.255.255.0
 negotiation auto
 !
!
interface Ethernet4/1
 ip address 10.42.13.2 255.255.255.0
 negotiation auto
 !
!
interface Ethernet4/2
 ip address 10.42.13.3 255.255.255.0
 negotiation auto
 !
!
```

Listing 8: Konfiguration der Interfaces eines Routers

Die dargestellte Konfiguration stellt für das in den Slot 4 virtuell eingesteckte Fast Ethernet Modul PA-8E, mit 8 100MBit Ethernet Ports, die IP Adressen 10.42.13.1-3 zur Verfügung, die jeweils mit einer Netzwerkmaske versehen sind. Außerdem wird festgelegt, dass die Geschwindigkeit der Verbindung durch die im Betrieb verbundenen Geräte automatisch festgelegt wird.

Als nächstes wird dem Router die Konfiguration der internen BGP Funktionalität übergeben wie in Listing 9 zu sehen, indem erst mit folgenden Befehlen festgelegt wird wie das BGP Routing funktioniert und zu welchem Netzwerk und AS der Router selbst gehört. Diese Angaben sind für das interne Routing eines Autonomen Systems und werden nicht nach außen propagiert. Zusätzlich zu dem Netzwerk, zu dem der Router gehört, können auch andere IP Bereiche angegeben werden, wenn dieser Router mehrere annoncieren soll. Weiterhin kann in diesem Bereich statisches Routing festgelegt werden, wie es im folgenden Beispiel gezeigt wird. Da für diese Arbeit das interne Routing nicht betrachtet wird, wird dies hier nicht näher erläutert, ist jedoch in den Handbüchern zu den Routern nachzulesen. [7]

```
router rip
 redistribute bgp 4711 metric 2
 network 42.08.15.0
 !
ip forward-protocol nd
 ip route 10.98.0.0 255.255.255.0 10.32.32.1
 !
!
```

Listing 9: Konfiguration der eigenen Announcements und von festen Routen

Zur Konfiguration eines Routers als Teilnehmer am Border Gateway Protocol (BGP) [17] sind die im folgenden Listing 10 gezeigten Schritte notwendig, teilweise in mehrfacher Ausführung, wenn etwa mehr als ein IP Bereich annonciert werden soll. Zuerst wird der Router in den BGP Konfigurationsmodus versetzt und dabei festgelegt zu welchem Autonomen System [5] er gerechnet wird. Dann wird festgelegt, welche Präfixe annonciert werden sollen, indem die Adresse mit der Angabe der Netzwerkmaske angegeben wird. Bei IPv6 Adressen wird statt der Netzwerkmaske die Prefixlänge angegeben. So werden in dem unten stehenden Beispiel die Präfixe *10.0.0.0/8*, *11.128.0.0/16* und *200:100::/32* annonciert werden. [6] Zusätzlich wird noch angegeben, welches benachbarte AS unter welcher IP-Adresse erreichbar ist, um festzulegen mit wem BGP Nachrichten über welchen Pfad ausgetauscht werden können.

```
router bgp 4711
network 10.0.0.0 mask 255.0.0.0
network 11.128.0.0 mask 255.255.0.0
network 200:100::/32
neighbor 15.1.2.2 remote-as 815
neighbor 22.111.42.2 remote-as 1113
```

Listing 10: BGP Konfiguration der eigenen Announcements und der bekannten Nachbarn

2.4 Bibliothek zur Verarbeitung von BGP Daten, libbgpdump

Die Bibliothek *libbgpdump* [4], die von *RIPE* gewartet wird und über ris.ripe.net [12] erhältlich ist, stellt ein Interface zur Verfügung um die binären Daten im Zebra Format [14] in eine Struktur zu bringen die in C und C++ Programmen einfach verarbeitet werden kann. Dazu stellt es die im folgenden erklärten Funktionen bereit, die im Listing 11 aufgeführt sind.

```
\begin{center}
BGPDUMP *bgpdump_open_dump(const char *filename);
void     bgpdump_close_dump(BGPDUMP *dump);
BGPDUMP_ENTRY*  bgpdump_read_next(BGPDUMP *dump);
void     bgpdump_free_mem(BGPDUMP_ENTRY *entry);
char *bgpdump_version(void);
\end{center}
```

Listing 11: Public Funktionen der Bibliothek bgpdump

Die Funktion bgpdump_open_dump öffnet eine kopimierte Binärdatei die BGP Daten im entsprechenden Format ethält, während bgpdump_close_dump diese wieder schließt und den verwendeten Speicher freigibt. bgpdump_read_next gibt den nächsten Eintrag in der Datei zurück, dessen Speicherplatz mit bgpdump_free_mem wieder freigegeben werden kann. Zur Versionskontrolle gibt es zusätzlich noch die Funktion bgpdump_verison.

3 Software

Die für diese Bachelorarbeit implementierte Software Automatische Routing Daten Analyse (ARDA) zum automatischen Erstellen von Konfigurationsdateien zur Simulation von realen Internet Routing Situationen mit GNS3 ist so aufgebaut, dass die Software, mit Angabe der benötigten Dateien, über die Kommandozeile gestartet wird. Daraufhin werden die Dateien der Reihe nach geöffnet, wobei die Reihenfolge durch den Zeitstempel im Namen der Dateien festgelegt ist. Der Inhalt jeder Datei wird durch die libbgpdump geparst und Eintrag für Eintrag an die Software übergeben. Diese überprüft die gelieferten Daten darauf, dass sie durch Einschränkungen von GNS3 und der Definition der Aufgabe vorgegebene Grenzwerte nicht überschreiten. Dann wird im Speicher eine Datenstruktur angelegt, die über Pointer zwischen Objekten die Kanten zwischen Routern nachbilden. Anschließend wird der Bereich, der simuliert werden soll, festgelegt anhand der auf der Kommandozeile definierten Bereiche. Alle nicht im Simulationsbereich befindlichen Autonomen Systeme werden entfernt und die von ihnen annoncierten Präfixe so gut wie möglich von möglichst wenigen zusätzlichen Routern übernommen. Zum Schluss werden aus den vorliegenden Daten die Konfigurationsdateien erzeugt und diese in die für GNS3 passende Ordnerstruktur verschoben. Damit ist die Aufgabe der Software abgeschlossen und das angelegte Projekt kann mit GNS3 geöffnet werden.

3.1 Softwaredesign

Die zu dieser Bachelorarbeit gehörende Software ARDA zum Einlesen, Verarbeiten von BGP Routing Daten und der Ausgabe von Konfigurationsdateien zur Simulation der eingelesenen Internet Routing Situationen ist in C++ geschrieben um das Design objektorientiert zu gestalten und die Erweiterbarkeit und Wiederverwendbarkeit der einzelnen Komponenten zu steigern. Aus Erfahrungen mit BART [9] konnte ein Teil der Funktionen zum Parsen der Eingabedaten im Zebra [14] Datenformat übernommen und an die Anforderungen dieser Software angepasst werden, um den Anforderungen gerecht zu werden.

3.1.1 Softwarefunktion

Die Software ARDA ist wie in Abbildung 5 gezeigt aus einigen Objekten zusammengesetzt, die im Folgenden beschrieben werden.

Ausgehend von der main.cpp, in der die in jedem C++ Programm als Einsprungspunkt benötigten *main* Funktion steht, wird die Software gestartet. Wie die Software ausgeführt werden sollte um volle Funktionalität zu gewährleisten, ist im Abschnitt 3.3 ausführlich beschrieben. In dieser Datei liegt außerdem die Funktionalität zur Auswertung der Kommandozeilen-Parameter. Wenn diese erkannt und darauf überprüft wurden, dass sie zumindest theoretisch korrekt sein, könnten wird das Parsen der übergebenen Dateien gestartet. Theoretisch korrekt bedeutet, dass die zusammen angegebenen Parameter mit einer denkbaren Kombination an Eingabedaten ein valides Ergebnis erzeugen könnten.

Wie in Abbildung 5 zu erkennen ist, wird dann für jede Datei, in der Reihenfolge des Alters der Daten, in der Klasse *input* die Funktion *readData* aufgerufen. Dort wird die Datei mit der libbgpdump, die in Abschnitt 2.4 der Grundlagen beschrieben wird, geöffnet und das Parsen der Daten begonnen.

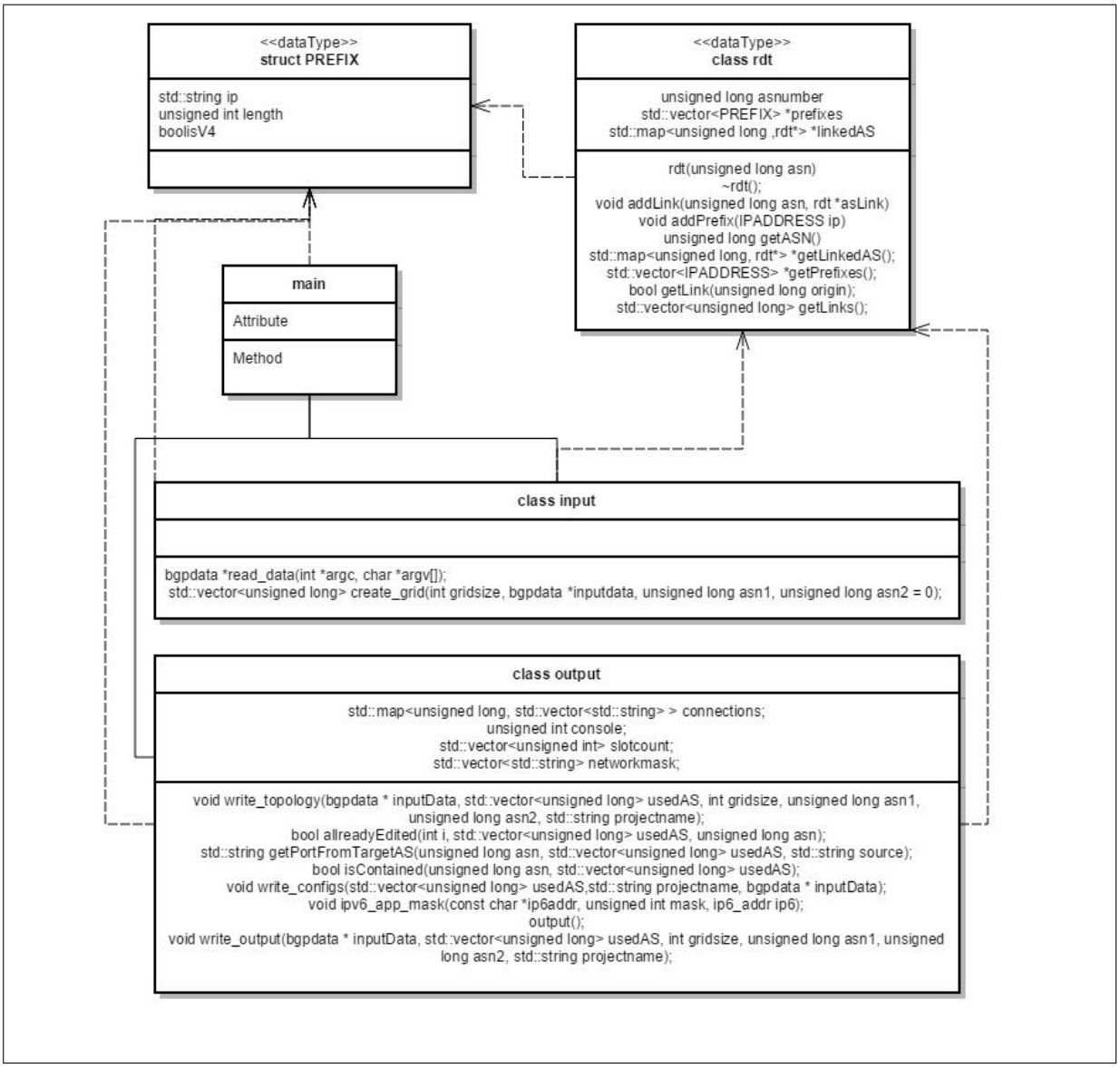

Abbildung 5: Klassendiagramm der Software

Dafür wird Zeile für Zeile ausgelesen und jeweils an die entsprechende Funktion der Klasse *bgpdata* übergeben, wo die Daten einer groben Kontrolle auf Fehleingaben und fehlende Angaben unterzogen werden. Ist diese Prüfung erfolgreich beendet, werden die Daten in das Dateiformat *rdt*, das im Abschnitt 3.1.2 beschrieben ist, überführt. Dabei werden die im Kapitel Grundlagen 2.1.2 beschriebenen MOAS Konflikte bewusst ignoriert, da diese auch in dem reellen Routing im Internet vorkommen.

Die so erstellte Struktur aus *rdt* Objekten, die alle für die Simulation wichtigen Daten aus den übergebenen Routingdaten enthält, wird dann an die Klasse *output* übergeben, in der die Daten anhand der Kommandozeilen-Parameter, die beim Aufruf der Software angegeben wurden, reduziert werden. Dafür wird kontrolliert, ob die angegebenen Autonomen Systeme in den Daten existieren und das Programm mit einer Fehlermeldung beendet, wenn dies nicht der Fall ist. Sind die Autonomen Systeme in den Daten vorhanden, werden alle Knotenpunkte übernommen, die die angegebene Anzahl der Hops zu den zu betrachtenden Autonomen Sys-

temen nicht überschreiten. Die Autonomen Systeme und die von ihnen Annoncierten Prefixe werden ab hier ignoriert, da sie für die Simulation unerheblich sind und die Simulation von allen Prefixen zu viel Rechenaufwand bedeutet.

So entsteht eine Struktur, die der Struktur in der späteren schon Simulation weitestgehend entspricht. Diese Struktur wird dann an die nächste Funktion in der Klasse *output* übergeben, in der die Konfigurationsdateien für GNS3 und die Cisco 7200 Router erstellt wird. Zuerst wird die Projektstruktur selbst angelegt, unter dem über die Kommandozeile angegebenen Namen, und mit der Struktur die von GNS3 benötigt wird.

Dann wird jeder Verbindung ein Wert zugewiesen, der der Darstellung der Verbindungen in GNS3 Konfigurationsdateien entspricht, wie in Abschnitt 2.3.1 im Kapitel Grundlagen beschrieben. Aus den dann vorliegenden Daten wird dann die GNS3 Konfiguration erstellt, bei der jeder Knotenpunkt in einem gleichmäßigen Raster verteilt wird, siehe dazu auch den Abschnitt 5.2. Zuletzt wird für jedes Autonome System eine Konfigurationsdatei für ein Cisco 7200 Router erstellt, der das jeweilige System repräsentiert. Alle Dateien werden in der anfangs erstellten Ordnerstruktur zusammengefasst.

3.1.2 Datenformat

```
class rdt{
protected:
        unsigned long asnumber;
        std::vector<PREFIX> *prefixes;
        std::map<unsigned long,rdt*> *linkedAS;
public:
        rdt(unsigned long asn);
        ~rdt();
        void addLink(unsigned long asn, rdt *aslink);
        void addPrefix(PREFIX prefixip);
        unsigned long getASN();
        std::map<unsigned long, rdt*> *getLinkedAS();
        std::vector<PREFIX> *getPrefixes();
        bool getLink(unsigned long origin);
        std::vector<unsigned long> getLinks();
};
```

Listing 12: Datenformat rdt (Routing Daten Transport)

Das Datenformat *rdt*, wie auf im Listing 12 zu erkennen, beinhaltet jeweils die Daten, die ein Autonomes System betreffen. Dazu gehört in der Variable *unsigned long asnumber* die eindeutige Identifikationsnummer des jeweiligen Autonomen Systems, das beschrieben werden soll. Weiter stehen in dem Vector *std::vector<PREFIX> *prefixes* eine Liste aus Präfixen, die von dem beschriebenen Autonomen System annonciert wurden. Außerdem befindet sich in der Variable *std::map<unsigned long,rdt*> *linkedAS* eine nach AS Nummer sortierte Liste von Pointern, die auf die *rdt* Daten von mit diesem Autonomen System verlinkten anderen Autonomen Systeme zeigen. Dadurch werden die Kanten zwischen den einzelnen Autonomen Systemen dargestellt. Weiterhin besitzt *rdt* Funktionen zum Hinzufügen von Präfixen, AS Nummern und Links sowie zum Abfragen dieser Werte. Zusätzlich gibt es eine Funktion zum Abfragen, ob ein Link schon

existiert, um das Einfügen von bereits existierenden Links zu verhindern und dadurch den Prozess zu beschleunigen. Der Konstruktor initialisiert die genutzten globalen Variablen und setzt die AS Nummer bei der Initialisierung eines *rdt* Objekts. Dahingegen gibt der Dekonstruktor den reservierten Speicher wieder frei.

Ein weiteres von dieser Software genutztes Datenformat sind die Konfigurationsdateien für GNS3 und die Cisco 7200 Router, die im Abschnitt 2.3.1 im Kapitel Grundlagen beschrieben werden.

3.1.3 Softwareaufbau

Nach einem Überblick in Abschnitt 3.1.1, was die Software in groben Zügen in welcher Reihenfolge erledigt und in Abschnitt 3.1.2 die Erläuterung der genutzten Datenformate, folgt nun eine Erklärung der Funktionen im Einzelnen. Dazu wird eine Datei nach der anderen durchgegangen und jeweils eine Klasse nach der anderen erklärt.

main.cpp

Die main.cpp enthält die Main Funktion von ARDA die bei einem Start als erstes aufgerufen wird. Hier werden die Kommandozeilen Parameter ausgelesen und überprüft. Wenn diese den vorgegebenen Grenzen entsprechen werden zuerst alle angegebenen Daten geparst und anschließend due Konfigurationsdateien erstellt.

main.h

```
struct PREFIX{
        std::string ip;
        unsigned int length;
        bool isV4;
};
```

Listing 13: main.h

In der *main.h*, die in Listing 13 dargestellt ist, wird neben einigen includes die hier nicht dargestellt sind, das struct *PREFIX* definiert, welches eine IP Adresse speichert. Weiterhin enthält es Platz zur Angabe der Länge des Präfix und ob die verwendete Adresse IP V4 oder V6 ist.

input

```
class input{
protected:

public:
        bgpdata *read_data(int *argc, char *argv[]);
        std::vector<unsigned long> create_grid
                (int gridsize, bgpdata *inputdata,
                unsigned long asn1, unsigned long asn2 = 0);
};
```

Listing 14: class input

Die in Listing 14 dargestellte Klasse *input* enthält die Funktion readData, die eine ihr übergebene Datei mit Routingdaten öffnet und je nachdem, ob es eine Datei mit Fulltable oder Update Daten ist, gibt sie Zeile für Zeile an die Klasse *bgpdata* weiter, die im Folgenden noch beschrieben wird. Außerdem enthält sie die Funktion create_grid, die aus allen geparsten Daten die auswählt, die für die spätere Simulation notwendig sind.

bgpdata

```
class bgpdata{
protected:
        std::map<unsigned long,rdt*> *linkedAS;
        std::vector<unsigned long> *createVectorLong(std::string *aspath);
public:
        bgpdata();
        ~bgpdata();
        bool insertbview(BGPDUMP_ENTRY *entry);
        bool insertupdate(BGPDUMP_ENTRY *entry);
        bool checkASN(unsigned long asn);
        unsigned int getMapSize();
        std::vector<unsigned long> getLinkedASN(unsigned long asn);
        std::vector<PREFIX> *getPrefixes(unsigned long asn);
};
```

Listing 15: class bgpdata

In Listing 15 ist die Klassendefinition von *bgpdata* dargestellt, die die Daten eines Eintrags der Daten von ris.ripe [12] in die Datenstruktur *rdt* einließt, die in Abschnitt 3.1.2 erläutert ist. Im protected Teil der Klassendefinition ist der Pointer auf die map *linkedAS* zu erkennen, in der die geparsten Daten gespeichert werden, die für die spätere Simulation relevant sind. Außerdem findet sich dort die Funktion createVectorLong, die aus dem String eines Pfades aus Autonomen Systemen einen Vektor mit den gleichen Nummern in derselben Reihenfolge wie im Ursprungs String generiert und zurück gibt. Danach finden sich im public Teil der Definition Konstruktor und Destruktor, die für das Reservieren und das Freigeben des Speichers für die Datenstruktur *linkedAS* zuständig sind. Weiter sind dort die beiden Funktionen *insertbview* und *insertupdate* zu sehen, die jeweils einen Eintrag aus einer Fulltable- oder Updatedatei in die Datenstruktur einfügen, nachdem diese darauf geprüft wurden, ob sie den definierten Grenzwerten für die jeweiligen Datenfelder entsprechen, die sie füllen sollen. Dann ist die Funktion *checkASN* zu sehen die überprüft ob ein angegebenes Autonomes System in der Datenstruktur existiert Die Funktion i*getMapSize* gibt zurück wie viele Autonomen Systeme in der Datenstruktur vorhanden sind. *getLinkedASN* ruft eine Liste der Autonomen Systeme ab, die mit dem angegebenen Autonomen System peeren und damit verbunden sind. Zuletzt die Funktion *getPrefixes* die eine Liste aller IP Prefixe zurück gibt die ein angegebenes Autonomes System annonciert hat.

output

```
class output{
protected:
        void write_topology(bgpdata * inputData,
                std::vector<unsigned long> usedAS,
                int gridsize, unsigned long asn1, unsigned long asn2,
                std::string projectname);
        std::map<unsigned long, std::vector<std::string> > connections;
        unsigned int console;
        bool allreadyEdited(int i, std::vector<unsigned long> usedAS,
                unsigned long asn);
        std::string getPortFromTargetAS(unsigned long asn,
                std::vector<unsigned long> usedAS, std::string source);
        std::vector<unsigned int> slotcount;
        bool isContained(unsigned long asn,
                std::vector<unsigned long> usedAS);
        std::vector<std::string> networkmask;
        void write_configs(std::vector<unsigned long> usedAS,
                std::string projectname, bgpdata * inputData);
public:
        output();
        void write_output(bgpdata * inputData,
                std::vector<unsigned long> usedAS, int gridsize,
                unsigned long asn1, unsigned long asn2,
                std::string projectname);
};
```

Listing 16: class output

Die im Listing 16 gezeigte Klasse *output* erzeugt aus den Daten die von *input* erstellt wurden die Konfigurationsdateien für GNS3. Dazu wird die Funktion *write_output* aufgerufen, die die nötige Ordnerstruktur unter dem per Kommandozeile angegebenen Namen erzeugt. Vorher wird beim erstellen des Objekts der Konstruktor aufgerufen, in dem die Netzwerkmasken für Prefixlängen von /0 bis /32 als Strings in Vektor *networkmask* abgelegt werden. Dann wird die Erzeugung der Konfigurationsdatei für GNS3 angestoßen, indem *write_topology* aufgerufen wird. Zum Schluss wird *write_configs* aufgerufen, wo die Konfigurationsdateien für die einzelnen Router in der Simulation erstellt werden. Die Funktion *allreadyEdited* überprüft ob ein Autonomes System in der Konfigurationsdatei für GNS3 bereits angelegt wurde. *getPortFromTargetAS* Fügt ein String zu einem angegebenen Autonomen System hinzu der die Angabe eines Ports enthält und gibt den Port zurück an dem das Autonome System von dem die Funktion aufgerufen wurde verbunden ist. Zuletzt ist noch die Funktion *isContained* zu erwähnen, die überprüft ob ein Autonomes System in der Simulation enthalten ist.

3.2 Kompilieren

```
g++     -I./libbgpdump
        -o arda
        main.cpp rdt.cpp input.cpp bgpdata.cpp output.cpp
        -lboost_filesystem
        -lbgpdump
```

Listing 17: Befehl zum Kompilieren des Sourcecodes

Um die Software zu kompilieren, ist neben einem kompatiblen System der in Listing 17 gezeigte Befehl notwendig. Dazu muss die libbgpdump entpackt und installiert vorliegen, da sowohl auf die Bibliothek verlinkt wird als auch Definitionen aus ihr verwendet werden. Gegebenenfalls sind die Pfade anzupassen. Da sonst keine Abhängigkeiten bestehen, die nicht auf dem genannten System vorhanden waren, sind diese hier nicht aufgeführt, sollten aber bei Fehlen bei einem Kompilierungsversuch genannt werden, wenn sie nicht vorhanden sind.

3.3 Start

Der Aufruf der Software funktioniert über die Kommandozeile mit den im folgenden erläuterten Parametern, bei denen nur auf fehlende Parameter, die unbedingt zur Funktion benötigt sind sowie auf falsche Angaben, die in der Theorie unmöglich sind, geprüft wird, weshalb besondere Sorgfalt bei der Auswahl der Parameter wallten sollte. Die Reihenfolge der Parameter spielt hingegen keine Rolle, sollte aber zur Kontrolle der Angaben von einer gewissen Ordnung nicht abweichen.

Die Angabe der zu verarbeitenden Daten muss in der Rheienfolge geschehen wie diese verarbeitet werden sollen, und kann einfach an beliebiger stelle nach dem Programmaufruf geschehen.

```
        -r [Zahl]
```

Listing 18: Parameter -r zur Angabe der um die Zielknoten herum zu betrachtenden Autonomen Systeme

Mit dem in Listing 18 dargestellten Parameter -r wird angegeben, wie viele Hops zu anderen Knoten vom eigentlich zu betrachtenden Knoten angezeigt und damit simuliert werden sollen. Bei der Eingabe ist zu beachten, dass die auf den Parameter -r folgende Zahl nur aus dem natürlichen Zahlenraum gewählt werden darf und die Geschwindigkeit der Simulation stark beeinflusst, da das Wachstum der Knoten je nach gewählten Bereich abhängig von r bis zu exponentiell wachsen kann. Genauso schnell kann der Speicherbedarf wachsen, weshalb dieser Parameter mit Bedacht zu wählen ist und man sich bei Versuchen langsam an die, auf der simulierenden Hardware möglichen, Umfänge herantasten sollte.

```
        -k [AS]
        -k [AS1] [AS2]
```

Listing 19: Parameter -k zur Angabe der zu beobachtenden Knoten

Zur Bestimmung der Knoten, die man auf jeden Fall beobachten möchte, dient der Parameter -k, der in den beiden in Listing 19 gezeigten Formen anwendbar ist. Die obere legt nur ein

Autonomes System als Knoten fest, das als Zentrum gewählt wird, um das herum eine mit dem Parameter -r festgelegte Anzahl von Knoten dargestellt wird. Die untere der möglichen Anwendungen des Parameter -k gibt zwei Knoten an sowie deren kürzeste Verbindung zueinander, die beobachtet werden soll. Wenn keine Verbindung zwischen diesen beiden Knoten besteht, diese zu lang ist, oder sich mit der durch den Parameter -r festgelegten Anzahl der zu betrachtenden Autonomen Systeme nicht verträgt, wird die Software durch eine Fehlermeldung beendet mit einem Hinweis auf die mögliche Fehlerquelle.

```
-n [Name]
```

Listing 20: Parameter -n zur Angabe des Projektnamen, unter dem die Konfigurationsdaten gespeichert werden sollen

Zur Angabe des Verzeichnisnamens, unter dem die fertigen Konfigurationsdateien als GNS3 Projekt angelegt werden sollen, verwendet man den in Listing 20 beschriebenen Parameter -n. Es ist zu beachten, dass nur Namen akzeptiert werden können, die auf dem genutzten Betriebssystem und dem Dateiverzeichnis der verwendeten Massenspeicher zu einem nutzbaren Verzeichnisnamen führen.

Ein Beispiel zum Start der Software ist dem Listing 21 zu entnehmen.

```
arda
        bview.20140701.0800.ripe-rrc12.gz
        bview.20140701.0800.ripe-rrc11.gz
      -r 4
      -k 4711 0815
      -n test
```

Listing 21: Beispiel zur Ausführung der Software

In dem dargestellten Beispiel wird das mit dem Namen arda kompilierte Programm mit 2 Fulltable aus Frankfurt und New York vom gleichen Zeitpunkt gestartet. Es sollen alle Knoten simuliert werden, die maximal 4 Hops von den beiden zu betrachtenden Autonomen Systemen 4711 und 0815 entfernt liegen. Als Bezeichnung für das Projekt ist in diesem Beispiel "test"gewählt.

3.4 Datenverwertung

Für die Software ARDA werden nur wenige Daten benötigt, um das Routing zu einem bestimmten Zeitpunkt rekonstruieren zu können. Zur Vereinfachung der Simulation und somit zur Reduktion des Rechenaufwands wird jedes Autonome System durch einen einzelnen Router dargestellt. Von all den gegebenen Daten wird beim einlesen nur gespeichert welches AS mit welchem anderen verbunden ist und welches Autonome System welchen IP Bereich für sich beansprucht. Dann müssen für die Simulation nur die Verbindungen zwischen den Routern erzeugt werden sowie die Router die ein AS darstellen das IP Adressräume annonciert so eingestellt werden, dass sie das auch im der Simulation machen. Durch die Beschränkung der Ressourcen kann nicht das ganze Internet aus allen bekannten Autonomen Systemen Simuliert werden.

4 Evaluation

Zur Evaluation der Funktionen der Software ARDA wird im folgenden Kapitel dargestellt, wie aus BGP Routing Daten durch ARDA die Konfigurationsdateien für GNS3 und die darin emulierten Router entstehen. Dazu werden zuerst im Abschnitt 4.1 die Daten definiert, die zur Evaluation eingesetzt wurden, um die Reproduzierbarkeit der Ergebnisse sicherzustellen. Anschließend wird im Abschnitt 4.2 betrachtet, wie sich die Daten für ein kleines Autonomes System in drei Schritten über den Verlauf von vier Jahren verändern. Daraufhin wird in Abschnitt 4.3 gezeigt, wie ARDA Simulationen erstellt, wenn mehrere Quellen vorliegen, die den gleichen Zeitpunkt aus verschiedenen Sichtweisen beschreiben. Als nächstes zeigt Abschnitt 4.4 die Erstellung von Simulationen durch ARDA, bei der gleichzeitig zwei verschiedene Autonome Systeme betrachtet werden sollen. Zuletzt werden im Abschnitt 4.5 Probleme angesprochen, die im Zuge der Evaluation aufgefallen sind.

4.1 Verwendete Daten

Für die Evaluation von ARDA wurden BGP Daten von ris.ripe.net [12] verwendet, die von den in Listing 22 angegebenen Routing Reflector Clients zu den ebenfalls angegebenen Zeitpunkten stammen. Diese Daten liegen außerdem auf dem dieser Arbeit beigefügten Datenträger bei, um Probleme durch spätere Veränderungen der Daten bei der Quelle für die Reproduktion der Ergebnisse zu vermeiden. Sie sind so gewählt, dass zum einen die Veränderung der Daten über die Zeit von vier Jahren sichtbar wird, indem ein Datensatz von vor vier Jahren, einer von vor zwei Jahren und ein zum Zeitpunkt der Evaluation aktueller Satz Daten von demselben Routing Reflector Client aus Frankfurt genutzt wird. Außerdem wird zur Demonstration der Fähigkeit ein großes Datenvolumen zu verarbeiten, eine Simulation erstellt, die Daten von sechs verschiedenen Routing Reflector Clients vom selben Zeitpunkt enthält. Zusätzlich lässt sich so darstellen, wie sich Simulationen, die aus den Daten nur eines Routing Reflector Clients erstellt werden, von denen die aus den Daten mehrerer Routing Reflector Clients unterscheiden.

RRC	ORT	Datum	Zeit
rrc01	London, England	24.09.2014	16:00
rrc06	Otemachi, Japan	24.09.2014	16:00
rrc12	Frankfurt, Deutschland	09.10.2010	07:59
rrc12	Frankfurt, Deutschland	09.10.2012	07.59
rrc12	Frankfurt, Deutschland	24.09.2014	16:00
rrc13	Moskau, Russland	24.09.2014	16:00
rrc14	Palo Alto, USA	24.09.2014	16:00
rrc15	Sao Paulo, Brasilien	24.09.2014	16:00

Listing 22: Zur Evaluation genutzte Daten

4.2 Änderungen über den Zeitablauf

In diesem Abschnitt wird das Autonome System 250 betrachtet, dass laut peeringdb.com ein System vom Typ Non-Profit ist, das laut eigenen Angaben Open Source und Community Projekte fördert und sowohl IPv4 als auch IPv6 unterstützt [11]. Die Daten für die drei Simulationen, die auf den Daten vom Routing Collector Client 12 aus Frankfurt basieren, stammen

vom 09.10.2010, 09.10.2012 und dem 24.09.2014 und liegen damit weit genug auseinander, um betrachten zu können, wie sich die Routing Konfiguration um dieses Autonome System im Laufe der Jahre verändert hat. Zum Zeitpunkt 09.10.2010 07:59 Uhr lagen die in Listing 23 gezeigten Daten vor, die mit dem der libbgpdump [**?**] beiliegenden Tool in ein lesbares Format gebracht wurden und sich im Verzeichnis *daten* mit dem Namen *bviewrrc12.20101009.1600* auf dem dieser Arbeit beiliegenden Datenträger befindet. Diese Datei enthält 2259588 Einträge, jeweils einen pro Zeile, die mit dem ebenfalls in dem Listing angegebenen *grep* Kommando nach Zeilen, mit für das Autonome System 250 relevantem Inhalt, durchsucht wurden.

```
grep '[ \|]250[\| ]' bviewrrc12.20101009.0759
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.219|33843|193.227.234.0/23|33843 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.35|1299|193.227.234.0/23|1299 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.83|13101|193.227.234.0/23|13101 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.133|29686|193.227.234.0/23|29686 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.23|20640|193.227.234.0/23|20640 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.14|4589|193.227.234.0/23|4589 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.220|19151|193.227.234.0/23|19151 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.194|6762|193.227.234.0/23|6762 3549 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.74|13237|193.227.234.0/23|13237 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.30|3257|193.227.234.0/23|3257 3549 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.49|5430|194.150.168.0/23|5430 6724 29670 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.219|33843|194.150.168.0/23|33843 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.83|13101|194.150.168.0/23|13101 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.35|1299|194.150.168.0/23|1299 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.46|2914|194.150.168.0/23|2914 2914 2914 286 12732 29670 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.133|29686|194.150.168.0/23|29686 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.23|20640|194.150.168.0/23|20640 33843 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.14|4589|194.150.168.0/23|4589 33843 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.220|19151|194.150.168.0/23|19151 33843 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.194|6762|194.150.168.0/23|6762 3549 8495 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.74|13237|194.150.168.0/23|13237 12732 29670 250|IGP
TABLE_DUMP_V2|10/09/10 07:59:58|A|80.81.192.30|3257|194.150.168.0/23|3257 13237 12732 29670 250|IGP
```

Listing 23: Daten zum AS 250 von RRC12 am 09.10.2010 um 07:59 Uhr

In diesem Listing ist in jeder Zeile zu erkennen:
- Der Typ des Datensatzes
- Datum und Uhrzeit der Daten
- IP Adresse des Viewpoints, der die Route gesehen hat
- AS Nummer, zu der der Viewpoint gehört
- Annonciertes IP Prefix
- Bekannter Pfad zum Origin des IP Prefix

Dabei ist der letzte Punkt des Pfades zum Origin das Origin selbst. Das zeigt, dass das Autonome System 250 ausschließlich Daten empfängt und keine weiterleitet, weshalb es vermutlich ein Endpunkt und damit ein Tier 3 Provider ist. Es annonciert zwei IP Prefixe, *193.227.234.0/23* und *194.150.168.0/23*, die auf mehreren Pfaden von verschiedenen Viewpoints zu erreichen sind. Außerdem ist zu erkennen, dass es ausschließlich mit den beiden Autonomen Systemen *8495* und *29670* peert. Diese Informationen sollen sich in der Simulation wiederfinden, was im Folgenden gezeigt wird.

Die graphische Repräsentation der Simulation, die ARDA mit dem in Listing 24 gezeigten Befehl erstellt, ist in Abbildung 6, in der von ARDA erstellten Ansicht, zu sehen. Man erkennt, wie GNS3 die Simulation öffnet, anzeigt und startet. Der Projektordner zu dieser Simulation, mit den für GNS3 notwendigen Daten, befindet sich auf dem dieser Arbeit beiliegenen Datenträger im Verzeichnis *Simulationen* im Projektverzeichnis *vierjahre250*.

```
./arda -r 1 -n vierjahre250 -k 250 bviewrrc12.20101009.0759.gz
```

Listing 24: Aufruf zum Erzeugen der Simulation *vierjahre250*

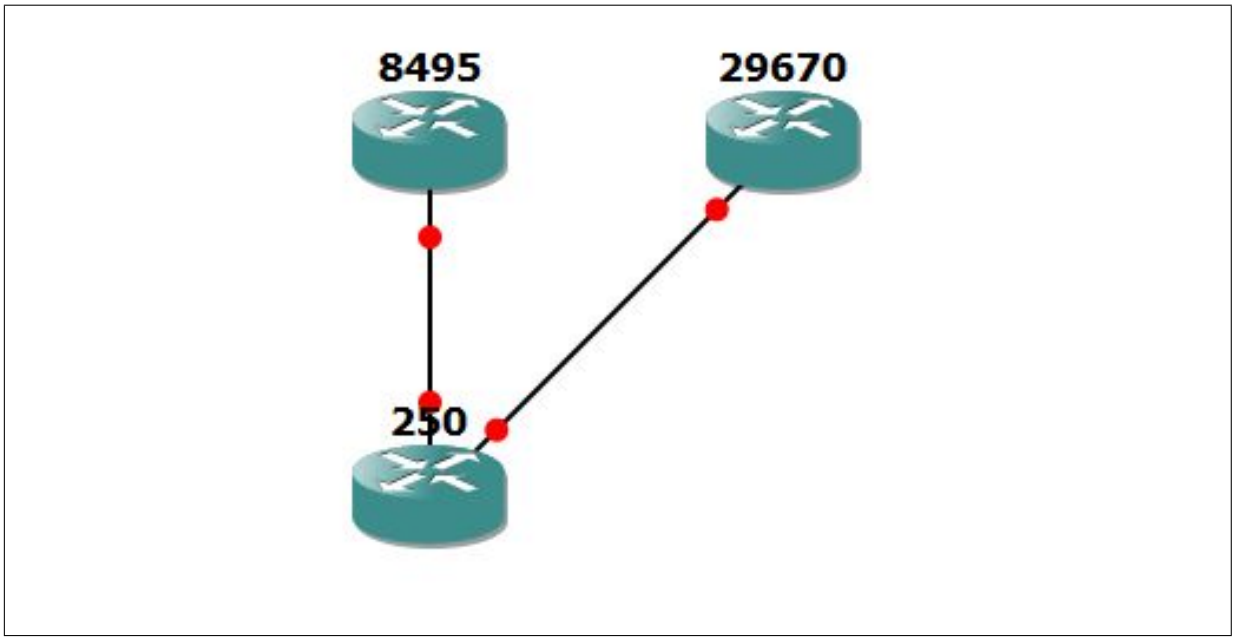

Abbildung 6: Graphische Repräsentation der Simulation *vierjahre250*, wie von ARDA erstellt

In Abbildung 6 ist zu erkennen, dass das Autonome System 250 mit den beiden Autonomen Systemen *8495* und *29670* peert, was den, aus den Daten extrahierten Werten, entspricht. Zusätzlich ist in dem Ausschnitt aus der Konfigurationsdatei für den Router 250, wie in Listing 25 zu sehen ist, zu erkennen, dass die beiden IP Prefixe, die in den Ausgangsdaten von dem Autonomen System 250 annonciert werden, auch von dem Router 250 in der Simulation annonciert werden.

```
network 193.227.234.0 mask 255.255.254.0
network 194.150.168.0 mask 255.255.254.0
```

Listing 25: Ausschnitt aus der Konfigurationsdatei von Router 250

Als nächstes wird der Zeitpunkt 09.10.2012 07:59 betrachtet, der genau 2 Jahre nach dem vorherigen liegt, und deren Unterschiede im folgenden Listing 26 betrachtet werden.Man sieht die Daten, die in der Datei von dem Route Reflector Client 12 aus Frankfurt zu der angegeben Zeit waren, die mit dem Autonomen System 250 in Verbindung stehen. Dieses Listing ist deutlich länger, da mit der Zeit die Anzahl der Autonomen Systeme gestiegen, die vergebenen IP Prefixe mehr geworden sind und die Aufzeichnung der Routing Daten besser geworden ist. Die Ursprungsdatei zu diesem Listing beinhaltet 6277355 Zeilen, und damit fast dreimal mehr Daten als die Datei zum vorhergehenden Zeitpunkt, und befindet sich auf dem Datenträger im Verzeichnis *daten* unter der Bezeichnung *bviewrrc12.20121009.0759*.

```
grep '[\| ]250[\| ]' bview.20121009.0759
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.119|34288|193.227.234.0/23|34288 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.133|29686|193.227.234.0/23|29686 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.194|6762|193.227.234.0/23|6762 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.23|20640|193.227.234.0/23|20640 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.14|4589|193.227.234.0/23|4589 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.82|8222|193.227.234.0/23|8222 1299 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.74|13237|193.227.234.0/23|13237 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.175|553|193.227.234.0/23|553 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.46|2914|193.227.234.0/23|2914 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.30|3257|193.227.234.0/23|3257 174 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.59|29140|193.227.234.0/23|29140 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.98|9189|193.227.234.0/23|9189 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.222|680|193.227.234.0/23|680 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.220|19151|193.227.234.0/23|19151 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.78|8359|193.227.234.0/23|8359 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.83|13101|193.227.234.0/23|13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.119|34288|194.150.168.0/23|34288 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.133|29686|194.150.168.0/23|29686 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.194|6762|194.150.168.0/23|6762 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.23|20640|194.150.168.0/23|20640 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.14|4589|194.150.168.0/23|4589 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.82|8222|194.150.168.0/23|8222 1299 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.74|13237|194.150.168.0/23|13237 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.175|553|194.150.168.0/23|553 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.46|2914|194.150.168.0/23|2914 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.30|3257|194.150.168.0/23|3257 174 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.59|29140|194.150.168.0/23|29140 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.98|9189|194.150.168.0/23|9189 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.222|680|194.150.168.0/23|680 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.220|19151|194.150.168.0/23|19151 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.78|8359|194.150.168.0/23|8359 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.83|13101|194.150.168.0/23|13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.119|34288|195.85.254.0/24|34288 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.133|29686|195.85.254.0/24|29686 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.194|6762|195.85.254.0/24|6762 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.23|20640|195.85.254.0/24|20640 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.14|4589|195.85.254.0/24|4589 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.82|8222|195.85.254.0/24|8222 1299 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.74|13237|195.85.254.0/24|13237 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.175|553|195.85.254.0/24|553 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.46|2914|195.85.254.0/24|2914 3320 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.30|3257|195.85.254.0/24|3257 174 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.194.59|29140|195.85.254.0/24|29140 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.98|9189|195.85.254.0/24|9189 13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.222|680|195.85.254.0/24|680 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.220|19151|195.85.254.0/24|19151 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.78|8359|195.85.254.0/24|8359 31025 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:55|A|80.81.192.83|13101|195.85.254.0/24|13101 8495 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::1b1b:0:1|6939|2001:4ce8::/32|6939 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::85f0:0:1|34288|2001:4ce8::/32|34288 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::1a0b:0:1|6667|2001:4ce8::/32|6667 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::cb9:0:1|3257|2001:4ce8::/32|3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::11ed:0:1|4589|2001:4ce8::/32|4589 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::b62:0:1|2914|2001:4ce8::/32|2914 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::1a6a:0:1|6762|2001:4ce8::/32|6762 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::32e6:0:1|13030|2001:4ce8::/32|13030 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::73f6:0:1|29686|2001:4ce8::/32|29686 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::63d8:0:1|25560|2001:4ce8::/32|25560 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::11e:0:1|286|2001:4ce8::/32|286 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::50a0:0:1|20640|2001:4ce8::/32|20640 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::2a8:0:1|680|2001:4ce8::/32|680 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::23e5:0:1|9189|2001:4ce8::/32|9189 6939 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::229:0:1|553|2001:4ce8::/32|553 1299 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::223b:0:1|8763|2001:4ce8::/32|8763 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::b29:0:1|2857|2001:4ce8::/32|2857 8365 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::232a:0:1|9002|2001:4ce8::/32|9002 2497 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::71d4:0:1|29140|2001:4ce8::/32|29140 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::33b5:0:1|13237|2001:4ce8::/32|13237 3257 250|IGP
TABLE_DUMP_V2|10/09/12 07:59:59|A|2001:7f8::8605:0:1|34309|2001:4ce8::/32|34309 3257 250|EGP
```

Listing 26: Daten zum AS 250 von RRC12 am 09.10.2012 um 07:59 Uhr

Die graphische Repräsentation der Simulation *zweijahre250*, die aus den Daten, mit dem in Listing 27 angegebenen Befehl, durch ARDA erstellt wurde, ist in Abbildung 7 zu sehen. Das zugehörige Projekt *zweijahre250* befindet sich auf dem dieser Arbeit beiliegenden Datenträger im entsprechenden Projektverzeichnis.

```
./arda -r 1 -n zweijahre250 -k 250 bviewrrc12.20121009.0759.gz
```

Listing 27: Daten zum AS 250 von RRC12 am 09.10.2012 um 07:59 Uhr

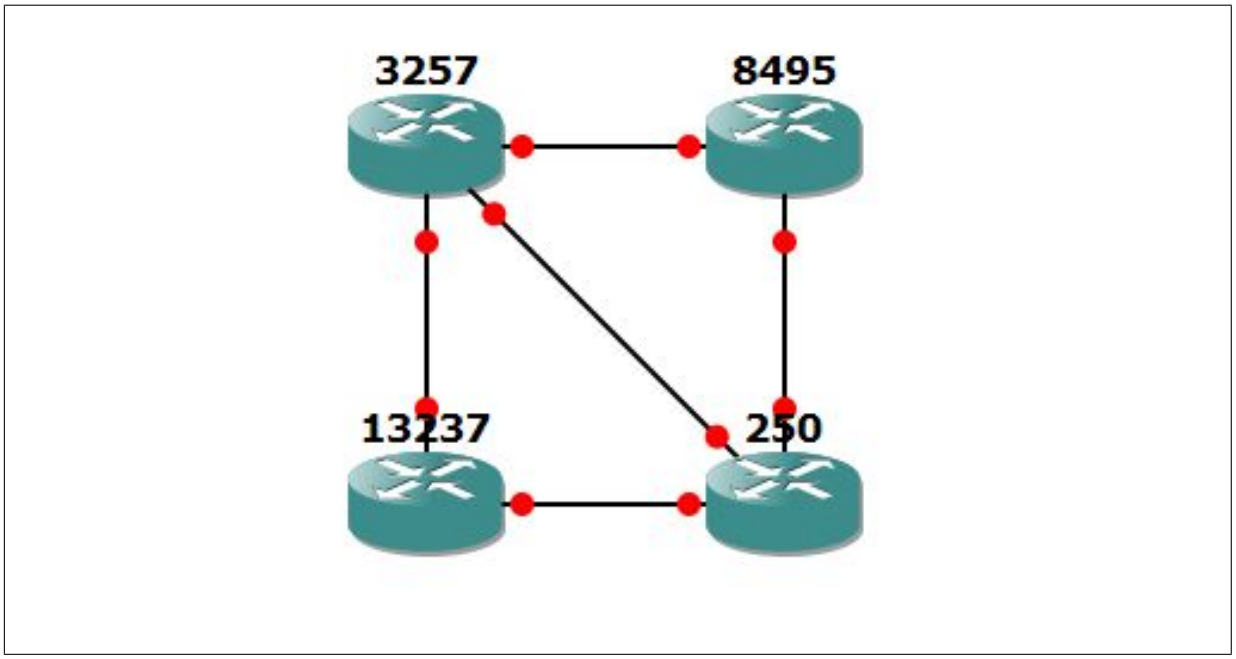

Abbildung 7: Graphische Repräsentation der Simulation *zweijahre250*, wie von ARDA erstellt

In den Daten im Listing 26, wie auch auf der Abbildung 7, ist zu erkennen, dass das Autonome System 250 mit den Autonomen Systemen *3257*, *8495* und *13237* peert. Außerdem sind dort die gleichen annoncierten IP Prefixe zu finden, die auch in der Konfigurationsdatei von Router 250 zu finden sind, von der ein Ausschnitt in Listing 28 zu sehen ist.

```
network 193.227.234.0 mask 255.255.254.0
network 194.150.168.0 mask 255.255.254.0
network 195.85.254.0 mask 255.255.255.0
address-family ipv6
network 2001:4ce8::/32
exit
```

Listing 28: Ausschnitt aus der Konfigurationsdatei von Router 250

Im Vergleich zwischen den Simulationen *vierjahre250* und *zweijahre250* ist außerdem festzustellen, dass nur das Autonome System 8495 in beiden Simulationen als Peeringpartner für das Autonome System 250 vorhanden ist. Weiterhin ist die Simulation von drei auf vier Autonome Systeme gewachsen, da zwei weitere hinzugekommen sind. Diese drei beziehungsweise vier Autonomen Systeme sind jeweils untereinander nicht vollverknüpft, wie es auch in den dazugehörigen Daten angegeben ist.

Als letztes in diesem Abschnitt der Evaluation wird die Simulation *aktuell250* betrachet, die aus Daten vom 24.09.2014 16:00 entstanden ist. Im Listing 29 ist ein Ausschnitt aus der Datei *bviewrrc12.20140924.1600* zu sehen, die sich auf dem dieser Arbeit beiliegenden Datenträger im Verzeichnis *daten* befindet, mit den Zeilen die das Autonome System 250 betreffen.

```
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.220|19151|193.227.234.0/23|19151 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.74|13237|193.227.234.0/23|13237 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|2001:7f8::bc26:0:1|48166|193.227.234.0/23|48166 251 250|IGP
```

```
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.194.204|48166|193.227.234.0/23|48166 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.194.82|8222|193.227.234.0/23|8222 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.23|20640|193.227.234.0/23|20640 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.133|29686|193.227.234.0/23|29686 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.194.140|25220|193.227.234.0/23|25220 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.98|9189|193.227.234.0/23|9189 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.14|4589|193.227.234.0/23|4589 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.194.119|34288|193.227.234.0/23|34288 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.194.59|29140|193.227.234.0/23|29140 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.30|3257|193.227.234.0/23|3257 1299 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.175|553|193.227.234.0/23|553 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.222|680|193.227.234.0/23|680 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.46|2914|193.227.234.0/23|2914 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:06|A|80.81.192.194|6762|193.227.234.0/23|6762 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.220|19151|194.150.168.0/23|19151 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.74|13237|194.150.168.0/23|13237 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|2001:7f8::bc26:0:1|48166|194.150.168.0/23|48166 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.204|48166|194.150.168.0/23|48166 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.82|8222|194.150.168.0/23|8222 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.23|20640|194.150.168.0/23|20640 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.133|29686|194.150.168.0/23|29686 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.140|25220|194.150.168.0/23|25220 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.98|9189|194.150.168.0/23|9189 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.14|4589|194.150.168.0/23|4589 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.119|34288|194.150.168.0/23|34288 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.59|29140|194.150.168.0/23|29140 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.30|3257|194.150.168.0/23|3257 1299 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.175|553|194.150.168.0/23|553 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.222|680|194.150.168.0/23|680 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.46|2914|194.150.168.0/23|2914 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.194|6762|194.150.168.0/23|6762 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.220|19151|195.85.254.0/24|19151 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.74|13237|195.85.254.0/24|13237 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|2001:7f8::bc26:0:1|48166|195.85.254.0/24|48166 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.204|48166|195.85.254.0/24|48166 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.82|8222|195.85.254.0/24|8222 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.23|20640|195.85.254.0/24|20640 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.133|29686|195.85.254.0/24|29686 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.140|25220|195.85.254.0/24|25220 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.98|9189|195.85.254.0/24|9189 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.14|4589|195.85.254.0/24|4589 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.119|34288|195.85.254.0/24|34288 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.59|29140|195.85.254.0/24|29140 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.30|3257|195.85.254.0/24|3257 1299 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.175|553|195.85.254.0/24|553 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.222|680|195.85.254.0/24|680 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.46|2914|195.85.254.0/24|2914 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.194|6762|195.85.254.0/24|6762 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.220|19151|195.245.114.0/23|19151 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.74|13237|195.245.114.0/23|13237 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|2001:7f8::bc26:0:1|48166|195.245.114.0/23|48166 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.204|48166|195.245.114.0/23|48166 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.82|8222|195.245.114.0/23|8222 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.23|20640|195.245.114.0/23|20640 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.133|29686|195.245.114.0/23|29686 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.140|25220|195.245.114.0/23|25220 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.98|9189|195.245.114.0/23|9189 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.14|4589|195.245.114.0/23|4589 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.119|34288|195.245.114.0/23|34288 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.194.59|29140|195.245.114.0/23|29140 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.30|3257|195.245.114.0/23|3257 1299 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.175|553|195.245.114.0/23|553 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.222|680|195.245.114.0/23|680 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.46|2914|195.245.114.0/23|2914 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:07|A|80.81.192.194|6762|195.245.114.0/23|6762 251 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::cb9:0:1|3257|2001:4ce8::/32|3257 6939 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::1a0b:0:1|6667|2001:4ce8::/32|6667 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::33b5:0:1|13237|2001:4ce8::/32|13237 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::b29:0:1|2857|2001:4ce8::/32|2857 6939 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::bc26:0:1|48166|2001:4ce8::/32|48166 9002 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::11e:0:1|286|2001:4ce8::/32|286 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::50a0:0:1|20640|2001:4ce8::/32|20640 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::71d4:0:1|29140|2001:4ce8::/32|29140 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::23e5:0:1|9189|2001:4ce8::/32|9189 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::6284:0:1|25220|2001:4ce8::/32|25220 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::73f6:0:1|29686|2001:4ce8::/32|29686 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::1b1b:0:1|6939|2001:4ce8::/32|6939 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::11ed:0:1|4589|2001:4ce8::/32|4589 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::63d8:0:1|25560|2001:4ce8::/32|25560 6939 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::223b:0:1|8763|2001:4ce8::/32|8763 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::229:0:1|553|2001:4ce8::/32|553 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::1a6a:0:1|6762|2001:4ce8::/32|6762 6939 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::2a8:0:1|680|2001:4ce8::/32|680 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::8605:0:1|34309|2001:4ce8::/32|34309 6939 250|EGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::b62:0:1|2914|2001:4ce8::/32|2914 1299 6724 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::232a:0:1|9002|2001:4ce8::/32|9002 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::85f0:0:1|34288|2001:4ce8::/32|34288 6939 250|IGP
TABLE_DUMP_V2|09/24/14 16:00:09|A|2001:7f8::32e6:0:1|13030|2001:4ce8::/32|13030 6724 250|IGP
```

Listing 29: Daten zum AS 250 von RRC12 am 24.09.2014 um 16:00 Uhr

Hier ist im Vergleich zu Abbildung 8 zu erkennen, dass ARDA die richtigen Autonomen Systeme gefunden und in die Simulation eingepflegt hat. Der Befehl zum Erstellen der Simulation ist in Listing 30 zu sehen. Die mit dem Autonomen System 250 peerenden Autonomen Systeme sind: *251*, *286*, *680*, *6724*, *6939*, *9002* und *13237*, welches als einziges aus der Simulation *zweijahre250* bekannt ist.

```
./arda -r 1 -n aktuell250 -k 250 bviewrrc12.20140924.1600.gz
```

Listing 30: Daten zum AS 250 von RRC12 am 24.09.2014 um 16:00 Uhr

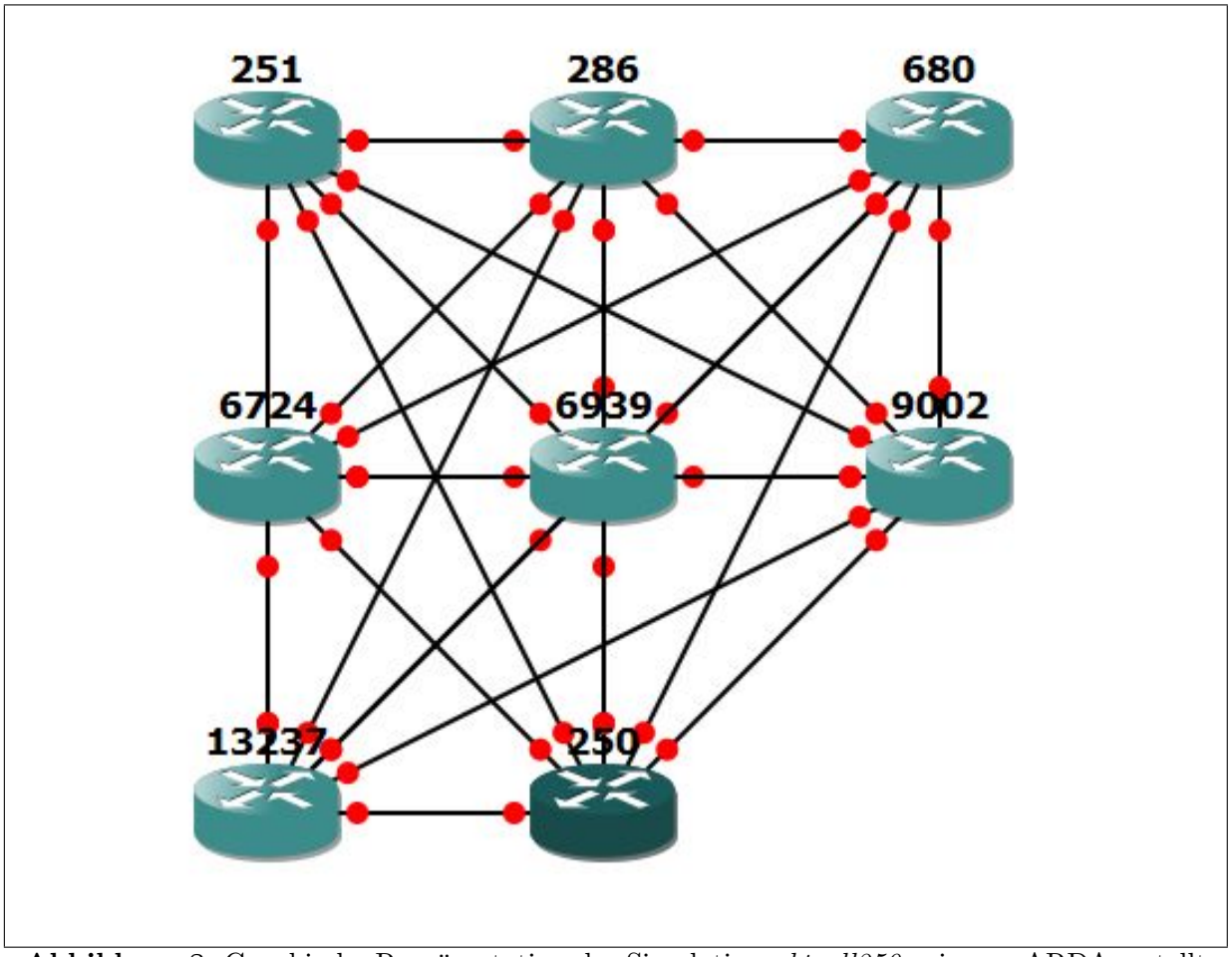

Abbildung 8: Graphische Repräsentation der Simulation *aktuell250*, wie von ARDA erstellt

Durch den Ausschnitt der Konfigurationsdatei von Router 250, der in Listing 31 gezeigt ist, lässt sich im Vergleich mit den Ursprungsdaten aus Listing 29 feststellen, dass auch alle IP Prefixe richtig annonciert wurden.

```
network 193.227.234.0 mask 255.255.254.0
network 194.150.168.0 mask 255.255.254.0
network 195.85.254.0 mask 255.255.255.0
network 195.245.114.0 mask 255.255.254.0
address-family ipv6
network 2001:4ce8::/32
exit
```

Listing 31: Ausschnitt aus der Konfigurationsdatei von Router 250

Zusätzlich ist in Abbildung 9 eine manuell nachbearbeitete Ansicht der graphischen Simulation *aktuell250* zu sehen, auf der alle Peerings zwischen den einzelnen Routern eindeutig zu erkennen sind, die auf der automatisch erstellten Version teilweise durch andere Router und Linien verdeckt waren.

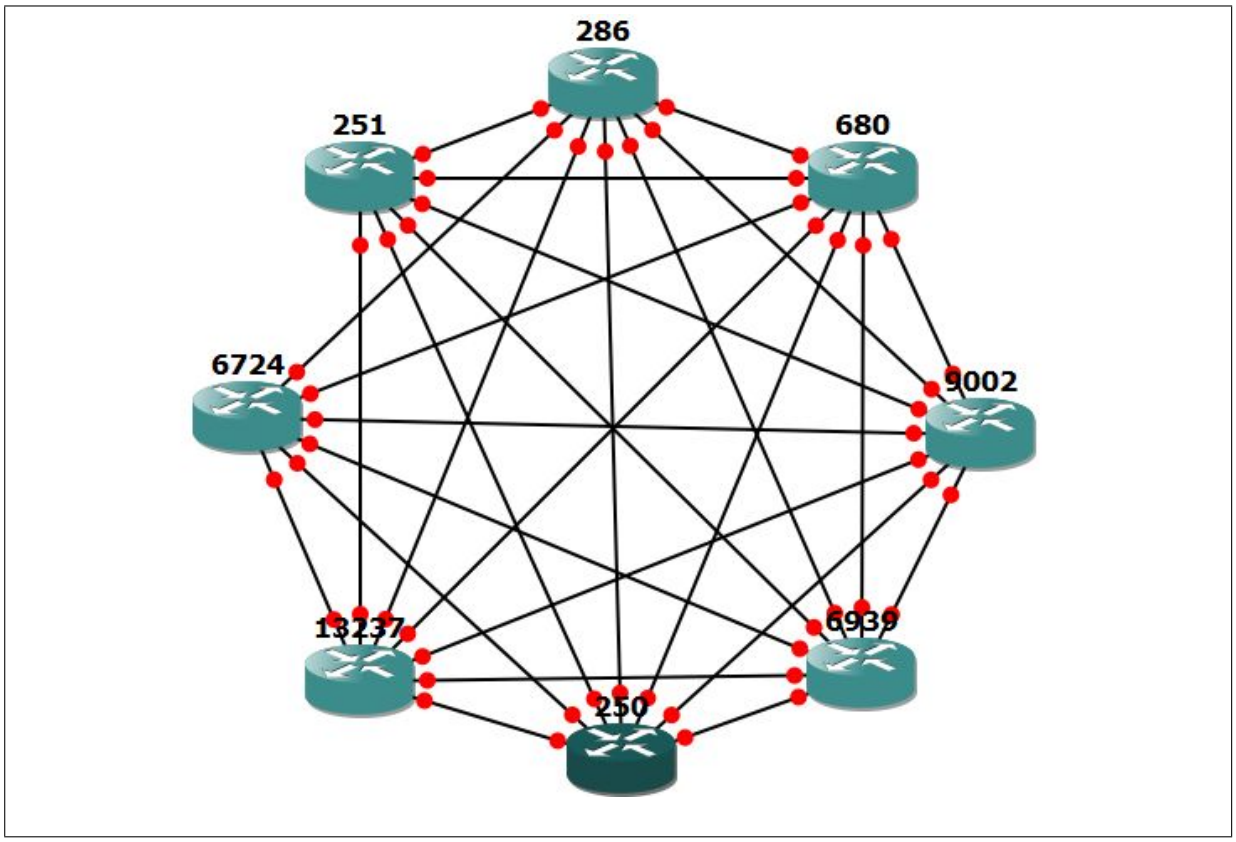

Abbildung 9: Graphische Repräsentation der Simulation *aktuell250*, manuell angeordnet

In diesem Abschnitt wurde damit nachgewiesen, dass ARDA für ein Autonomes System und seine Nachbarn aus den Daten einer Quelle eine Simulation erstellen kann, die der tatsächlichen Internet Routing Situation entspricht und die lauffähig ist, um damit arbeiten zu können. Weiterhin ist erkennbar, dass sich die Internet Routing Situation im Laufe der Zeit verändert und daher für die Analyse von Anomalien immer neue Simulationen erstellt werden müssen.

4.3 Mehrere Quellen

Die Software ist auch in der lage mehrere Quellen in die Generierung einer Simulation einzubinden. Um das zu demonstrieren, wurde in der folgenden Simulation *aktuellgross250* der gleiche Ausschnitt, wie in der im vorherigen Abschnitt 4.2 beschriebenen Simulation *aktuell250*, um das Autonome System 250 betrachtet. Während im Vergleich von Abbildung 9 von Simulation *aktuell250* und Abbildung 10 von Simulation *aktuellgross250* kein Unterschied im Peering vom Autonomen System 250 auffällt, lassen sich in den Annoncierungen in den Konfigurationsdateien der Router viele Unterschiede finden. Die beiden Konfigurationsdateien der beiden Router 250

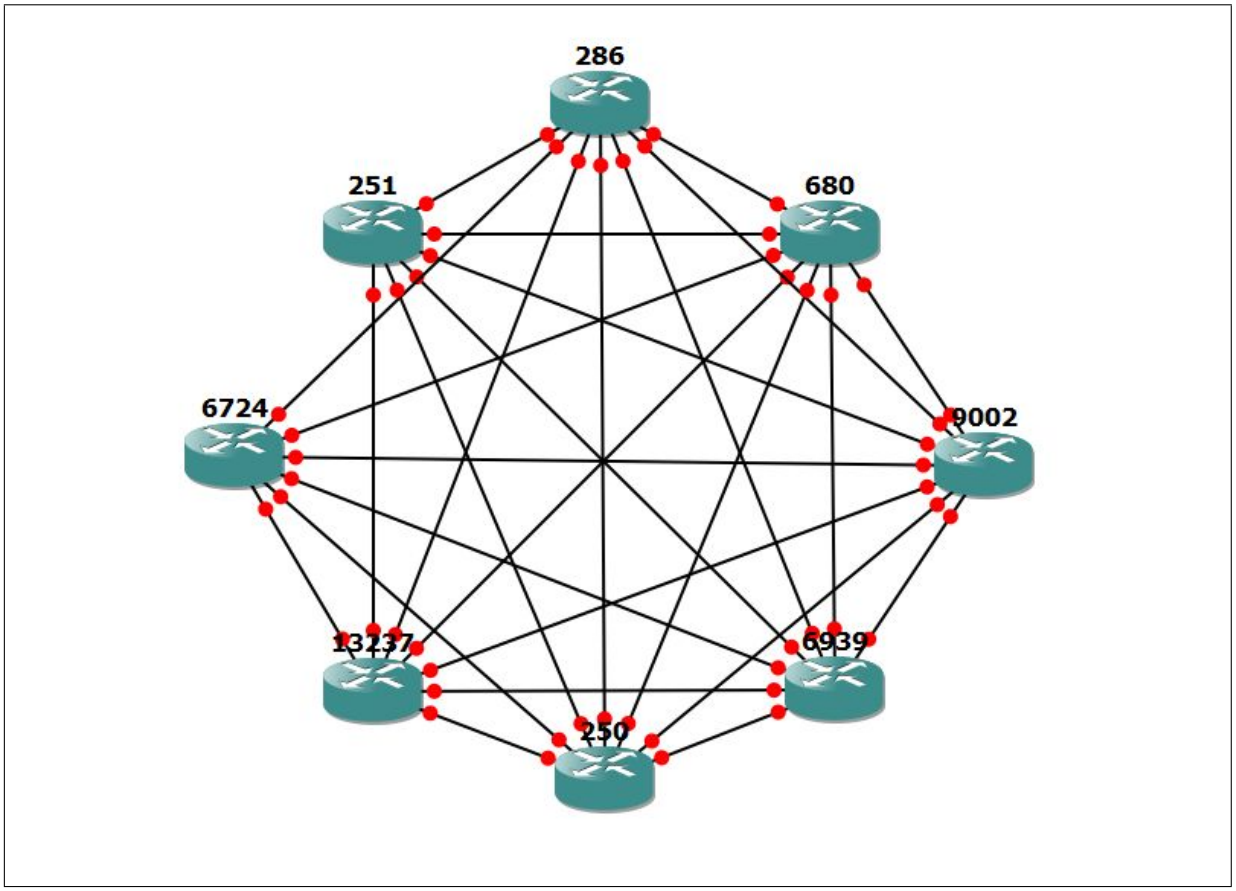

Abbildung 10: Graphische Repräsentation der Simulation *aktuellgross250*, manuell angeordnet

in den beiden Simulationen weisen keinen Unterschied auf, dafür aber einige der anderen. Als Beispiel wird hier im Folgenden der Unterschied der Dateien der Router 286 genutzt und hier gezeigt. Dazu wird in Abbildung 11 der Abschnitt der Annoncierungen von Router 286 aus der Simulation *aktuell250* und der Simulation *aktuellgross250* gegenübergestellt. Um die Übersicht zu verbessern, ist in Listing 32 die Differenz der beiden Konfigurationsdateien dargestellt, wie mit dem Linux Tool *diff* erstellt.

```
diff aktuell250/configs/286.cfg aktuellgross250/configs/286.cfg
365d364
<  network 62.41.3.0 mask 255.255.255.0
383a383,384
>  network 91.213.218.0 mask 255.255.255.192
>  network 91.213.218.128 mask 255.255.255.192
392a394
>  network 193.172.150.128 mask 255.255.255.224
398a401,402
>  network 194.120.0.252 mask 255.255.255.255
>  network 194.120.12.240 mask 255.255.255.240
406d409
<  network 194.122.226.0 mask 255.255.255.0
413a417
>  network 194.221.41.0 mask 255.255.255.248
417a422,423
>  network 212.1.21.144 mask 255.255.255.248
>  network 212.1.21.192 mask 255.255.255.224
421a428,430
>  network 212.189.3.40 mask 255.255.255.254
>  network 212.189.7.160 mask 255.255.255.224
>  network 212.189.81.70 mask 255.255.255.255
422a432,435
>  network 212.189.122.112 mask 255.255.255.240
>  network 212.189.123.184 mask 255.255.255.248
>  network 62.41.3.0 mask 255.255.255.0
>  network 194.122.226.0 mask 255.255.255.0
```

Listing 32: Differenz zwischen den Konfigurationsdateien

In dieser Darstellung ist zu sehen, dass bei der Simulation *aktuellgross250* einige Annoncierungen mehr vorliegen als in der Simulation *aktuell250*, bei der nur eine Quelle genutzt wurde. Obwohl aus Sicht des Border Gateway Protocol kein Grund dafür besteht, scheinen andere Viewpoints zusätzliche Annoncierungen vom Router 286 zu kennen, als die die in Frankfurt stehen.

Durch diese Darstellung wird gezeigt, dass ARDA die Daten aus mehreren Quellen richtig verarbeitet und in die Simulationen einfließen lässt.

```
network 41.207.107.0 mask 255.255.255.0
network 41.207.108.0 mask 255.255.255.0
network 62.41.0.0 mask 255.255.240.0
network 62.41.3.0 mask 255.255.255.0
network 62.41.16.0 mask 255.255.248.0
network 62.41.24.0 mask 255.255.252.0
network 62.41.56.0 mask 255.255.248.0
network 62.41.64.0 mask 255.255.240.0
network 62.41.80.0 mask 255.255.252.0
network 62.41.84.0 mask 255.255.254.0
network 62.41.160.0 mask 255.255.255.0
network 62.132.0.0 mask 255.255.252.0
network 62.132.24.0 mask 255.255.254.0
network 62.132.28.0 mask 255.255.255.0
network 62.132.42.0 mask 255.255.254.0
network 62.132.114.0 mask 255.255.254.0
network 62.132.116.0 mask 255.255.254.0
network 62.132.132.0 mask 255.255.254.0
network 78.40.64.0 mask 255.255.255.0
network 78.40.65.0 mask 255.255.255.0
network 83.138.80.0 mask 255.255.248.0
network 91.213.218.0 mask 255.255.255.0
network 92.71.0.0 mask 255.255.128.0
network 134.222.0.0 mask 255.255.0.0
network 192.67.196.0 mask 255.255.255.0
network 192.129.32.0 mask 255.255.254.0
network 193.17.185.0 mask 255.255.255.0
network 193.17.186.0 mask 255.255.255.0
network 193.27.0.0 mask 255.255.255.0
network 193.141.0.0 mask 255.255.252.0
network 193.141.40.0 mask 255.255.252.0
network 193.242.80.0 mask 255.255.240.0
network 194.45.12.0 mask 255.255.254.0
network 194.45.46.0 mask 255.255.254.0
network 194.45.46.0 mask 255.255.255.0
network 194.45.96.0 mask 255.255.252.0
network 194.120.0.0 mask 255.255.255.0
network 194.121.104.0 mask 255.255.254.0
network 194.121.123.0 mask 255.255.255.0
network 194.121.220.0 mask 255.255.252.0
network 194.122.76.0 mask 255.255.252.0
network 194.122.80.0 mask 255.255.248.0
network 194.122.131.0 mask 255.255.255.0
network 194.122.224.0 mask 255.255.240.0
network 194.122.226.0 mask 255.255.255.0
network 194.122.240.0 mask 255.255.248.0
network 194.122.248.0 mask 255.255.252.0
network 194.122.252.0 mask 255.255.254.0
network 194.123.7.0 mask 255.255.255.0
network 194.123.122.0 mask 255.255.255.0
network 194.123.164.0 mask 255.255.255.0
network 194.151.203.0 mask 255.255.255.0
network 194.221.186.0 mask 255.255.255.0
network 195.27.237.0 mask 255.255.255.0
network 195.234.152.0 mask 255.255.255.0
network 197.231.254.0 mask 255.255.255.0
network 212.50.42.0 mask 255.255.255.0
network 212.50.45.0 mask 255.255.255.0
network 212.165.80.0 mask 255.255.240.0
network 212.165.112.0 mask 255.255.240.0
network 212.189.102.0 mask 255.255.255.0
address-family ipv6
network 200:120:2001:680::/32
network 200:130:2a01:4da0::/48
exit
```

Listing 33: *aktuell250*

```
network 41.207.107.0 mask 255.255.255.0
network 41.207.108.0 mask 255.255.255.0
network 62.41.0.0 mask 255.255.240.0
network 62.41.16.0 mask 255.255.248.0
network 62.41.24.0 mask 255.255.252.0
network 62.41.56.0 mask 255.255.248.0
network 62.41.64.0 mask 255.255.240.0
network 62.41.80.0 mask 255.255.252.0
network 62.41.84.0 mask 255.255.254.0
network 62.41.160.0 mask 255.255.255.0
network 62.132.0.0 mask 255.255.252.0
network 62.132.24.0 mask 255.255.254.0
network 62.132.28.0 mask 255.255.255.0
network 62.132.42.0 mask 255.255.254.0
network 62.132.114.0 mask 255.255.254.0
network 62.132.116.0 mask 255.255.254.0
network 62.132.132.0 mask 255.255.254.0
network 78.40.64.0 mask 255.255.255.0
network 78.40.65.0 mask 255.255.255.0
network 83.138.80.0 mask 255.255.248.0
network 91.213.218.0 mask 255.255.255.0
network 91.213.218.0 mask 255.255.255.192
network 91.213.218.128 mask 255.255.255.192
network 92.71.0.0 mask 255.255.128.0
network 134.222.0.0 mask 255.255.0.0
network 192.67.196.0 mask 255.255.255.0
network 192.129.32.0 mask 255.255.254.0
network 193.17.185.0 mask 255.255.255.0
network 193.17.186.0 mask 255.255.255.0
network 193.27.0.0 mask 255.255.255.0
network 193.141.0.0 mask 255.255.252.0
network 193.141.40.0 mask 255.255.252.0
network 193.172.150.128 mask 255.255.255.224
network 193.242.80.0 mask 255.255.240.0
network 194.45.12.0 mask 255.255.254.0
network 194.45.46.0 mask 255.255.254.0
network 194.45.46.0 mask 255.255.255.0
network 194.45.96.0 mask 255.255.252.0
network 194.120.0.0 mask 255.255.255.0
network 194.120.0.252 mask 255.255.255.255
network 194.120.12.240 mask 255.255.255.240
network 194.121.104.0 mask 255.255.254.0
network 194.121.123.0 mask 255.255.255.0
network 194.121.220.0 mask 255.255.252.0
network 194.122.76.0 mask 255.255.252.0
network 194.122.80.0 mask 255.255.248.0
network 194.122.131.0 mask 255.255.255.0
network 194.122.224.0 mask 255.255.240.0
network 194.122.240.0 mask 255.255.248.0
network 194.122.248.0 mask 255.255.252.0
network 194.122.252.0 mask 255.255.254.0
network 194.123.7.0 mask 255.255.255.0
network 194.123.122.0 mask 255.255.255.0
network 194.123.164.0 mask 255.255.255.0
network 194.151.203.0 mask 255.255.255.0
network 194.221.41.0 mask 255.255.255.248
network 194.221.186.0 mask 255.255.255.0
network 195.27.237.0 mask 255.255.255.0
network 195.234.152.0 mask 255.255.255.0
network 197.231.254.0 mask 255.255.255.0
network 212.1.21.144 mask 255.255.255.248
network 212.1.21.192 mask 255.255.255.224
network 212.50.42.0 mask 255.255.255.0
network 212.50.45.0 mask 255.255.255.0
network 212.165.80.0 mask 255.255.240.0
network 212.165.112.0 mask 255.255.240.0
network 212.189.3.40 mask 255.255.255.254
network 212.189.7.160 mask 255.255.255.224
network 212.189.81.70 mask 255.255.255.255
network 212.189.102.0 mask 255.255.255.0
network 212.189.122.112 mask 255.255.255.240
network 212.189.123.184 mask 255.255.255.248
network 62.41.3.0 mask 255.255.255.0
network 194.122.226.0 mask 255.255.255.0
address-family ipv6
network 200:120:2001:680::/32
network 200:130:2a01:4da0::/48
exit
```

Listing 34: *aktuellgross250*

Abbildung 11: Gegenüberstellung von Annoncierungen von Router 286

4.4 Mehrere Autonome Systeme betrachten

In diesem Abschnitt wird gezeigt, dass ARDA auch zwei Autonome Systeme betrachten kann, die nicht direkt miteinander peeren. Außerdem wird auch die Simulation von vielen Autonomen Systemen gezeigt, die jedoch stark durch die Kapazität der vorhandenen Hardware beschränkt wird. Wie in der Simulation *vierjahre250* in Abschnitt 4.2 gezeigt, hat das Autonome System zu diesem Zeitpunkt keine Verbindung zu dem Autonomen System 680. Zum gleichen Zeitpunkt lässt sich das Autonome System 680 betrachten, das wiederum kein Peering mit dem Autonomen System 250 unterhält. Die Simulation *vierjahre680*, deren manuell angeordnete graphische Simulation man in Abbildung 12 sehen kann, bestätigt diesen Sachverhalt und dient der Gegenüberstellung zur Betrachtung beider Autonomen Systeme in einer Simulation. Diese Simulation heißt *vierjahre250680* und wird als manuell angeordnete graphische Repräsentation in Abbildung 13 gezeigt. Beide Simulationen befinden sich auf dem dieser Arbeit beiliegenden Datenträger im Verzeichnis *projekte* im jeweils nach ihnen benannten Projektverzeichnis.

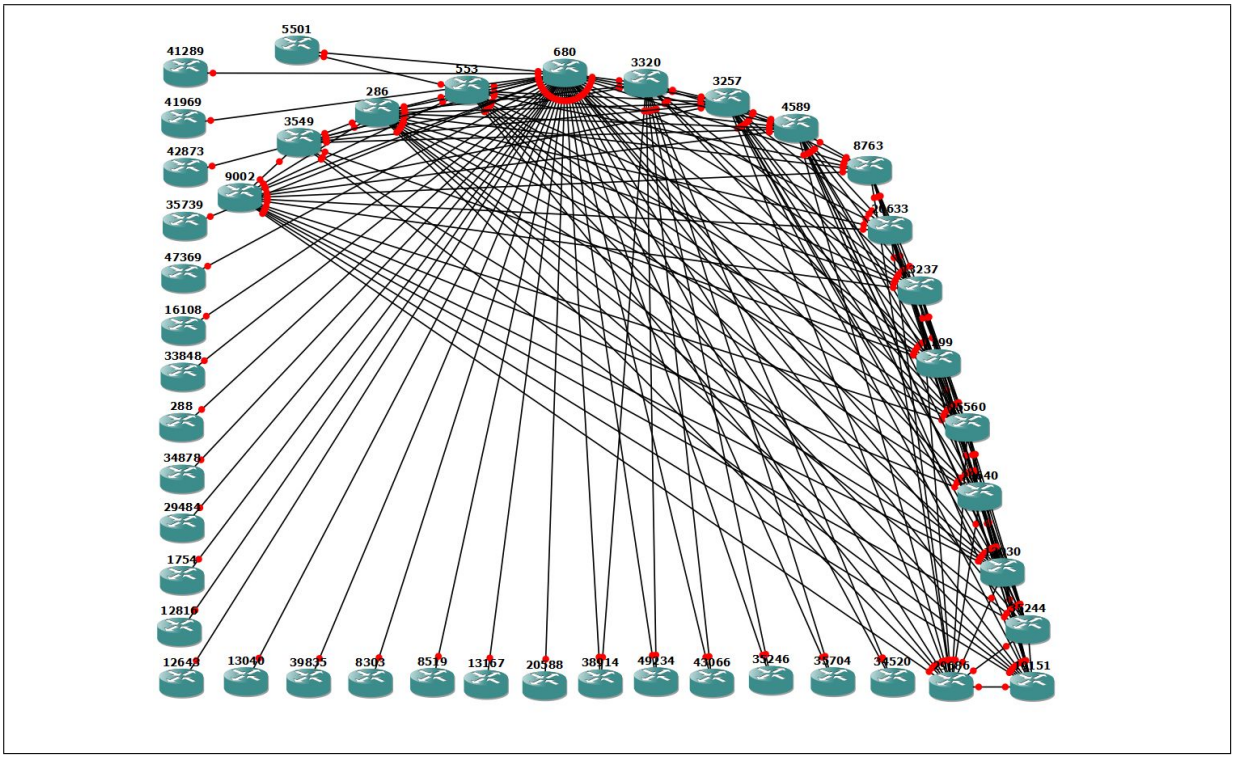

Abbildung 12: Graphische Repräsentation der Simulation *vierjahre680*, manuell angeordnet

In Abbildung 13 lässt sich die Abbildung 12 wiedererkennen, ergänzt um die Abbildung 6 links unten und die Verbindungen zwischen den beiden Abbildungen, die die Pfade zeigen, die Daten zwischen dem Autonomen System 250 und 680 zu dem Zeitpunkt nehmen konnten.

Die Simulationen *vierjahre680* und *vierjahre250680* sind bereits so groß, dass die Hardware, die zur Simulation genutzt wurde überlastet war. Der Startvorgang der Simulationen dauert über eine halbe Stunde und lastet bereits nach kurzer Zeit den Speicher vollständig aus. Zusätzlich ist auch die CPU sofort so überlastet, dass nach dem Start, der etwa eine halbe Stunde dauert, eine produktive Arbeit mit den Simulationen nicht möglich ist.

Hier wurde aber gezeigt, dass ARDA in der Lage ist so große Simulationen zu erstellen, die lauffähig sind, wenn die Hardware dies verkraftet. Weiterhin wurde gezeigt, dass auch die

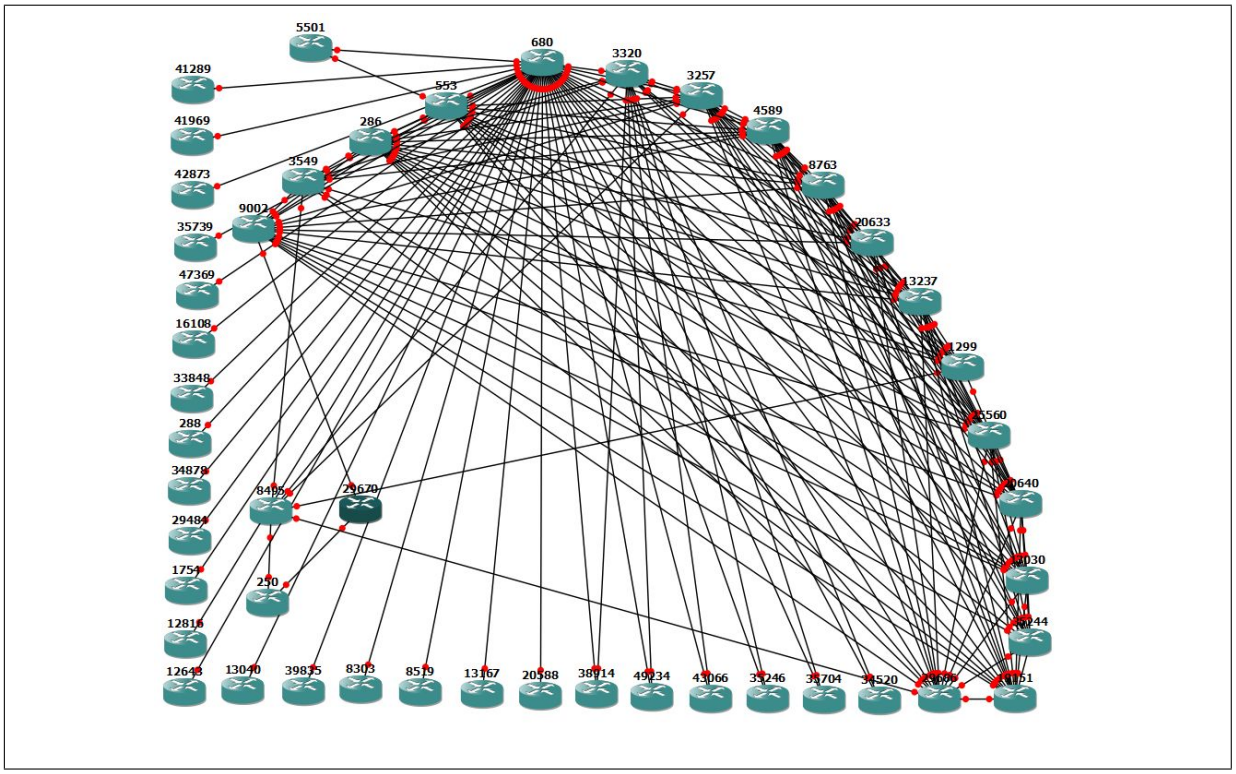

Abbildung 13: Graphische Repräsentation der Simulation *vierjahre250680*, manuell angeordnet

Betrachtung von zwei unterschiedlichen Autonomen Systemen möglich ist.

4.5 Aufgetretene Probleme

Im Zuge der Evaluation der Software ARDA ist oft das Problem der Größe der Simulation aufgetreten. Zum Einen wird diese beschränkt durch die maximale Anzahl der virtuellen Netzwerkports, die jeder Router besitzen kann, und zum Anderen durch die Gesamtanzahl der Router, die simuliert werden können, ohne dass die Hardware, auf der die Simulation läuft, so überlastet ist, dass damit nicht mehr zu arbeiten ist. Wie das Problem der maximalen Anzahl der Netzwerkports zu lösen sein könnte, wird in Abschnitt 5.5 behandelt. Ob das Problem der überlasteten Hardware durch stärkere Hardware zu lösen ist, hängt von der Skalierbarkeit von GNS3 und im Speziellen vom Emulator Dynamips ab. Wenn diese beliebig viele Prozessoren und Speicher benutzen können, ist es möglich große Simulationen auf starken Servern zu starten. Sollte auch das nicht ausreichen, ist die Lösung, die in Absschnitt 5.6 beschrieben wird, anzustreben.

5 Ausblick

In den nachfolgenden Abschnitten werden verschiedene Ideen beschrieben, mit denen man die Software verbessern könnte, zu deren Umsetzung aber im Rahmen dieser Bachelorarbeit die Zeit fehlte. Zur Umsetzung dieser Ideen fehlte die Zeit nicht nur wegen der zur Implementierung notwendigen Zeit, sondern bei manchen auch wegen umfangreicher Grundlagen, in die sich zusätzlich einzuarbeiten den Zeitrahmen allein schon gesprengt hätte.

5.1 Graphische Oberfläche

Die Bedienung der Software ließe sich durch Entwicklung eines Graphischen User Interfaces (GUI) erheblich vereinfachen. So ist es denkbar die Routen Kollektoren nach ihrem Standort auszuwählen, und verschiedene Anomalien auf einer Timeline anzuklicken. Diese Timeline könnte von BART [9] mit möglicherweise interessanten Punkten geladen werden, die der Nutzer dann auswählen kann. Außerdem ist eine Erweiterung der Daten mit geographischen Werten denkbar, sodass sich auch die Darstellung der einzelnen Autonomen Systeme, zumindest soweit möglich, an ihrem geographisch richtigem Ort auf dem Hintergrund einer Weltkarte befinden würde. Zusätzlich ist es sicher wünschenswert mehrere graphische Repräsentationen von vor einer Anomalie, währenddessen und von einem Zeitpunkt hinterher in unterschiedlicher farbiger Darstellung halbtransparent übereinander zu legen und die Unterschiede hervorzuheben. Auch eine optische Hervorhebung von Anomalien wäre ein wünschenswertes Feature, um die Identifikation dieser zu vereinfachen. Wenn diese dann noch zwischen Konfigurationsfehler, Fehlfunktion, beabsichtigter Anomalie und Angriff zu unterscheiden sind, würde die Usability der Software stark verbessert. Diese Erweiterungen könnten die Bedienbarkeit und den Nutzen der Software erheblich erhöhen, jedoch ist die klassifikation selbst ein komplexes Gebiet das den Umfang einer Bachelorarbeit erheblich überschreitet.

5.2 Anordnung der Knoten in der graphischen Repräsentation

Zur Vereinfachung der Arbeit mit der Software ist es hilfreich, wenn die graphische Anzeige der Knotenpunkte so geordnet ist, dass es möglichst wenige sich überschneidende Kanten zwischen den einzelnen Knoten gibt. Also das der Graph aller Kanten und Knoten planar ist. Dies ist mit wachsender Größe des Graphen ein zunehmend komplexes Vorhaben, zu dem es in der Literatur diverse Ansätze gibt, in die man sich ausführlich einarbeiten müsste, um dies umzusetzen.

5.3 Vereinfachung des Downloads

In der vorliegenden Version der Software ist es notwendig, die Routingdaten selbst vor der Verarbeitung von ris.ripe.org [12] herunterzuladen. Dies ließe sich vereinfachen, indem ein Modul geschrieben wird, dass alle Daten, die von allen Routenkollektoren zu einem vom User angegebenen Zeitpunkt vorliegen, herunterlädt. Dazu muss für jeden Routenkollektor die letzte Fulltable Datei vor dem angegebenen Zeitpunkt gefunden und geladen werden. Weiterhin müsste jede Update Datei zwischen dem Fulltable und dem angegebenen Zeitpunkt geladen werden, um die Daten zu vervollständigen.

5.4 Präzessierung des Zeitpunkts der Simulation

Es würde eine Verbesserung darstellen, wenn sich ein genauer Zeitpunkt angeben ließe, an dem das Einlesen in einer Update Datei gestoppt wird. Die vorliegende Version der Software liest immer ganze Updatefiles ein, was durch die Rasterung der Daten durch das ris.ripe Projekt [12] zu einem Raster von fünf Minuten führt, dass sich nicht verändern lässt. Zur genaueren Analyse von Anomalien im Routing wäre es jedoch sinnvoll dieses Raster zu eliminieren, indem beim Parsen von Update Dateien auf den Zeitstempel geachtet wird, und nur Updates eingelesen werden, die vor oder genau an einem spezifizierten Zeitpunkt stattgefunden haben.

5.5 Große Anzahl von Verbindungen pro Router

Um große Simulationen darstellen zu können muß, abgesehen von einer sehr leistungsfähigen Hardware, jeser Router in der Lage sein mehr als die aktuellen 48 Verbindungen aufbauen zu können. Die lässt sich ändern, indem jeder Router einen Switch zur Seite gestellt bekommt, den man in der graphischen Representation eine Ebene unter dem Router anordnen könnte, um die Struktur nicht noch weiter zu komlizieren. Die Switches in GNS3 sind in der Lage beliebig viele Ports anzusteuern und würden so auch die Hardwareemulation jedes Routers zu verringern, da nur eine Steckkarte mit einem Netzwerkport emuliert werden muss. Ob das jedoch die Hardware Anforderungen senkt muss nach der Implementierung dieses Features überprüft werden, da auch die Switches in GNS3 simuliert werden müssen.

5.6 Aufteilen in mehrere Simulationen

Ein weiterer Lösungsansatz zur Simulation von großen mengen von Routern ist das aufteilen auf mehrere Simulationen die auf unterschiedlichen Geräten ausgeführt werden und über ein reales Netzwerk verbunden sind. GNS3 ist in der Lage reale Netzwerkkarten in die simulierte Infrastruktur einzubinden. Wenn man dadurch mehrere Simulationen verbinden kann ist es möglich auch mit schwacher Hardware größere Netze zu simulieren. Die Aufteilung ist jedoch komliziert, da zu beachten ist, wie viele Daten über das Netzwerk geletiet werden müssen und wie viele virtuelle Verbindungen sich über ein realen Netzwerkanschluss verbinden lassen.

6 Fazit

In Zuge dieser Bachelorarbeit sollte eine Software entstehen, die BGP Routing Daten parst und analysiert, um daraus eine funktionierende Simulation zu erstellen. Diese Simulation soll in GNS 3 laufen und die Internet Routing Situation nachbilden um eine manuelle Analyse von Anomalien zu erleichtern. Die Software ARDA wurde mit diesem Ziel entwickelt und implementiert, wie im Kapitel 3 beschrieben. Diese Software erfüllt die Anforderungen die durch dieses Entwicklungsziel gestellt werden. Die durch ARDA erstellten Simulationen sind vollständig und funktional einsetzbar zur Analyse von Routing Anomalien. Das der Funktionsumfang der Software ARDA vollständig ist wird im Kapitel 4 gezeigt. Anschließend wird im Kapitel 5 dargestellt in welche Richtungen man ARDA weiterentwickeln kann, um den Funktionsumfang zu erhöhen und die Bedienung zu vereinfachen.

Die Anforderungen dieser Arbeit wurden umgesetzt und vollständig beschrieben. Alle Daten die zur Reproduktion der Ergebnisse notwendig befinden sich auf dem dieser Arbeit beiliegenden Datenträger. Zusammen lassen sich mit dieser Arbeit BGP Routingdaten aus verschiedenen Quellen leicht verarbeiten und analysieren.

7 Anhang

7.1 Datenträger

7.2 Softwaredokumentation

Reference Manual

Generated by Doxygen 1.6.1

Sun Sep 28 20:29:17 2014

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 bgpdata Class Reference

lesen und speichern der BGP Daten

`#include <bgpdata.h>`

Public Member Functions

- bgpdata ()

  Reserviert Speicher.

- ~bgpdata ()

  Gibt reservierten Speicher wieder frei.

- bool insertbview (BGPDUMP_ENTRY *entry)

  Fügt ein Fulltable in die Datenstruktur ein.

- bool **insertupdate** (BGPDUMP_ENTRY *entry)
- bool checkASN (unsigned long asn)

  returns if a given as is linked in this structure

- unsigned int getMapSize ()

  Gibt die Anzahl der Elemente in der map zurueck.

- std::vector< unsigned long > getLinkedASN (unsigned long asn)

  Gibt Eine Liste von links zurück, die mit einem bestimmten as bestehen.

- std::vector< PREFIX > * getPrefixes (unsigned long asn)

  Gibt eine Liste aller Prefixe zurück die mit dem angegebenen AS peeren.

Protected Member Functions

- std::vector< unsigned long > * createVectorLong (std::string *aspath)

creates a Vector of all ases in the given as path

Protected Attributes

- std::map< unsigned long, rdt * > * **linkedAS**

2.1.1 Detailed Description

lesen und speichern der BGP Daten

Author:

Thomas Trimborn

Version:

1.0

Date:

28.09.2014

2.1.2 Constructor & Destructor Documentation

2.1.2.1 bgpdata::bgpdata ()

Reserviert Speicher. Konstructor

2.1.2.2 bgpdata::~bgpdata ()

Gibt reservierten Speicher wieder frei. Destruktor

2.1.3 Member Function Documentation

2.1.3.1 bool bgpdata::checkASN (unsigned long *asn*)

returns if a given as is linked in this structure

Parameters:

unsigned long asn as number to check

Returns:

if the given asnumber is linked in the intern map

2.1.3.2 std::vector< unsigned long > * bgpdata::createVectorLong (std::string * *aspath*) [protected]

creates a Vector of all ases in the given as path

Parameters:

string *aspath pointer to string of aspath

Returns:

pointer to vector with ases

2.1.3.3 std::vector< unsigned long > bgpdata::getLinkedASN (unsigned long *asn*)

Gibt Eine Liste von links zurück, die mit einem bestimmten as bestehen.

Parameters:

unsigned long asn as nummer zu der die links gesucht sind

Returns:

liste der verlinkten ase

2.1.3.4 unsigned int bgpdata::getMapSize ()

Gibt die Anzahl der Elemente in der map zurueck.

Returns:

anzahl der maps

2.1.3.5 std::vector< PREFIX > * bgpdata::getPrefixes (unsigned long *asn*)

Gibt eine Liste aller Prefixe zurück die mit dem angegebenen AS peeren.

Parameters:

unsigned long asn AS Nummer zu der die Prefixe gesucht werden

Returns:

Liste der Prefixe

2.1.3.6 bool bgpdata::insertbview (BGPDUMP_ENTRY * *entry*)

Fügt ein Fulltable in die Datenstruktur ein. insertbview

Extrahiert aus einem Eintrag eines fulltable prefix, origin, aspfad um diese Daten dann in die Datenstruktur in rdt zu speichern

Parameters:

BGPDUMP_ENTRY ∗entry Aktueller Eintrag der abgearbeitet werden muss

Returns:

true wenn aufgabe erfüllt sonst false

The documentation for this class was generated from the following files:

- bgpdata.h
- bgpdata.cpp

2.2 input Class Reference

öffnen und parsen der bgp datensätze.

`#include <input.h>`

Public Member Functions

- bgpdata * read_data (int argc, char *argv[ ])

  Alle Dateien öffnen und parsen.

- std::vector< unsigned long > create_grid (int gridsize, bgpdata *inputdata, unsigned long asn1, unsigned long asn2=0)

  Benötigte ASe raussuchen und speichern.

2.2.1 Detailed Description

öffnen und parsen der bgp datensätze.

Author:

Thomas Trimborn

Version:

1.0

Date:

28.09.2014

2.2.2 Member Function Documentation

2.2.2.1 std::vector< unsigned long > input::create_grid (int *gridsize*, bgpdata * *inputdata*, unsigned long *asn1*, unsigned long *asn2* = 0)

Benötigte ASe raussuchen und speichern.

Parameters:

int gridsize wie viele hops vom ausgangs as

bgpdata *inputdate alle eingelesenen Daten

unsigned long asn1 AS 1

unsigned long asn2 AS2

Returns:

Vektor mit allen beteiligten ASen

2.2.2.2 bgpdata * input::read_data (int *argc*, char * *argv*[ ])

Alle Dateien öffnen und parsen.

Parameters:

int argc anzahl der parameter

char *argv[] Parameter

Returns:

bgpdata Datenstruktur mit allen geparsten ASen, prefixen und Verbindungen

The documentation for this class was generated from the following files:

- input.h
- input.cpp

2.3 output Class Reference

Aus den Daten die Konfigurationsdateien erstellen.

`#include <output.h>`

Public Member Functions

- output ()

  Erstelle Netzwerkmasken von /0 bis /32.

- void write_output (bgpdata *inputData, std::vector< unsigned long > usedAS, int gridsize, unsigned long asn1, unsigned long asn2, std::string projectname)

Protected Member Functions

- void write_topology (bgpdata *inputData, std::vector< unsigned long > usedAS, int gridsize, unsigned long asn1, unsigned long asn2, std::string projectname)

  Die Konfiguration mit der Topologie für GNS3 erzeugen.

- bool allreadyEdited (int i, std::vector< unsigned long > usedAS, unsigned long asn)
- std::string getPortFromTargetAS (unsigned long asn, std::vector< unsigned long > usedAS, std::string source)
- bool isContained (unsigned long asn, std::vector< unsigned long > usedAS)
- void write_configs (std::vector< unsigned long > usedAS, std::string projectname, bgpdata *inputData)

  Schreibe Konfig Dateien für Cisco Router.

Protected Attributes

- std::map< unsigned long, std::vector< std::string > > **connections**
- unsigned int **console**
- std::vector< unsigned int > **slotcount**
- std::vector< std::string > **networkmask**

2.3.1 Detailed Description

Aus den Daten die Konfigurationsdateien erstellen.

Author:

Thomas Trimborn

Version:

1.0

Date:

28.09.2014

2.3.2 Member Function Documentation

2.3.2.1 bool output::allreadyEdited (int *i*, std::vector< unsigned long > *usedAS*, unsigned long *asn*) [protected]

Überprüft ob ein angegebenes Autonomes System bereits angelegt wurde in der Konfigurationsdatei für GNS3

Parameters:

int i Startpunkt ab dem usedAS durchsucht wird

std::vector<unsigned long> usedAS liste aller verwendeten AS

unsigned long asn gesuchtes as

Returns:

true wenn bereits angelegt, sonst false

2.3.2.2 std::string output::getPortFromTargetAS (unsigned long *asn*, std::vector< unsigned long > *usedAS*, std::string *source*) [protected]

Fügt ein port zu einem Autonomen System hinzu, das noch nicht bearbeitet wurde, um in dem gerade bearbeiteten AS den port angeben zu können der benötigt wird.

Parameters:

unsigned long asn as nummer von dem der port benötigt wird

std::vector<unsigned long> usedAS liste aller ase die verwendet werden um die position in der liste herauszufinden

std::string source quellport, der bei dem neuen as eingetragen werden muss

Returns:

string mit der angabe des ports zu dem das aufrufende as verbunden wird

2.3.2.3 bool output::isContained (unsigned long *asn*, std::vector< unsigned long > *usedAS*) [protected]

Überprüft ob ein bestimmtes AS in der übergebneen Liste enthalten ist

Parameters:

unsigned long asn Nummer des zu überprüfenden AS

std::vector<unsigned long> usedAS Liste aller verwendeter AS

Returns:

true wenn enthalten, sonst false

2.3.2.4 void output::write_configs (std::vector< unsigned long > *usedAS*, std::string *projectname*, bgpdata * *inputData*) [protected]

Schreibe Konfig Dateien für Cisco Router.

Parameters:

std::vector<unsigned long> usedAS Alle beteiligten ASe

std::string projectname Name des Projekts

bgpdata *inputData Alle Daten

2.3.2.5 void output::write_output (bgpdata * *inputData*, std::vector< unsigned long > *usedAS*, int *gridsize*, unsigned long *asn1*, unsigned long *asn2*, std::string *projectname*)

Einzige Public Funktion von output, die erst die Ordnerstruktur anlegt und dann das erzeugen der Topologiedaten und der Konfig Daten anstösst

Parameters:

bgpdata * inputData Alle von input eingelesenen BGP Daten

std::vector<unsigned long> usedAS Alle AS die in der Simulation vorkommen

int gridsize Maximaler Abstand der AS vom ausgangspunkt

unsigned long asn1 AS Nummer 1

unsigned long asn2 AS Nummer 2

std::string projectname Name des Projektverzeichnisses

2.3.2.6 void output::write_topology (bgpdata * *inputData*, std::vector< unsigned long > *usedAS*, int *gridsize*, unsigned long *asn1*, unsigned long *asn2*, std::string *projectname*) [protected]

Die Konfiguration mit der Topologie für GNS3 erzeugen.

Parameters:

bgpdata *inputdata Alle Datensätze

std::vector<unsigned long> usedAS Alle beteiligten AS

int gridsize Anzahl der hops

unsigned long asn1 AS1

unsigned long asn2 AS2

std::string projectname Name des Projekts

The documentation for this class was generated from the following files:

- output.h
- output.cpp

Literatur

[1] So funktioniert Internet-Routing.
http://www.heise.de/netze/artikel/So-funktioniert-Internet-Routing-221495.html, 2008.

[2] About Prefix Hijacking in the Internet.
https://net.cs.uni-bonn.de/de/wg/cs/mitarbeiter/alumni/bornhauser/, 2011.

[3] Automatic Analysis and Classification of Multiple Origin AS (MOAS) Conflicts.
http://www.ieeelcn.org/prior/LCN36/lcn36demos/lcn-demo2011_bornhauser.pdf, 2011.

[4] libBGPdump.
https://bitbucket.org/ripencc/bgpdump/wiki/Home, 2011.

[5] Autonomes System.
http://de.wikipedia.org/wiki/Autonomes_System, 2014.

[6] Basic BGP.
http://www.getnetworking.net/bgp/basic-bgp, 2014.

[7] Cisco Router Konfiguration.
http://www.cisco.com/c/en/us/td/docs/routers/7200/roadmaps/7200_series_doc_roadmap/3512.html, 2014.

[8] Graphical Network Simulator 3.
http://www.gns3.net, 2014.

[9] MonIKA, Monitoring durch Informationsfusion und Klassifikation zur Anomalieerkennung.
https://net.cs.uni-bonn.de/wg/itsec/projects/monika/, 2014.

[10] Oracle VirtualBox.
https://virtualbox.org, 2014.

[11] Peering Database.
http://www.peeringdb.com/view.php?asn=250, 2014.

[12] RIPE Network Coordiation Centre, Routing Information Service.
https://www.ripe.net/data-tools/stats/ris/, 2014.

[13] Through the Looking-Glass, and What Eve Found There.
http://www.heise.de/newsticker/meldung/Def-Con-22-Kritische-Internet-Infrastruktur-du
html, 2014.

[14] Zebra.
https://www.gnu.org/software/zebra/, 2014.

[15] M. Meier M. Wübbeling, T.Elsner. Inter-AS Routing Anomalies: Improved Detection and Classification. 2014.

[16] Iljitsch van Beijnum. *BGP*. O'REILLY, 2002.

[17] S. Hares Y. Rekhter, T. Li. Border Gateway Protocoll.
http://tools.ietf.org/html/rfc4271, 2006.