EasyPeasyVPN

Bachelorthesis

Titel:	**A Tool For Creating Ad-Hoc-VPNs**
Stand:	**18.09.17**
Name:	**Kai Hartz, 420583**
Studienfach:	**Informatik**
Thesis Zeitraum:	**15.08 - 18.09.2017**
Betreut durch:	**Dr. Lammers**

Abstract

Netzwerke werden in der Informatik sehr häufig angewendet. Die Kommunikation über Diese sollte dabei sicher und vertraulich sein. Sie bieten in ihrer physikalischen Form nicht immer die gewollte direkte Adressierbarkeit in bestimmte Teilnetze. Eine gute Möglichkeit hierfür bieten virtuelle private Netzwerke. Allerdings sind diese recht kompliziert und erfordern meist einiges an Zeit und Aufwand zur Einrichtung. Zudem sind sie selten benutzerfreundlich.

Anwendungen wie „LogMeIn Hamachi“[1] oder „Tungle“[2] ermöglichen bereits schnelle und einfache End-zu-End-Netzwerke. Diese sind jedoch proprietäre Produkte, das heißt, es ist nicht bekannt, was mit den Daten weiter geschieht beziehungsweise wie gut diese gesichert und transportiert werden.

Daher möchte ich mich in meiner Bachelorarbeit mit der zugrunde liegenden Technik vertraut machen und eine vergleichbare, offene und konfigurierbare Software entwickeln. Die Entwicklung dieser Software endet nicht mit dieser Arbeit und bietet zunächst nur grundlegende Funktionen.

1 https://www.vpn.net

2 https://www.tunngle.net/de/features

Inhaltsverzeichnis

Abstract....2
Einleitung....6
1. Grundlagen von Virtual Private Networks....7
1.1 Grundlagen von Computernetzen....7
1.1.1 IP....7
1.1.2 DHCP....8
1.1.3 OSI Schichtenmodell....8
1.2 Was ist ein VPN....10
1.3 VPN Arten....11
1.3.1 End to Site....11
1.3.2 End to End....11
1.3.3 Site to Site....11
1.3.4 Geschlossener Tunnel....11
1.3.5 Geteilter Tunnel....12
1.4 Protokolle....12
1.4.1 PPTP....12
1.4.2 L2TP....13
1.4.3 IPSec....13
1.4.4 TCP und UDP....14
1.5 VPN Anwendungsfälle....14
1.6 Bestehende Konzepte....15
1.6.1 Hamachi....15
1.6.2 Tunngle....17
1.6.3 OpenVPN....18
1.6.4 Alternativen....18
1.7 Docker....19

Abstract

1.8 GitHub....20

1.9 Sicherheitsaspekte....20

1.9.1 SSL / TLS....20

1.9.2 RSA....21

1.9.3 Zwei Faktor Authentifizierung....22

2. Umsetzung der Software....23

2.1 Definition des Funktionsumfangs....23

2.1.1 Client....23

2.1.2 Server....24

2.2 Technische Grundlagen der Anwendung....25

2.2.1 Zielsysteme....25

2.2.2 Programmiersprache....25

2.2.3 Argumentation und Wahl des Adapters....26

2.2.4 Entwicklungsumgebung....27

2.3 Implementierung und Beispielcode....28

2.3.1 Designentscheidungen....28

2.3.2 Einrichtung OpenVPN....29

2.3.3 Client-Server-Architektur....31

2.3.4 Betriebssystemunabhängigkeit....34

2.3.5 Persistenz....34

2.4 Projektstruktur und Anwendungsbeispiele....35

2.4.1 Projektstruktur....35

2.4.2 Anwendungsszenarien....35

2.5 Allgemeine Einrichtung der Anwendung....36

2.6 Exemplarische Beispielumgebung....36

2.6.1 Einrichtung des Servers....36

2.6.2 Einrichtung des Clients....37

2.7 Analyse und Tests....38

Abstract

2.7.1 Performance / Delay 38

2.7.2 Sicherheit 39

3. Abschluss 40

3.1 Veröffentlichung auf GitHub 40

3.2 Dokumentation 41

3.3 Ausblick 41

3.4 Kritische Auseinandersetzung und Fazit 42

Glossar 44

Abkürzungsverzeichnis 46

Literaturverzeichnis 47

Abbildungsverzeichnis 53

Anhang 54

1 Klassendiagramm VPN Konfiguration 54

2 Beispielkonfiguration Router Portforwarding 55

3 OpenVPN Windows Installation 56

Einleitung

Der erste Teil dieser Bachelorarbeit befasst sich mit dem Thema „Virtual Private Network“ (VPN). Zunächst folgt eine Erläuterung der grundlegenden Begriffe sowie zugrunde liegender Techniken. Dabei soll auf verschiedene bereits existierende Implementierungen eingegangen werden.

Abhängig von dieser Analyse wird ein VPN-Adapter ausgewählt und basierend darauf eine Client- und Server-Anwendung geschrieben, die es dem Nutzer erlaubt einfach und unkompliziert den Adapter zu installieren und eine grafische Benutzerschnittstelle zur Konfiguration des Netzwerkes bietet. Die Programmiersprache wird erst nach der Auswahl des Adapters bestimmt, abhängig davon, ob die Anwendung plattformunabhängig sein kann und ob es Einschränkungen bei der Performance oder Sicherheit gibt.

Grundlegende Eigenschaften der Software sollen die benutzerfreundliche Bedienung und Einrichtung sowie Sicherheit, Transparenz und Privatisierung sein. Das Programm soll auf GitHub veröffentlicht werden um Transparenz zu gewährleisten. Es soll möglich sein ein VPN zu erstellen und zu hosten, sodass keine Daten von Drittanbietern beeinflusst werden können. Es sollen grundlegende Sicherheitskonzepte angewandt werden, wie z.B. Verschlüsselung zur Geheimhaltung, Authentizität, via Zertifikaten, und auch mehrerer Kommunikationsebenen, zur Sicherung und Vereinfachung der Verbindung.

Die Art des VPNs (End-to-End, End-to-Site oder Site-to-Site) kann von der Wahl des Adapters abhängen. Zunächst soll nur eine Variante umgesetzt werden. Die Wahl soll erst konkret werden, sobald der primär unterstützte Adapter feststeht. Über die Umsetzung dieser Software, im weiteren „Easy-Peasy-VPN“ benannt, wird es im zweiten Teil der Arbeit gehen.

Der dritte Teil wird der Abschluss mit einer letzten kritischen Bewertung sein. Hier soll auch darauf eingegangen werden, welche Funktionalitäten noch vervollständigt werden müssen. Außerdem wird auf die grundlegende Bedienung der Software eingegangen und auf entsprechende Dokumente, wie etwa Handbuch etc., verwiesen.

1. Grundlagen von Virtual Private Networks

1.1 Grundlagen von Computernetzen

Ein Netzwerk besteht aus mehreren Klienten. Diese sind alle unter einander verbunden. Sie können Daten untereinander empfangen und senden.

Die Topologie beschreibt die Art wie die Klienten miteinander verbunden sind. Oft gibt es einen zentralen Knotenpunkt über den alle Computer miteinander verbunden sind. Dies nennt man die Stern-Topologie [1]. Das wohl größte bekannte Rechnernetz ist das Internet. Es ist mit vielen kleinen lokalen Netzen verbunden. Der Router bildet dabei den zentralen Knotenpunkt des lokalen Netzes, der gleichzeitig der Zugangspunkt für das Internet ist. So kann jeder Klient nur über den Router mit dem Internet kommunizieren [2]. Das lokale Netz wird Local Area Network genannt, kurz LAN.

Die Kommunikation in den Netzen basiert auf sogenannten Protokollen [3], um den Ablauf der Interaktion festzulegen.

Da einige Netze sehr groß werden ist es mittlerweile Standard Netze in Teilnetze, sogenannte Subnetze, zu unterteilen. Mittels NAT (Network Address Translation), also ein Mapping von Adressen auß- und innerhalb der verschiedenen Netze, kommunizieren die Teilnehmer eines Netzes über ein Gateway mit anderen Netzen. Das Gateway übernimmt dabei die Rolle eines Vertreters, der für die Netzteilnehmer die gewünschten Daten anfordert bzw. weiterleitet.

1.1.1 IP

Das Internet Protokoll, kurz IP, ist das Protokoll um Rechner zu adressieren. Die Adresse eines Rechners setzt sich aus drei verschiedenen Komponenten zusammen: Der IP-Adresse, der Subnetzmaske und dem Gateway. Es gibt aktuell zwei Versionen des Protokolls: IPv4 und IPv6. Der Anlass für IPv6 war, dass der Adressraum des IPv4 für das Internet zu klein wurde. Die Adresse besteht hier aus vier Bytes deren textuelle Repräsentation der Zahlenwerte jeweils durch einen Punkt getrennt werden. Beispielsweise 192.168.2.1. Das IPv6-Protokoll hat eine 16-Byte-Adresse, in der immer zwei Bytes in ihrem Hexadezimalwert mit einem Doppelpunkt getrennt werden.

Die Subnetzmaske wird mit der Adresse bitweise über eine Und-Verknüpfung ausgewertet und ergibt so die Netzadresse. Diese beschränkt den möglichen Adressraum in einem LAN.

Es wird die IP-Adresse des Gateways benötigt, also des Gerätes, der Informationen für Netze hinter dem LAN sammelt und bereitstellt [4].

1.1.2 DHCP

Das Dynamic Host Configuration Protocol, kurz DHCP, ist ein Protokoll, um IP-Adressen in einem Netzwerk zu verwalten und an die anfragenden Hosts sinnvoll zu verteilen. Durch DHCP ist jeder Netzwerk-Teilnehmer in der Lage sich eine vorgegebene Konfiguration zu laden.

Damit ein Teilnehmer erfolgreich kommunizieren kann, werden ihm folgende Informationen geliefert, seine eindeutige IP-Adresse , seine Subnetzmaske, die IP-Adresse des Standard-Gateway und die IP-Adresse des DNS-Servers [5]. Diese Informationen könnten sonst auch manuell eingetragen werden.

1.1.3 OSI Schichtenmodell

Kommunikation über Netzwerke läuft über verschiedene Medien und auch die Teilnehmer in einem Netzwerk sind meistens nicht gleich. Dabei beeinflussen unterschiedliche Software und Hardware die Geschwindigkeit und Latenz, also die Zeit bis auf eine Anfrage eine Antwort erfolgt, damit dennoch Datenaustausch stattfinden kann, wurde ein Modell erstellt, das etwa eine Norm der einzelnen Teilschritte darstellt. Das wohl verbreitetste Modell ist das OSI Schichtenmodell.

Die Aufgaben des Modells wurden in möglichst kleine Partitionen, genauer gesagt Schichten, unterteilt. Man nennt sie Schichten (Englisch: Layer), da Informationen nur Top-Down und Bottom-Up an die benachbarten Schichten weitergegeben werden.

Wie der Begriff Modell schon sagt, ist dies ein Ideal. Das bedeutet, dass die konkreten Umsetzungen von der eigentlichen Definition abweichen können. Da es in der praktischen Anwendung gelegentlich dazu kommt, dass die Theorie nicht die optimale Effizienz bietet.

Das OSI-Modell für Netzwerke bietet eine solche Separierung und Abstraktion.

Einer Schicht wird eine feste und eindeutige Funktion zugeteilt. Somit können die konkreten Umsetzungen basierend auf verschiedenen Standards erstellt werden und dennoch mit ihren Nachbarschichten effizient und fehlerfrei arbeiten.

In der Praxis sind die Grenzen jedoch eher fließend. Das Modell hat insgesamt sieben Schichten.

1. Physical Layer / Bitübertragungsschicht

Im Physical Layer werden die physikalischen und technischen Eigenschaften des Transport-Mediums definiert.

2. Link Layer / Sicherungsschicht

Im Link Layer findet die Fehlererkennung und Synchronisation der Datenübertragung statt. Dadurch werden trotz möglicher Störungen die Daten fehlerfrei übertragen.

3. Network Layer / Vermittlungsschicht

Im Network Layer geschieht das Routing: Die Daten werden in in die richtige Richtung geschickt.

4. Transport Layer / Transportschicht

Im TransportLayer findet die Flusskontrolle- und Fehlerkontrolle statt. Hier werden die Daten in Pakete unterteilt bzw. beim Empfänger zusammengeführt.

5. Session Layer / Sitzungsschicht

Der Session Layer regelt den Auf- und Abbau der Kommunikation, sowie die Wiederherstellung der Verbindung nach Störungen in den vier unteren Ebenen.

6. Presentation Layer / Datendarstellungsschicht

Im Presentation Layer geschieht die Vereinbarung über Datenformat und -codierung. Auf dieser Schicht findet auch z.B. Kodierung, Kompression, Kryptographie statt.

7. Application Layer / Anwendungsschicht

Die Anwendungsschicht stellt den Anwendungen die Funktionalität "Kommunikation" zur Verfügung.

Diese Zusammenfassung der einzelnen Schichten ist ein Zitat, da es im Umfang und Aussagekraft sehr gut passt [6]. Der praktische Ablauf dieses Modells wird in der Grafik noch einmal verdeutlicht..

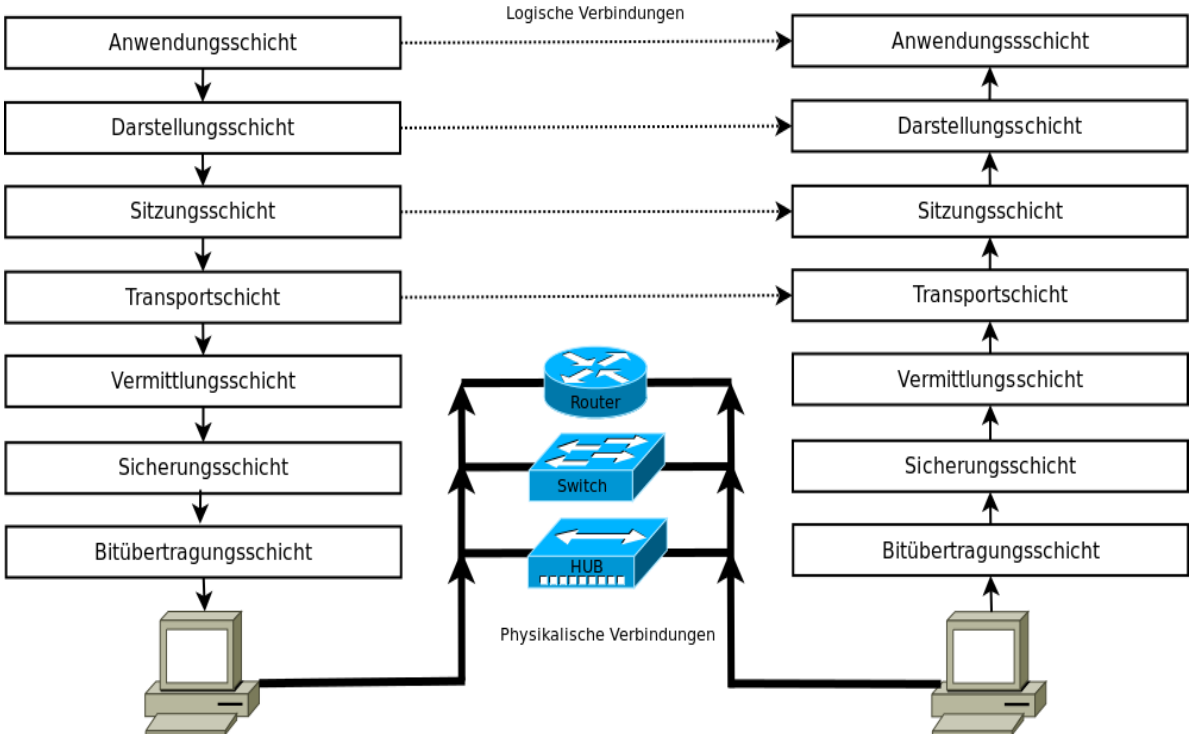


Abbildung 1: Der Ablauf der Kommunikation zwischen zwei Rechnern über OSI

1.2 Was ist ein VPN

Ein Virtual Private Network ist ein rein logisches Netzwerk. Es läuft über ein physikalisches Netzwerk indem jeder Teilnehmer zumindest theoretisch dem virtuellen Netzwerk beitreten kann.

Dabei ist es wichtig, dass Authentizität, Vertraulichkeit und Integrität sichergestellt werden, damit der Informationsaustausch auch nur die gewünschten Teilnehmer erreicht.

Authentizität bedeutet, dass jeder Teilnehmer in dem Netzwerk sich ausweisen und identifizieren kann, dass er autorisiert ist das Netzwerk zu nutzen. Und das alle Daten von ihm kommen. Vertraulichkeit heißt in diesem Zusammenhang, dass Daten vor lesenden Zugriffen Dritter geschützt sind. Durch Ver- bzw. Entschlüsselung wird also gewährleistet, dass die Daten nur von den spezifizierten Kommunikationspartnern gelesen werden können. Integrität gewährleistet, dass die Daten während der Übertragung nicht verändert werden, sollten Daten Änderungen unterliegen soll dies erkannt werden können [7].

Die Verfügbarkeit ist auch wichtig, kann aber nur in Abhängigkeit der zugrundeliegenden Infrastruktur gewährleistet werden. Somit bedienen sich VPNs an den vier Grundlegenden Sicherheitsanforderungen [8].

1.3 VPN Arten

Es gibt verschiedene Möglichkeiten ein VPN aufzubauen. Welche genutzt werden sollte hängt von den Anforderungen an das Netz ab. Die Komplexität der Infrastruktur zwischen den VPN-Teilnehmern ist dabei nicht relevant [7][9].

1.3.1 End to Site

Diese Art beschreibt wie einzelne Teilnehmer aus verschiedenen Netzen durch das VPN in ein Physikalisches Netz eingegliedert werden. Beispielsweise können sich so Mitarbeiter eines Unternehmens von außen über das Internet in das firmeneigene Intranet bzw. LAN einklinken und sind somit von allen Teilnehmern in dem LAN erreichbar, genauso wie sie alle Teilnehmer des LANs erreichen können. Die Anzahl an Teilnehmern in dem VPN ist somit größer gleich der Teilnehmeranzahl im LAN.

1.3.2 End to End

In diesem Fall besteht das VPN nur aus virtuellen Netzwerkteilnehmern. Das VPN Netz hat somit keinerlei Beziehung zur tatsächlichen physikalischen Grundlage. Keiner der VPN-Teilnehmer ist, ohne weitere Einrichtungen, in der Lage Teilnehmer aus den LANs der anderen VPN-Teilnehmer zu erreichen.

1.3.3 Site to Site

Hier werden mehrere physikalisch vorhandene Netzwerke miteinander verbunden und bilden so ein einzelnes gesamtes Netz. Dabei ist es im Normalfall so, dass die beiden LANs nicht direkt miteinander verbunden sind, sondern ein Extranet, wie beispielsweise das Internet, dazwischen liegt.

Die Anzahl Teilnehmer im VPN ist somit exakt gleich zu der Summe der Teilnehmer der einzelnen Netze.

1.3.4 Geschlossener Tunnel

Der Tunnel bezeichnet hierbei die Verbindung vom VPN-Teilnehmer zum VPN-Gateway bzw. Server.

Ein geschlossener Tunnel (Englisch: closed tunnel) heißt, dass alle Verbindungen des VPN-Teilnehmers über das VPN-Gateway aufgebaut werden. Andere

Verbindungen über das Default-Gateway sind somit nicht mehr möglich und verweigern einem so den Zugriff auf Teilnehmer des LANs des VPN-Teilnehmers. Man beachte das die VPN-Verbindung natürlich dennoch über das Default-Gateway aufgebaut wird. Allerdings sind alle sonstigen Daten dann über das VPN-Verschlüsselt. Anfragen an das Internet werden nun vom VPN-Server aus weitergeleitet beziehungsweise angefragt. Somit sieht es für außenstehende Provider so aus als würde der VPN-Server bzw. dessen Gateway die Verbindung aufbauen wollen. Dies dient der Anonymisierung der VPN-Teilnehmer.

1.3.5 Geteilter Tunnel

Der geteilte Tunnel (Englisch: split tunnel) erlaubt den Zugriff auf mehrere Netzwerke gleichzeitig, sodass man beispielsweise den Zugriff auf das eigene und ein weiteres LAN hat. Das hat zum einen den Vorteil beide Netze erreichen zu können und zum anderen kann in bestimmten Fällen eine bessere Geschwindigkeit und Performance erreicht werden, denn falls man beispielsweise ins Internet möchte, ist dies ohne den Umweg über das VPN-Gateway direkt über das eigene Gateway möglich. Zudem wird bei dieser Kommunikation im Normalfall auf Verschlüsselung verzichtet, wodurch Performance gewonnen wird.

Bei DNS-Anfragen kann es ungewollt dazu kommen, dass man Informationen in beide Netze Preis gibt, da die DNS-Anfrage ggf. erst in das Netz geleitet wird das keine Auflösung bietet.

Anwendung findet diese VPN-Art oft in Firmen, wenn die Mitarbeiter Zugriff auf das LAN der Firma benötigen, da so keine Einschränkungen an Erreichbarkeit und Geschwindigkeit gemacht werden. Der Faktor Privatsphäre ist hier eher unwichtig.

1.4 Protokolle

1.4.1 PPTP

Das Point-to-Point-Tunneling-Protocol bezeichnet ein Protokoll, das seit ca. 1996 verfügbar ist. Meistens wird es genutzt, um ein End-to-End-VPN zu erstellen, gelegentlich auch für End-to-Site [10].

Das Protokoll selbst legt keine konkreten Sicherheitsmechanismen fest. Es können verschiedene Verschlüsselungen und Authentifizierungsverfahren genutzt werden [11].

Das Protokoll wurde in allen größeren Betriebssystemen (z.B. Windows, Linux, MacOS) integriert und ist somit weitestgehend plattformunabhängig. Jedoch ist durch das Alter bedingt, die Sicherheit nicht mehr vollständig gewährleistet [12].

Die Quelle hat das MSCHAPv2 Authentifizierungsverfahren erfolgreich entschlüsselt.

1.4.2 L2TP

Das Layer-2-Tunneling-Protocol basiert auf Techniken von PPTP und L2F (Layer-2-Forwarding). Es ist ein unsicheres und veraltetes Protokoll, das von einer Gruppe an Unternehmen, wie etwa Microsoft, entwickelt wurde.

Das Protokoll selbst hat, wie PPTP, keine eigenen Verschlüsselungs- und Authentisierungsstandards. Es wird meistens mittels zuvor erstellten Benutzerkonten und Preshared-Keys die Authentifikation durchgeführt. Preshared-Keys ist ein Verfahren bei denen die Teilnehmer ihre Zugangsdaten vor dem Verbindungsaufbau bereits erhalten haben müssen. Die Verschlüsselung wird im Normalfall durch andere Protokolle ergänzt, was durchaus sinnvoll ist, da der Algorithmus so einfach ausgetauscht werden kann [13].

1.4.3 IPSec

IPsec (Internet Protocol Security) ermöglicht es dem Internet Protocol Verschlüsselung und Authentifikation zu nutzen. Im alten IPv4 Standard war dies ursprünglich nicht integriert, wird nun aber nachträglich auch unterstützt. IPsec wurde von der Internet Engineering Task Force (IETF) entwickelt. Die IETF ist eine offene Gruppierung die sich das Ziel gesetzt hat, Probleme rund um das Internet zu lösen und dazu passende Protokolle zu entwickeln.

IPsec ermöglicht folgende Dinge:

- Betriebssystemunabhängige Kommunikation
- Verschlüsselung
- Überprüfung der Übertragenen Daten auf Fehler
- Authentisierung
- Schlüsselmanagement, also den sicherem Austausch der Verschlüsselungsparameter

Das Protokoll arbeitet auf der Vermittlungsschicht des OSI-Models, also dem dritten Layer. Es ist möglich alle VPN-Arten mit diesem Protokoll umzusetzen. Da es auf einer relativ niedrigen Schicht arbeitet und daher recht performant ist, wird es oft für Site-to-Site Netze genutzt.[14]

Möchte man IPsec über verschiedene Subnetze hinweg verwenden, kommt man zu dem Problem, dass durch NAT die Daten neue Ursprungsadressen bekommen. Wird jedoch ein IPsec-Paket verändert, so wird dieses ungültig. Daher benötigt man ein Verfahren um dieses Problem zu umgehen. Weitverbreitet ist hierbei die Verwendung eines UDP-Wrappers, welcher das Daten-Packet in ein UDP-Paket verpackt. So kann das UDP-Paket verändert werden ohne, dass das IPsec-Paket ungültig wird. Ähnliches geschieht auch beim UDP-Hole-Punching, hierzu später mehr[15] [16] .

1.4.4 TCP und UDP

TCP (Transmission Control Protocol) und UDP (User Datagram Protocol) sind Protokolle die auf dem vierten Layer des OSI-Modells arbeiten. Sie sind die Grundlage für jede Art von Kommunikation, die Anwendungen benutzen. Bei VPNs kann manchmal auch gewählt werden welches man verwenden möchte. Hier reicht es aus die grundlegenden Vorteile der beiden Protokolle zu kennen.

Während TCP zuverlässige Kommunikation mit Flusskontroller ermöglicht, ist UDP schneller, da UDP auf eine solche Flusskontrolle verzichetet. So wird TCP beispielsweise im Browser zum Laden der Daten einer Website eingesetzt, da hier alle Daten notwendig sind um die angeforderte Seite korrekt anzeigen zu können. Die Ladezeit ist in diesem Beispiel zweitrangig.

Wohingegen für Internet und Videotelefonie UDP genutzt wird, hier sollen schnell relativ viele Daten übertragen werden während ein geringer Qualitätsverlust in Kauf genommen werden kann. [17]

1.5 VPN Anwendungsfälle

Es gibt verschiedene Anwendungsmöglichkeiten, für die man ein VPN nutzen kann.

1. Anwendungsfall: End-to-Site-Netzwerk

Ein Mitarbeiter benötigt Zugriff auf das Firmennetz, um Daten, die nur im LAN verfügbar sind, zu erhalten. Der Zugriff geschieht dabei von außerhalb über eine nicht vertrauenswürdige Infrastruktur, wie beispielsweise das Internet.

2. Anwendungsfall: Site-to-Site-Netzwerk

   Wenn Unternehmen sich beispielsweise an mehreren Standorten befinden, jedoch ein gemeinsames Netz benötigen, kann man das VPN dazu nutzen, mehrere Netze virtuell zu verbinden.

3. Anwendungsfall: End-to-End-Netzwerk

   Es soll ein vollkommen neues Netzwerk erstellt werden, das keinerlei Bezug zur realen Infrastruktur aufweist. Es sollen beliebige Teilnehmer ein eigenes virtuelles LAN aufbauen können. Dies kann dazu genutzt werden, um in bestimmten Gruppen einfacher Daten auszutauschen oder auch alte Computerprogramme, die nur LAN-fähig sind, wie etwa Computerspiele, wieder nutzen zu können [18].

4. Anwendungsfall: Anonymisierung

   Das VPN kann auch zur Anonymisierung genutzt werden. Mittels Closed Tunnel kann man verschleiern welche Anfragen von dem eigenen Rechner kommen. Dabei ist man jedoch niemals wirklich anonym. Das VPN-Gateway kann sowohl den Ursprung als auch den angefragten Inhalt speichern. Daher ist hier darauf zu achten, dass man dem Betreiber trauen kann. Nützlich ist dies beispielsweise, um den eignen Verkehr zu verschleiern und selber anonym zu bleiben.

1.6 Bestehende Konzepte

1.6.1 Hamachi

Hamachi [19] ist eine proprietäre VPN-Software, die für Windows und Linux verfügbar ist. Der Client bietet die Möglichkeit einfache End-to-End-Netzwerke aufzubauen. Es sind nur Netzwerke mit bis zu fünf Teilnehmern kostenlos möglich. Es gibt weitere Möglichkeiten, die aber kostenpflichtig sind. Es ist zudem eine Registrierung nötig, um den Dienst nutzen zu können.

Der genaue Ablauf des Verbindungsaufbaus und das Konzept hinter der Software ist nicht bekannt. Aber grundlegende Abläufe sind aus dem Jahre 2006 bekannt, deren technische Grundlage aber sehr wahrscheinlich noch aktuell ist.

Die Verbindung zwischen den Clients wird über einen Vermittler aufgebaut, den sogenannten Mediation Server. Dabei kümmert sich dieser um Identifikation, also Authentizität, und die Erstellung und den Austausch der Schlüssel. Die genauen technischen Spezifikationen kann man im Internet nachlesen [20][21]. Wichtige Schlagwörter sind hier etwa RSA Schlüsselpaare, Diffie-Hellman-Schlüsselvereinbarung, AES-256-CBC Verschlüsselung und Encapsulating Security Payload (ESP). Diese werden später genauer Erklärt oder nur als Vergleich zu den anderen Produkten verwendet.

Auch bei dieser Software gibt es Problematiken mit der NAT. Im Normalfall akzeptiert das Gateway nur Verbindungen, die vorher aus dem LAN aufgebaut wurden. Somit kann keiner der Clients in das Netz des anderen gelangen. Deswegen wird zunächst eine UDP-Verbindung von den Clients aus zu dem Mediation-Server aufgebaut, da dieser eine statische bekannte Adresse hat. Die Verbindungsdaten der Clients reicht der Vermittler nun an den jeweils anderen Client weiter, so kann auch durch NAT eine Verbindung aufgebaut werden. Dies bezeichnet man auch als UDP-Holepunching, dabei beizeichnet UDP nur das Protokoll das für den Aufbau der Verbindung verwendet wird. Ähnlich ist dies auch mit TCP möglich, allerdings mit geringerer Erfolgswahrscheinlichkeit [16].

Nach der erfolgreichen Vermittlung läuft das VPN prinzipiell nur noch zwischen den Clients. Der Mediation-Server übernimmt nur noch Kontrollmechanismen, die greifen falls die Verbindung zwischen den Clients abbricht oder ähnliches.

Da dieses VPN auch mit IP-Adressen arbeitet, kann es zu Kollisionen der Adressbereiche unter den verschiedenen Netzen kommen. Grundlegend werden bestimmte Adressbereiche bestimmten Unternehmen und Anwendungsbereichen zugeteilt. So hat Beispielsweise das LAN im Normalfall eine Netzadresse ähnlich zu „192.168.x.x“. Dieses Problem trat auch bei Hamachi auf als deren ursprünglicher Adressbereich offiziell an jemand anders vergeben wurde. Dies kann dazu führen, dass gewisse Server nicht mehr zu erreichen sind, wenn man sich im VPN befindet [22][23].

Die untere Abbildung ist an die des Elektronikkompendiums angelehnt und soll den Ablauf des Hole-Punching verdeutlichen.

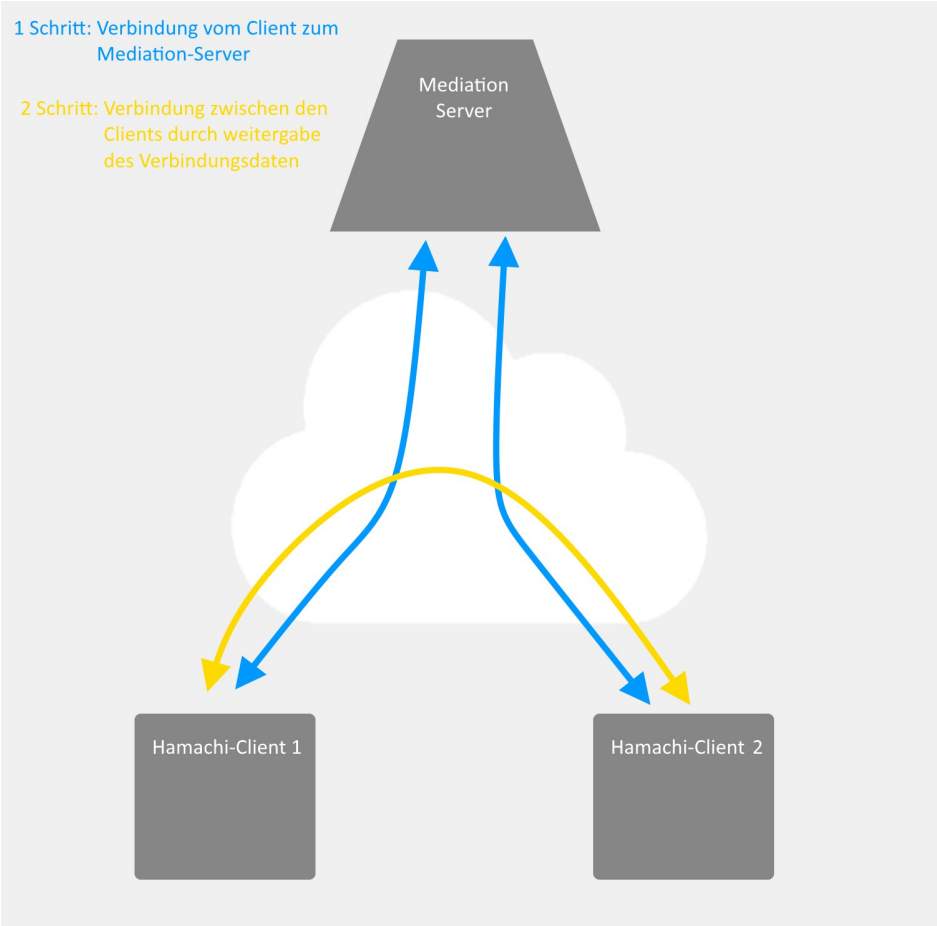


Abbildung 2: Hole-Punching Veranschaulichung

1.6.2 Tunngle

Tunngles [24] Hauptfunktion ist genau wie Hamachi das Erzeugen von Ende-zu-Ende-Verbindungen. Die genutzten Sicherheitsstandards sind nach Herstellerangaben die gleichen. Über den konkreten Ablauf der Kommunikation ist allerdings nichts genaues bekannt [25]. Die Kosten der Software sind vergleichbar mit denen von Hamachi. Es gibt eine kostenlose Version, die Werbung enthält und Netzwerke mit bis zu 32 Teilnehmern zulässt. Gegen gebühren kann man auch größere Netze erstellen. Die Software ist bisher scheinbar nur für Windows erhältlich. Herstellerangaben für andere Betriebssysteme gibt es nicht [26].

Tunngle hat eine eigene „Mini-Firewall“ integriert und bietet auch die Möglichkeit über eine API eigene Erweiterungen für die Anwendung zu schreiben.

1.6.3 OpenVPN

OpenVPN [27] ist eine Software, die für diverse Betriebssysteme zur Verfügung steht und es ermöglicht eigene VPNs zu erstellen. Anders als viele andere Produkte handelt es sich um Open-Source-Software und man kann sowohl Client als auch Server hosten.

Die Nutzung ist für private Zwecke kostenlos, für kommerzielle jedoch kostenpflichtig.

Die Verbindung zwischen den Teilnehmern kann mit UDP oder auch TCP aufgebaut werden.

OpenVPN nutzt für Verschlüsselung und Authentifikation TLS und SSL, mehr dazu später. Im Idealfall bekommt jeder Client ein eigenes Zertifikat, um jeden eindeutig identifizieren zu können. Je nach Konfiguration arbeitet OpenVPN auf dem zweiten oder dritten OSI-Layer. Insgesamt lässt sich OpenVPN sehr flexibel konfigurieren und bietet diverse Möglichkeiten. Eine GUI ist nicht zwangsweise nötig bzw. enthalten. Die offizielle Grafische Benutzeroberfläche (Englisch: GUI) bietet auch nur grundlegende Funktionen. Die nur das Betreiben eines Clients ermöglicht.

Daher lässt sich OpenVPN nur schwer mit Hamachi oder Tunngle vergleichen und bietet ist insgesamt weniger Benutzerfreundlichkeit. OpenVPN lässt sich eher mit dem IPsec Protokoll vergleichen, wenngleich OpenVPN eine Software ist und zugleich ein eigenes Sicherheitsprotokoll enthält [28]. Beim Vergleich der beiden stellt sich heraus, das OpenVPN schneller Arbeitet als IPsec [29] [30] [31].

OpenVPN unterstützt die gleichen Sicherheitsstandards wie Hamachi, bietet aber auch viele andere Konfigurationen.

1.6.4 Alternativen

Die zugrundeliegende Tabelle ist selbsterstellt. Und soll grob grundlegende Eigenschaften verschiedener VPN-Software gegenüberzustellen. Die Auswahl ist selbstverständlich nicht vollständig. Es gibt weitere Alternativen (z.B. GNUnet, tinc, etc.). Die gelisteten sind für diese Arbeit aufgrund ihrer Eigenschaften ausgewählt worden.

Name	Benutzerfreundlich	Login frei	Open-Source	Plattformunabhängig	Alle Arten unterstützt	Link
DynVPN	x		x	x		https://www.dynvpn.com/
SoftEtherVPN		x	x	x	x	http://www.softether.org
ZeroTierOne	x		x	x		https://www.zerotier.com
Freelan		x	x	x		http://www.freelan.org
Libreswan		x	x			https://libreswan.org
OpenVPN		x	x	x	x	https://openvpn.net

Abbildung 3: VPN-Adapter alternativen Übersicht

Dabei sind die Einschätzungen zur Benutzerfreundlichkeit sehr subjektiv und etwas ungenau. SoftEtherVPN ist beispielsweise nur durch den Funktionsumfang recht unübersichtlich und viele Produkte sind nur ohne GUI erhältlich, was durchaus benutzerunfreundlicher ist. Weiterhin lässt sich bei OpenVPN ein optionaler Login in das VPN konfigurieren.

Auch die Ausprägung des Open-Source-Merkmals sind verschieden. So kann beispielsweise der Client Open-Source sein, während der Code des Servers nicht offen liegt.

Zudem ist die Frage, ob alle VPN-Arten nativ unterstützt werden nicht unbedingt ausschlaggebend. Es kann genügen wenn End-to-End funktioniert. Hierdurch kann durch Weiterleitungen oder andere Einstellungen der Effekt eines Site-to-Site bzw. End-to-Site erzielt werden.

1.7 Docker

Docker [32] ist eine Software, die es ermöglicht Linux-Systeme virtualisiert laufen zu lassen. So laufen bestimmte Anwendungen getrennt voneinander auf eigenen Systemen. Dadurch ist es sehr einfach möglich vorkonfigurierte Systeme zu verteilen und auch privat ohne großen Aufwand funktionstüchtig aufzusetzen.

Dies kann sinnvoll sein, wenn man vorkonfigurierte Server an Nutzer mit wenig Erfahrung geben möchte, damit sie die Funktionalität auch ohne Hintergrundwissen nutzen können.

Auf die genauen Vorteile von Docker gegenüber anderen Virtualisierungen wird in dieser Arbeit nicht weiter eingegangen.

1.8 GitHub

GitHub [33] ist eine Online Plattform, deren Hauptfunktion die Versionsverwaltung von Programmcode ist. GitHub bietet verschiedene Zusatzfeatures, die die Organisation und Entwicklung von Anwendungen erleichtern. Darunter fällt der Issuetracker, der es einem erlaubt bestehende Fehler und offene Features in einer Art Forum zu dokumentieren, bearbeiten und übersichtlich zu verwalten. Auch bietet GitHub eine Wiki-Funktion, in der man Handbuch und Programmdokumentation schnell und übersichtlich unterbringen kann. Weiterhin eignet sich die Plattform sehr gut, um in größeren Teams zu arbeiten, da Code-Review und Branching unterstützt werden [34].

Es können kostenlos sogenannte Repositories erstellt werden sofern diese öffentlich sind. Private Repositories ,bzw. Repositories, die nicht jedem Zugänglich seien sollen, können gegen Bezahlung genutzt werden. Bei beiden Varianten werden alle Daten von GitHub gehostet, das heißt man selbst muss sich nicht um die Infrastruktur etc. kümmern, anders als bei alternativen wie zum Beispiel GitLab [35].

Software als Open-Source zu verbreiten hat diverse Vorteile. Zum einen wird der Verbreitungsgrad erhöht, zum anderen ist die Qualität des Programmcodes nachweisbar. Weiterhin wird durch das Wissen, das andere den Code sehen werden, voraussichtlich die Motivation für Qualität und Dokumentation gesteigert. Nachteile sind unter anderem der meist eher geringe Support, meist keine Garantie auf Korrektheit, fragwürdige Langlebigkeit und oft auch eher wenig Benutzerfreundlichkeit bzw. schlechte Schulungsmöglichkeiten in der Software.

Ein großer weiterer Vorteil ist, dass es nicht möglich ist, sogenannte Hintertüren für Zugriff auf ausführenden Rechner zu nutzen oder Daten zu sammeln, die man einer proprietären Software nicht ansehen würde [36].

1.9 Sicherheitsaspekte

1.9.1 SSL / TLS

TLS (Transport Layer Security) ist der Nachfolger von SSL (Secure Socket Layer). Die verschiedenen Versionen von SSL und TLS sind im Allgemeinem nicht

abwärtskompatibel. Bei diesen Verfahren werden Zertifikate verwendet, um den Kommunikationspartner zu Authentifizieren.

TLS wird in verschiedenen Anwendungen eingesetzt. Oft wird es im Browser für HTTPS-Verbindungen, also simple verschlüsselte Website anfragen, angewandt. In dieser Arbeit wird es thematisiert, da OpenVPN bzw. OpenSSL Verschlüsselung und Authentifikation einsetzen die auf TLS basiert.

Zunächst baut der Client die Verbindung zum Server auf. Dabei wird der Server über alle unterstützten Verschlüsselungsprotokolle informiert, dies sind ausschließlich symmetrische Verfahren, da diese wesentlich performanter sind. Der Server authentisiert sich mittels Zertifikat bei dem Client. Daraufhin hat der Client die Möglichkeit sich zu authentisieren. Bei HTTPS wird hierauf beispielsweise Verzichtet da es hierbei im Allgemeinen unnötig ist, bei OpenVPN ist dies aber durchaus sinnvoll. Nach erfolgreicher Authentifikation wird entweder durch den Client ein Schlüssel festgelegt oder es wird sich mit dem Diffie-Hellman-Verfahren durch beide Parteien sich auf einen Schlüssel verständigt.

TLS ermöglicht Komprimierung, Verschlüsselung, Vertraulichkeit und Integrität der Daten. TLS arbeitet auf den Schichten 4 und 5 des OSI-Modells [37] [38] [39] [40].

1.9.2 RSA

Grundlage des RSA Verfahrens ist das mathematische Problem, dass sich Primzahlen nicht effizient berechnen lassen. Es ist ein asymmetrisches Verschlüsselungsverfahren und benötigt daher einen öffentlichen Schlüssel E zum Verschlüsseln und einen privaten Schlüssel D zum Entschlüsseln. Die Anwendung kann auch ohne Grundlage der Zahlentheorie erfolgen. Man wählt zwei Primzahlen p und q und berechnet $n = p \circ q$. Nun wählt man E, so dass gilt: $x = (p-1) \circ (q-1) \wedge E \nmid x$

Danach berechnet man D, indem man das multiplikativ Inverse zu E in der Restklasse x bestimmt. Also so dass gilt: $D \circ E = 1 \, mod \, (x)$

Die Nachricht N wird wie folgt zu N' Verschlüsselt : $N' = N^E$

Entschlüsselung geschieht durch: $N = N'^D$

Es gibt verschiedene Einschränkungen an die Wahl der Parameter damit das Verfahren als sicher gilt. Diese werden hier aber nicht weiter beleuchtet [41] [42].

1.9.3 Zwei Faktor Authentifizierung

Die Zwei-Faktor-Authentifizierung beschreibt ein Verfahren, in dem ein Nutzer sich mittels zwei Komponenten authentisieren muss. Es ist mittlerweile Standard bei allen Anwendungen, die im Internet genutzt werden. Dabei sind die Komponenten im Normalfall der Benutzername und das Passwort. Der Name ist oft nicht geheim, das zugehörige Passwort sollte es sein. Somit wird nachgewiesen, dass nur berechtigte Personen Zugriff auf gewisse Daten erhalten. Der Nutzername dient meistens auch der Identifikation, um bestimmte Features zu personalisieren.

Der Vorgang der Authentisierung kann auch automatisiert werden, sodass keine bzw. kaum manuelle Eingaben nötig sind [43] [44] [45].

2. Umsetzung der Software

2.1 Definition des Funktionsumfangs

Es soll die Anwendung „EasyPeasyVPN“ entwickelt werden, die es schnell und einfach ermöglicht VPNs einzurichten. Prinzipiell soll die Anwendung sehr flexibel sein und verschiedene Einstellungen ermöglichen. Es soll eine Standardkonfiguration vorgegeben sein, die ähnlich zu Hamachi funktioniert, sodass keine technischen Kenntnisse zur Erstellung benötigt werden, aber die Konfigurationsmöglichkeiten hierdurch nicht eingeschränkt werden. So soll es mittels eines Mediation-Servers möglich sein allein mit dem Netzwerknamen und gewissen Anmeldedaten sich dem Netzwerk anzuschließen. Für diese Arbeit wird ein solcher Server privat betrieben. Der Code dafür soll auch öffentlich zugänglich sein, da der private Server vermutlich nicht sehr lange laufen wird. Aufgrund der geringen Zeit wird der Funktionsrahmen etwas weiter Beschränkt. Es wird voraussichtlich zunächst nur auf End-to-End VPNs gesetzt. Für den Mediation-Server kann es Einschränkungen auf bestimmte Protokolle geben, da die Vermittlung über Firewalls Protokoll abhängig sein kann. Es soll aber schon darauf geachtet werden die Anwendung in Zukunft erweitern zu können.

Eine grafische Benutzerschnittstelle soll sowohl zur Einrichtung des Netzes als auch zur Übersicht des aktuellen VPN-Status dienen.

Es soll eine einzige Anwendung geben, die sowohl den Client als auch den VPN-Server erstellt, da dies für End-to-End notwendig ist. Der Mediation-Server soll in einem eigenem Prozess laufen, da es nur einen Mediation-Server für beliebig viele Clients bedarf.

2.1.1 Client

Beim Start der Anwendung soll die Standardansicht zu sehen sein. Bei bestehender Verbindung, soll die eigene VPN-IP angezeigt werden sowie eine Liste aller Teilnehmer im aktuellen Netz. Man hat die Funktionen ein Netzwerk zu erstellen und einem bestehenden beizutreten. Das Beitreten soll dabei direkt über die IP oder über einen Netzwerknamen und die Adresse eines Mediation-Servers funktionieren. Zunächst werde die Verbindungsdaten des privat gehostete Mediation-Server in die

Anwendung integriert, ein Austausch des Servers wird vermutlich ohne neu zu kompilieren nicht möglich sein.

Man soll sich immer nur in maximal einem VPN-Netzwerk gleichzeitig befinden können, um Sicherheitsrisiken vorzubeugen. Da, wenn man zeitgleich in verschiedenen Netzen aktiv ist, der Rechner ferngesteuert werden könnte und so leicht Zugriff auf diverse Netze möglich wäre. Wenn man offline ist bzw. kein Netzwerk ausgewählt hat werden sinnfreie Funktionen abgeschaltet.

Die Verbindung zum Server soll mit möglichst wenig Daten erfolgen, sodass nur die Adresse und Zugangsdaten benötigt werden. Die Anwendung soll sich um die Installation und Konfiguration der Komponenten kümmern die nötig sind.

2.1.2 Server

Der Vorgang ein Netzwerk zu erstellen soll sehr einfach gehalten werden. Es soll Schritt für Schritt verschiedene Einstellungsmöglichkeiten geben.

Die erste Wahl soll sein, ob man eine vorgegebene Konfiguration verwenden will oder durch Schritt-für-Schritt-Anleitungen eine eigene Konfiguration erschaffen will.

Zunächst wird es nur die Vorgegebene Konfiguration geben. Bei Zeit dann die andere Möglichkeit. In der folgende Einrichtungen möglich wären:

1) Die Wahl des VPN Adapters, zunächst Beschränkt auf OpenVPN.
2) Die Wahl des Layer-4 Protokolls: UDP oder TCP.
3) Die Möglichkeit eine Authentifikation einzurichten, also Liste von Nutzernamen mit zugehörigem Passwort zu erstellen.
4) Den VPN-IP-Adressbereich anzugeben.
5) Die Verschlüsselung einzurichten und auch die Schlüsselgröße festzulegen.
6) Die Wahl, ob die Clients weitestgehend Anonym bleiben, sodass beispielsweise keine Broadcast möglich sind.

Es besteht auch die Möglichkeit in Zukunft einen kleinen Netzwerk-Chat hinzuzufügen. Da die Verschlüsselung schon durch das VPN sichergestellt wird, wäre ein einfacher End-to-End-Chat ohne weitere Probleme möglich.

Für die Standardkonfiguration sollte eine durchschnittliche Verschlüsselung, ohne funktionale Einschränkung und gute Performance im Vordergrund stehen. Daher wird

die Standardkonfiguration auf UDP, ohne weitere Authentifikation des Nutzer, mit einer eher durchschnittlichen Verschlüsselung und Broadcast-Unterstützung gesetzt.

2.2 Technische Grundlagen der Anwendung

2.2.1 Zielsysteme

Netzwerke sind betriebssystemunabhängig. Somit gilt dasselbe auch für die Infrastruktur für VPNs. Daher empfiehlt es sich auch die Anwendung Betriebssystem übergreifend Verfügbar zu machen. Zunächst sollten aber die beiden größten Vertreter unterstützt werden. Hierbei wird sich auf Windows und einige Linux-Systeme beschränkt. Da es aber sehr viele Linux-Distributionen gibt, wird zunächst nur eine konkrete Testmaschine getestet. Es ist aber absehbar, das die Unterschiede marginal sein werden und so auch eine Vielzahl Systeme abgedeckt sein werden.

Der Schwerpunkt der Anwendung liegt zunächst auf dem Desktopbetrieb. Mobile Geräte wie etwa Smartphones werden zunächst vernachlässigt, sollten aber dennoch bei der Wahl der Komponenten berücksichtigt werden und in Zukunft ggf. auch verwendbar sein.

2.2.2 Programmiersprache

Da die Anwendung möglichst viele Plattformen unterstützen sollte, macht es Sinn eine Sprache zu wählen die dies von Haus aus unterstützt. Und so möglichst wenig Aufwand beim Portieren der gesamten Anwendung auf andere Geräte zu haben.

Zum einen würden sich Websprachen gut eignen da diese auf fast allen Betriebssystemen bereits unterstützt werden. Allerdings sind diese auch leicht zu manipulieren, da diese nur zur Laufzeit interpretiert werden und fast jeder Browser die Möglichkeit hierfür bietet. Zudem weiß ich aus Erfahrung, dass diese meist nur sehr wenig betriebssystemnahe Funktionen bieten, wie etwa das Auslesen aller verwendeten Netzwerkadressen etc. Theoretisch wäre es aber möglich ein VPN über den Browser laufen zu lassen [46].

Eine Alternative wäre Python. Es bietet die Möglichkeit auch auf gewisse Betriebssystemfunktionen zu zugreifen. Zudem ist die Syntax recht intuitiv und es bietet verschiedene Arten der Programmierung. Sowohl funktionale als auch objektorientierte Paradigmen werden unterstützt. Einer der Vorteile ist, dass Python in recht wenig Zeilen Code viel erreichen kann. Daher ist es für Einsteiger und kleine

Projekte mit wenig Zeit sehr gut geeignet [47] [48]. Was den Anforderungen dieser Arbeit eher weniger genügt.

Java ist eine weit verbreitete Sprache und bietet einen ähnlichen Funktionsumfang wie Python. Abgesehen von der funktionalen Programmierung, die nicht vorhanden ist. Zudem ist Java ähnlich Performant wie Python [49] und ist durch strikte Typisierung und redundante Informationen, beispielsweise bei der Typisierung, leichter für Entwickler um in ein neues Projekt einzusteigen. Zudem gefällt mir aus meinen Erfahrungen Java besser als Python. Außerdem wird der Code von Java compiliert und erst danach interpretiert. Daher lässt er sich nach fertigstellung einfacher an Leute verteilen, die selber keine Erfahrung mit Anwendungsentwicklung haben.

Zusammenfassend bietet Java ggf. schneller die Möglichkeit weitere Entwickler an das Projekt heranzuführen und da das Projekt zunächst von mir umgesetzt wird liegt es nahe die passende Sprache zu wählen.

2.2.3 Argumentation und Wahl des Adapters

Es gibt grundsätzlich mehrere VPN-Netzwerk-Adapter, die die Anforderungen, sicher, Open-Source und privates Hosten, erfüllen. Im ersten Teil der Arbeit wurden ein paar oberflächlich vorgestellt und auf Grundlagen dazu eingegangen. So kristallisiert sich das die Wahl nicht eindeutig sein kann und auch wie bereits erwähnt in Zukunft ggf. weiter unterstützt werden sollen. Da die Verbindung zeitgemäß sicher sein soll, scheidet PPTP aus. Somit bleibt noch die Wahl zwischen dem weit verbreiteten Standard IPsec oder dem SSL-basierten OpenVPN.

IPsec hat eine schlechtere Performance als OpenVPN und bietet vergleichbare Sicherheit. Daher ist OpenVPN geeigneter.

SoftEther ist ein relativ neues und sehr umfangreiches Projekt aus dem Jahr 2013. Es unterstützt sowohl IPsec als auch OpenVPN und einige andere Protokolle, damit wäre es perfekt geeignet [50]. Allerdings kommt es durch eine Vielzahl an Features auch zu einer höheren Fehleranfälligkeit. Zudem ist Open-Source-Software meist öfter geprüft, je älter diese ist. Dies ist natürlich auch abhängig davon wie hoch die Reputation ist. Vergleicht man die Angaben auf GitHub so genießt SoftEther eine größere Aufmerksamkeit, hat allerdings auch mehr als 200 offene Fragen und Probleme [51] [52].

Da es zu OpenVPN mehr Hintergrundtexte und Beispielkonfigurationen gibt, wird sich sowohl wegen Einarbeitungszeit als auch wissenschaftlicher Eigenleistung für OpenVPN entschieden. Es ermöglicht das Auseinandersetzen mit Kommunikation über NATs und Firewalls hinweg, was SoftEther in gewissem Umfang schon bereit stellt. Zudem sind die Konfigurationsmöglichkeiten von OpenVPN schon gewaltig und bieten gute Performance wie auch Sicherheit.

2.2.4 Entwicklungsumgebung

Es gibt verschiedene Entwicklungsumgebungen für Java. Darunter auch einige qualifizierte für große Projekte. Beispielsweise Netbeans, Eclipse und IntelliJ IDEA. Letztere wurde auch von dem Großkonzern Google für die Entwicklung des Android Studio genutzt [53]. Auch aus eigener Erfahrung bietet dies diverse Möglichkeiten die Entwicklung zu beschleunigen. Daher fällt die Wahl auf dieses Tool, im Folgenden als IntelliJ oder IDE bezeichnet. Als Versionsverwaltungsserver wird aus gegebenen Gründen GitHub genutzt. Das Repository lässt sich aus IntelliJ heraus verwalten. Auf ein Branching-Modell wird zurzeit verzichtet, da dies nur richtig Effizient ist wenn in einer größeren Gruppe zusammen an den gleichen Dateien gearbeitet wird [54].

Da es bei der Entwicklung um Netzwerke geht, ist es notwendig diese Anwendung auch auf verschiedenen Geräten gleichzeitig laufen zu lassen. Aus dem Grund wird auf verschiedenen Geräten getestet. Die gesamte Entwicklung findet auf einem Notebook (Acer Aspire R-7) mit Windows 10 statt. Als zweite Partei, Server bzw. Client, wird ein RaspBerry Pi 1 Model A mit dem Betriebssystem Raspbian, basierend auf Debian [55], genutzt. Als Mediation-Server soll ein Cubietruck [56] mit Cubian, ebenso eine Debian basiert [57], genutzt werden. Zudem wird ggf. ein kostenloser Webhoster genutzt, um Probleme mit dynamischen IP-Adressen zu umgehen. Also IP Adressen, die sich täglich ändern und somit keine direkte Adressierung ermöglichen.

Auf die Einrichtung der verschiedenen Geräte wird nicht eingegangen. Alle Geräte sind über das selbe W-Lan mit einander Verbunden. Dennoch können die Komponenten in dem VPN über das Internet Kommunizieren. Dateien bzw. aktuellen Versionen der Software werden mittels SFTP (SSH File Transfer Protocol) durch die Anwendung FileZilla übertragen.

Für die Dokumentation der einzelnen Klassen und Methoden soll das von Java bereitgestellte JavaDoc dienen. Zudem soll auf GitHub auch eine kleine Anleitung

vergleichbar mit einem Handbuch zur Verfügung gestellt werden. Andere Funktionen von GitHub werden zunächst nicht genutzt, da diese meist nur der Kommunikation von mehreren Projektteilnehmern dienen.

2.3 Implementierung und Beispielcode

2.3.1 Designentscheidungen

Bei der Implementierung wurde zunächst mit der GUI für die Installation bzw. Konfiguration angefangen. Da hier mehrere Schritte durchlaufen werden sollen, bietet es sich an eine einheitliche Oberfläche zu haben und Gemeinsamkeiten in eine eigene Klasse auszulagern. So entstanden der InstallationController ,InstallationFrame und das InstallationPanel [58]. Der Controller kümmert sich um die Verwaltung, welche Ansicht geladen werden soll. Als eine Art Wrapper zeigt der Frame die verschiedenen InstallationPanels an, die auf die Eingaben des Frames wie „Weiter“ oder „Zurück“ auch reagieren können. Die Klassen zusammen mit dem ConfigState [59] bilden grob ein Entwurfsmuster das dem Model-View-Presenter ein wenig nahe kommt. Diese Abwandlung des Model-View-Controllers [60] trennt ebenso Daten / Model , Ansicht / View und Logik bzw. Verwaltung / Controller bzw. Presenter. Unterschied hierbei ist aber die eindeutigere Trennung der Komponenten [61]. Die verschiedenen Implementierungen der InstallationPanels widersprechen auf den ersten Blick diesem Grundsatz. Bei genauerer Betrachtung sind die Panels jedoch eher ein weiterer Subcontroller des InstallationControllers. Diese Initialisieren die GUI im Konstruktor und binden via ActionListener meist an diesen Controller. Oft wird hier auch der RadioButtonWithDescription verwendet, der diese Trennung offensichtlicher repräsentiert. Um es eindeutiger zu halten und Verstöße gegen das Design zu vermeiden, wäre eine weitere Modularisierung, also eine Trennung auf Dateiebene, weitaus besser. So würde jedes Panel eine weitere Controller / Presenter Klasse bekommen.

Darauf wurde hier aus verschiedenen Gründen verzichtet. Zunächst hat kaum eine der Klassen eine so komplexe GUI und dementsprechend auch nur geringfügige Anzahl Zeilen Code. Sodass eine Modularisierung nur unnötig die Klassenstruktur aufblähen würde. Zum anderen wird durch Modularisierung für alles weitere Schnittstellen benötigt, die den Code wiederum aufblähen würden. Des Weiteren wurde in der Anwendung zunächst auf automatisierte Tests verzichtet. Was nicht heißt das keine Fehler abgefangen werden, im Programmcode sind

Fehlerrückmeldungen verbaut, die es in einem so frühem Stadium der Software ermöglichen die Ursache zurück zu verfolgen. Falls Tests nötige wären , wären sie aber durch Reflection möglich. Die Komplexität der Software liegt weniger in der GUI und den gespeicherten Daten, sondern vielmehr in der Einrichtung eines funktionierenden Netzwerks. Hierbei lassen sich nur schwer automatisierte Tests durchführen, denn man bräuchte ein automatisiertes Deployment, also ein automatisiertes Verteilen, Einrichten, Ausführen und Auswerten der Anwendung, für mehrere verschiedene Testmaschinen.

Das zuvor angesprochene Verhalten der InstallationPanels auf Vor- und Zurück-Navigation zu reagieren oder diese zu unterbinden kann als Delegation verstanden werden. Dabei gibt der InstallationController diese Aufgabe an den PanelController weiter [62] [63].

Hardware-Ressourcen stehen nur begrenzt zur Verfügung. Daher ist es oft notwendig den Zugriff durch einen allgemeinen Verwalter auszuüben. Dieser sollte auch alle direkten Zugriffe auf die Komponenten verhindern. Dafür eignet sich die Singleton-Architektur. Sie wird hier eingesetzt, um unter anderem den Zugriff auf Netzwerkkommunikation, zentrale Ansprechpartner, wie den InstallationController, oder die Festplatte und Daten kontrolliert zu nutzen.

Bis auf das Singelton-Designpattern sind in dieser Arbeit keine reinen Softwarearchitekturen verbaut. Das liegt zum Teil daran das es gelegentlich keinen Sinn macht die Theorie in konkrete Anwendungsfällen eins zu eins Praktisch umzusetzen. Unter anderem spielt hierbei der Faktor Zeit und Ergebnis eine große Rolle, der zumindest in der Wirtschaft eine Kernkompetenz ist. Qualität sollte aber natürlich dennoch gewährleistet sein.

2.3.2 Einrichtung OpenVPN

Unabhängig davon, ob man die Anwendung als Server oder Client einrichtet wird aktuell der OpenVPN-Adapter gewählt und eingerichtet. Hierbei wird durch die Anwendung OpenVPN installiert und benötigte Konfigurationsdateien erstellt und abgespeichert. Da für die Installation des Adapters Administrationsrechte benötigt werden, muss die JVM auch mit Administrationsrechten gestartet werden. Falls der Adapter schon installiert ist, wird dies auch festgestellt und auf eine Installation verzichtet. Dafür werden auch keine Administrationsrechte mehr benötigt. So kann man die Installation auch manuell durchführen, wenn einem dies lieber ist bzw.

OpenVPN bereits installiert war. Die Installation unterscheidet sich bei Linux und bei Windows, denn für Windows gibt es einen Installer, während man bei Linux auf den Paketmanager zurückgreifen sollte. Man kann den Code aber auch für sein Betriebssystem selber kompilieren. Alle drei Arten werden zum Teil unterstützt. Allerdings gibt es bei allen Varianten etwas zu beachten. Unter Windows kann mit dem Installer nicht gewährleistet werden, dass alle benötigten Komponenten mitinstalliert werden. Folgende Dinge sollten dabei beachtet werden, um einen VPN-Server zu erstellen muss easy-rsa mitinstalliert werden. Zudem benötigt man die GUI von OpenVPN, die vorselektiert ist, nicht. Hierzu ist eine Veranschaulichung im Anhang auf Seite 56. Der Installer liegt zudem zur Zeit als Resource im JAR-Archiv, dies sollte geändert werden, sodass immer die aktuellste Version über das Internet geladen wird. Unter Linux werden aktuell nur Debian und Ubuntu Distributionen unterstützt, da bisher nur die Installation durch das Advanced Packaging Tool (APT) [64] implementiert ist. Diese lädt die aktuellste Version. Die Installation über eigenes Kompilieren ist auch verbaut. Allerdings funktioniert diese nur, falls OpenSSL vorher installiert wurde. Da es weiter Implementierung bedarf, wurde dieses Feature zunächst deaktiviert. Zudem wird hier auch noch nicht die aktuellste Version geladen.

Unter Linux werden die Installationen prinzipiell nach und nach über ein Terminal im Hintergrund ausgeführt.

Die Konfiguration des VPNs verläuft für Client und Server unterschiedlich. Der Server benötigt einen größeren Aufwand, als Vergleich ist hierfür im Anhang auf Seite 54 ein Klassendiagramm. Als erstes werden Konfigurationsdateien erstellt. Die Parameter sind hierbei von OpenVPN vorgegeben, hierbei wird auch schon die Konfiguration aus den vorherigen Installationsschritten berücksichtigt. Allerdings gibt es aktuell nur die Expressinstallation. Das Erstellen dieser Datei kann bestehende manuell erzeugte OpenVPN-Konfigurationen überschreiben. Danach müssen die Zertifikate via easy-rsa erstellt werden. Während der Implementierung war es hier schwierig die Befehle, die zur Erstellung in der Kommandozeile eingegeben werden mussten, im richtigen Moment mit den richtigen Parametern einzugeben, da dies sehr schwierig zu debuggen ist. Es wird für alle Clients aktuell das gleiche Zertifikat verwendet. Dies birgt Risiken und sollte in Zukunft geändert werden. Die Konfiguration des Servers ist damit geschehen. Nun kann der OpenVPN-Dienst mit der Konfiguration gestartet werden.

Die Konfiguration des Clients benötigt nur die Erstellung der Konfigurationsdatei. Einige Parameter sind dabei identisch. Allerdings werden auch andere benötigt. Beispielsweise müssen auch Verbindungsdaten zum Server angegeben werden.

Damit der Client gestartet werden kann benötigt dieser auch eines der vom Server erstellten Zertifikate. Zusätzlich benötigt der Client die Informationen über die Konfiguration des Servers damit beide miteinander kompatibel sind. Dafür wurde ein Meta-Server, ein Server der den Austausch von Metainformationen dient , bzw. Meta-Client geschrieben. Der sich mit dem Austausch dieser Daten befasst.

2.3.3 Client-Server-Architektur

In der Anwendung wird viel zwischen verschiedenen Partnern auf den jeweils anderen Rechnern kommuniziert. Man kann wohl annehmen, dass der größte Teil der Daten zwischen den beiden OpenVPN-Anwendungen übertragen wird. Der Aufbau der Verbindung mit den Metainformationen ist dabei geringer. Damit diese aber den anderen Partner finden bedarf es weiterem Datenaustausch. Siehe hierzu die Abbildung 4 am Ende dieses Abschnitts.

Im einfachsten Fall ist dem Client die IP-Adresse des Servers bereits bekannt und die Nutzer befinden sich hinter der gleichen NAT. Diese verbinden sich nun mit dem Meta-Server. Der Client fragt nun zuerst nach den Netzwerkkonfigurationen und danach nach den benötigten Zertifikaten. An dieser Stelle sollte zukünftig die gleiche Authentifikation ausgeführt werden wie auch OpenVPN dies tun würde. In einer Datei sollten Benutzername und Passworthash gespeichert werden. Ein Java Aufruf überprüft ob die angegebenen Daten enthalten sind und beendet je nachdem den Java-Prozess mit einem anderem Exitcode [65]. Die Zertifikate werden bereits verschlüsselt übertragen. Gegebenenfalls kann man sich die Frage stellen ob durch doppelte Verschlüsselung diese nicht wieder aufgehoben wird. Allerdings ist die Wahrscheinlichkeit verschwindend gering, da das Zertifizierungsverfahren und das Verschlüsselungsverfahren unabhängig von einander funktionieren. Nach erfolgreicher Übertragung kann sich der Client nun mit dem OpenVPN-Server verbinden.

Um den Verbindungsaufbau noch einfacher zu halten, habe ich versucht einen Mediation-Server zwischen die beiden Partner zu schalten. Dieser soll zu einem Netzwerknamen, da Namen für Menschen einfacher als IP-Adressen zu merken sind, den OpenVPN-Server ermitteln und durch UDP-Hole-Punching eine

Verbindung durch verschiedene NATs ermöglichen. Der Mediation-Server funktioniert soweit, das die Verbindungsdaten eines Netzwerks an einen Client weitergeben werden.

Eine weitere Besonderheit die mit eingebaut wurde ist eine Pseudonamensauflösung für den Mediation-Server. Da man als privat Person von seinem Provider oft nur gegen Gebühren eine statische IP Adresse bekommt, es aber genug kostenlose Webhoster mit kostenloser Subdomain gibt, kann der Mediator dort seine aktuelle IP hinterlegen und via HTTP-Request an eine fixe URL durch den Client erfragt werden. Ursprünglich wurde auch versucht das Hole-Punching allein über diesen Webserver laufen zu lassen, dies war aber nicht erfolgreich da man dafür das HTTP über UDP hätte implementieren müssen und der Server solche Verbindungen nicht blockieren dürfte [66].

Grundsätzlich funktioniert das Hole-Punching so, dass jede Verbindung, also Daten mit IP-Adressen und Ports, die durch das Gateway herausgeht einen eigenen Port zu der externen IP-Adresse bekommt, auf diesem kommen auch alle Antworten der Gegenstelle dieser Verbindung an. So kann eine Verbindung auf UDP-Basis vom VPN-Server zum Mediation-Server aufgebaut werden. Der Client fragt nach einer Verbindung zum VPN-Server. Nun gibt der Mediation-Server die Adressdaten des VPN-Server weiter und der Client kann sich über den gleichen Port verbinden und gelangt durchs NAT. Dies funktioniert mit UDP so einfach da es hier keine Flusskontrolle der Daten gibt, also der genaue Gesprächspartner auch nicht überprüft wird.

An der praktischen Umsetzung dieses Verfahrens bin ich allerdings leider Gescheitert. Grundsätzlich hat die Weiterleitung der Verbindungsdaten funktioniert, jedoch gelang es nicht OpenVPN auf den Port lauschen zu lassen. So kam es von der Client-Seite zu Verbindungs-Timeouts und der Server hat keine Daten erhalten. Die genaue Ursache konnte mit dieser Fehlermeldung nicht herausgefunden werden.

Die Tests wurden zum Teil manuell konfiguriert. Daher lässt sich der Versuch nicht ohne Weiteres mit der Anwendung alleine nachstellen.

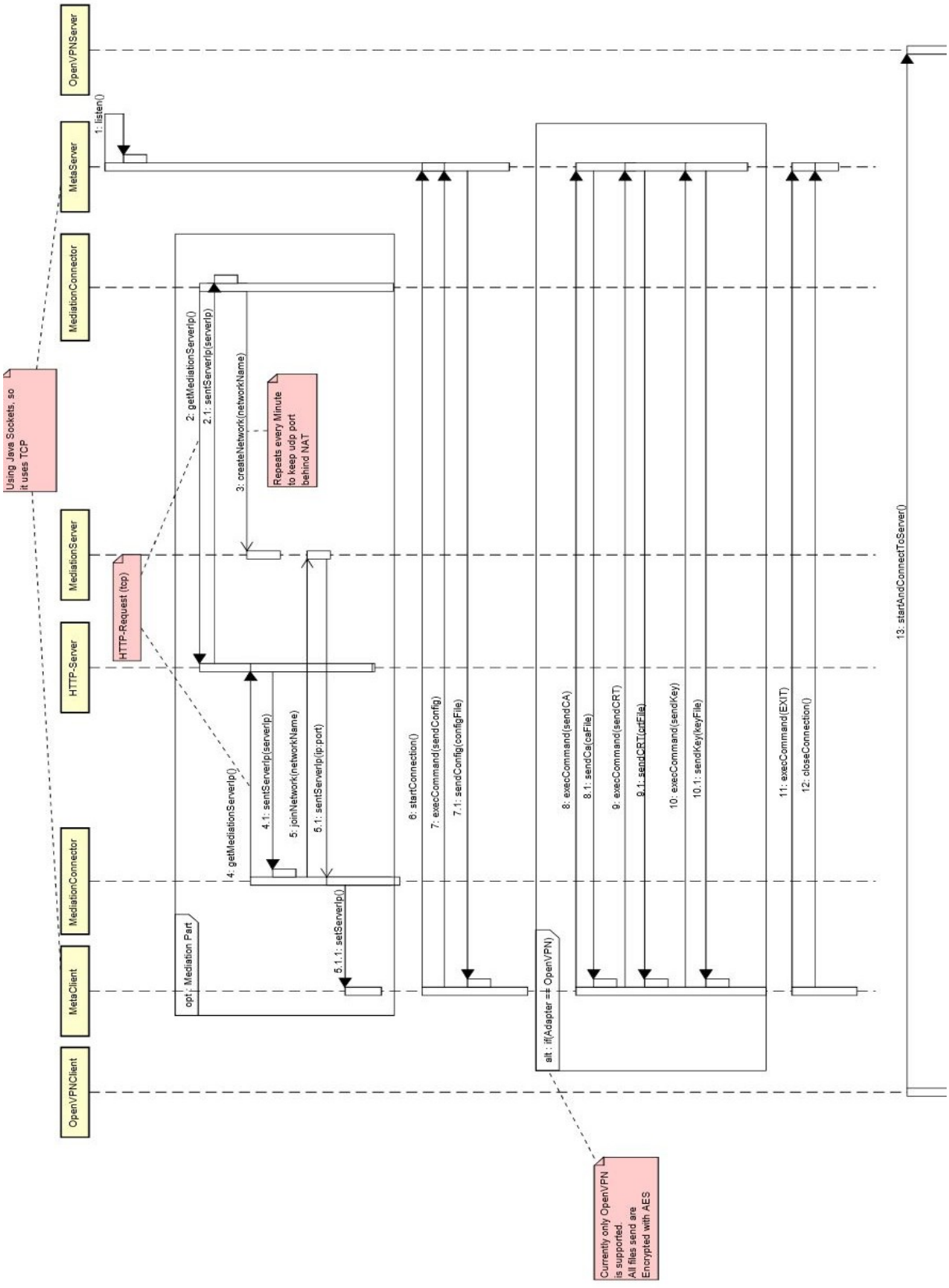


Abbildung 4: Das Sequenzdiagramm zum Verbindungsaufbau der Anwendung.

2.3.4 Betriebssystemunabhängigkeit

Um vom Betriebssystem unabhängig zu sein, reicht es nicht aus einfach nur Java zu verwenden. Es gibt diverse Problematiken, die mit einer kontrollierten Fallentscheidung für jedes Betriebssystem einzeln angepasst werden müssen. So ist beispielsweise das Dateisystem und der Pfad zum Speichern von Benutzerdaten und im Speziellen zum Speichern der OpenVPN-Daten unterschiedlich. Um dies einfach zu halten, habe ich eine Helper-Klasse genutzt, die angibt ob das aktuelle System ein Linux oder Windows System ist.

Des Weiteren habe ich Java-Annotations genutzt, um Code der nur unter einem der beiden Systeme funktioniert, zu kennzeichnen. So kann sehr schnell festgestellt werden, ob ein Fehler dadurch verursacht wurde, dass bestimmter Code nicht auf diesem System ausgeführt werden kann. Sollte eine Methode im Stacktrace auftauchen, die eine andere Annotation hat, sollte geguckt werden, warum diese Methode aufgerufen wurde. Optimieren bzw. Automatisieren könnte man dies ggf. mit Reflection bzw. Assertion, sodass automatisch eine Exception geworfen wird wenn das System ein anderes ist als annotiert.

2.3.5 Persistenz

Alle Daten, die für die Anwendung dauerhaft gespeichert werden müssen, befinden sich in einer serialisierbaren Klasse. Java kann damit Objekte in einen persistenten Zustand überführen. Dabei sind Verweise auf andere Objekte nicht mehr dieselben, aber die gleichen. Die Daten werden im Nutzerverzeichnis in einem Ordner „.EasyPeasyVPN“ gespeichert. Problematisch bei diesem Verfahren ist, dass sollte sich die Datenhaltungsklasse ändern ein auslesen der Daten nicht mehr möglich ist. Dies könnte man umgehen indem man eine statische serialVersionUID zur Klasse hinzufügt [67].

Alle weiteren Dateien, die durch die Anwendung erzeugt werden, stehen in Zusammenhang mit OpenVPN und werden daher auch in dessen Installationsverzeichnis gespeichert. So kann das VPN auch ohne EasyPeasyVPN erstellt werden, falls die Anwendung an gewissen Stellen nicht ganz funktionstüchtig sein sollte.

2.4 Projektstruktur und Anwendungsbeispiele

2.4.1 Projektstruktur

Das Projekt wurde nach programmspezifischen Kriterien strukturiert, denn hier spielt es eine wichtige Rolle wie die verschiedenen Klassen auch rechnerübergreifend miteinander arbeiten. Auf oberster Ebene liegen die beiden Packages für die eigentliche Anwendung und den Mediation-Server. Da diese beiden zwar miteinander kommunizieren, aber im Normalfall nicht auf dem selben Rechner laufen. Die anderen beiden Packages enthalten unterstützende Klassen deren Teile, an verschiedenen anderen Stellen benutzt werden. Einmal die Utilitys, die nur Klassenmethoden enthalten, und die Helper für Annotations und Klassen, die Instanziiert werden können.

Weitere Packages gibt es nur im main-Package. Hier wird zunächst nur zwischen Clienten- oder Server-Code sowie Installationscode unterteilt. Der Installationscode wiederum ist auch client- und serverspezifisch geteilt worden. Eine Teilung zu View, Model und Controller wäre vorstellbar gewesen. Wurde aber drauf verzichtet da die Verteilung hierbei sehr einseitig wäre und somit kein großer nutzen für die Struktur vorliegen würde.

2.4.2 Anwendungsszenarien

Es folgt eine Liste von User-Stories. Die folgenden User-Stories sind im aktuellem Stand (08.09.2017) in der Anwendung umgesetzt worden.

- Als Nutzer möchte ich einen VPN-Server erstellen, um es meinen Freunden zu erlauben alte LAN-Anwendungen über das Internet zu nutzen.
- Als Nutzer möchte ich einem bestehenden VPN-Netzwerk mit der mir bekannten Server-IP beitreten, um Bekannten den Zugriff auf meine Daten zu gestatten.
- Als Nutzer möchte ich sehen wer und wie viele Teilnehmer sich im aktuellen VPN befinden, um zu prüfen, ob unbefugte Benutzer ebenso Verbunden sind.
- Als Nutzer möchte ich das Protokoll bzw. den Verlauf meiner VPN-Verbindung einsehen, um Probleme zu überprüfen.

- Als Nutzer möchte ich eine Übersicht aller von mir verwendeten IP-Adressen für alle Netzwerk-Adapter sehen, um zu überprüfen, ob es Kollisionen und dadurch Probleme gibt.

Die nächsten User Stories sind Aufgrund von Komplikationen nicht voll funktionstüchtig Umgesetzt, hätten für ein Adhoc VPN aber durch aus Vorteile gehabt.

- Als Hoster möchte ich einen Mediation-Server hosten, um anderen Nutzern zu ermöglichen ohne bekannte IP-Adresse einem Netzwerk beizutreten.
- Als Nutzer möchte ich einem bestehenden VPN-Netzwerk mit dem mir bekannten Netzwerknamen und dem mir bekannten Vermittlungsserver über ein NAT hinweg beitreten, um schnell eine Verbindung aufzubauen.
- Als Nutzer möchte ich die Netzwerkdaten meines erstellten Servers auf dem Mediation-Server speichern und anderen Netzwerkteilnehmern zur Verfügung stellen um das Netzwerk dezentral zu halten.

2.5 Allgemeine Einrichtung der Anwendung

Die Benutzung der Anwendung erfordert nur eine Installierte Java-Runtime-Environment (JRE) [68]. Empfohlen und getestet wird hierfür mindestens Version 1.7, aber vermutlich ist die Anwendung auch abwärtskompatibel.

Die benötigte Installation und Konfiguration für ein VPN wird aus der Anwendung heraus erledigt. So muss nur entweder ein Netzwerk bzw. Server erstellt werden oder einem bestehenden beigetreten werden. Darauf hin wird der benötigte Adapter installiert und eingerichtet. Hier kann es ggf. auch notwendig sein die Anwendung mit Administrationsrechten zu starten, um Zugriff und Schreibrechte auf das Installationsverzeichnis von OpenVPN zu erhalten. Fehlerbehandlungen sind implementiert, jedoch ist es wie bei jeder Software in Entwicklung fragwürdig ob wirklich alle Fälle betrachtet werden konnten.

2.6 Exemplarische Beispielumgebung

2.6.1 Einrichtung des Servers

Der Server wird eingerichtet, indem man nach Start der Anwendung über die Menüleiste „Network“ → „Create“ auswählt, im darauffolgenden Installations-Fenster „Express installation“ wählt, nun via „Next“ zum nächsten Panel weitergeleitet wird

und hier einen Netzwerknamen zur Identifikation des Netzwerks eingibt. Beim nächsten klick auf „Next“ wird die Installation und Konfiguration gestartet. Hier kann es zu Problemen kommen, sollte die Anwendung nicht als Administrator gestartet worden sein. Unter Windows wird der Installationsassistent von OpenVPN gestartet. Hier gibt es Einstellungen, die es zu beachten gilt. Es muss „OpenVPN RSA Certificate Management Scripts“ ausgewählt werden, was es standardmäßig nicht ist. Die GUI muss nicht installiert werden, würde aber keine Probleme verursachen. Vergleich Anhang auf Seite 56.

Nach erfolgreicher Installation muss auf dem letzten Panel „Finish“ bestätigt werden. Danach schließt sich das Installationsfenster und im Hauptfenster wird sichtbar das der Server und alle nötigen Dienste gestartet werden. Bei erfolgreichem Start wird die Statusanzeige grün und stellt den Netzwerknamen, die eigene Rolle und auch die eigene VPN-IP dar.

2.6.1.1 Einrichtung des Routers für öffentlichen Zugang

Der Server läuft so aktuell nur im lokalen Netzwerk, was nicht viele Anwendungsfälle abdeckt. Im Normalfall wünscht man sich Kommunikation über ein größeres Medium wie etwa das Internet. Da aktuell die Vermittlung bzw. Mediation-Server nicht funktionieren, muss für einen Zugriff noch das Gateway bzw. der Router konfiguriert werden. Die meisten Geräte liefern die Möglichkeit für Portforwarding, d.h. ankommende Anfragen aus dem Internet auf dem Port X können so eingerichtet werde, dass sie an den Rechner hinter dem Router auf Port Y weitergeleitet werden. Im Anhang auf Seite 55 findet sich ein Beispielbild einer Einrichtung.

Die Einrichtung hängt von den Geräten ab, daher kann und wird hier darauf nicht weiter eingegangen.

2.6.2 Einrichtung des Clients

Recht analog zum Vorgehen der Servereinrichtung muss nach dem Start der Anwendung über „Network“ → „Join“ der Installationskontext geöffnet werden. Daraufhin wählt man „Connect via IP“ und trägt in die beiden Felder die IP-Adresse und den Port des bestehenden VPN-Servers ein. Gegebenenfalls sind dies die Angaben für das Gateway des Servers. Nach dem Austausch der benötigten Daten wird mit dem nächste Panel die Installation des VPN-Adapters ausgeführt. Ähnlich wie bei der Serverinstallation kann es hier zu Problemen kommen, falls Administrationsrechte für die Installation oder Schreibrechte benötigt werden. Nach

der erfolgreichen Installation muss im nächsten Panel wieder „Finish“ bestätigt werden. Danach schließt sich das Installations-Fenster und die Anwendung ist erfolgreich Verbunden sobald das Statussymbol grün wird.

2.7 Analyse und Tests

2.7.1 Performance / Delay

Wichtig bei der Kommunikation ist die Dauer für die Übertragung der Daten. Daher wurde auch überprüft wie gut die Daten über das VPN übertragen werden können und welche Auswirkungen Teile der Anwendung darauf haben. Hierfür wurde ein einfacher Ping, eine sehr simple Anfrage ob eine bestimmte IP-Adresse im Netzwerk erreichbar ist, als Maßstab genommen. Umfangreiche Datenübertragungen verhalten sich ggf. abweichend.

Für einen Ping gibt es vier grundlegende Kriterien: die längste sowie kürzeste Dauer, die durchschnittliche Dauer und die Angabe wie viele Pakete verloren gehen bzw. erst gar keine Antwort erhalten. Hier könnten weitaus mehr Kriterien aus der Statistik wie etwa die Standardabweichung etc. herangezogen werden. Allerdings sollte diese bei ausreichend Tests durch die vier oben genannten Kriterien keine allzu große Relevanz treffen.

Zunächst wurde getestet welche Werte über die aktuelle Infrastruktur gegeben sind. Währenddessen war die VPN-Verbindung bereits aktiv. Pings benutzen das ICMP (Internet Control Message Protocol) das auf dem OSI Layer 3 arbeitet, also einer Schicht unterhalb von UDP und TCP. Bei schlechten Ergebnissen der Verbindung bzw. bei Verlust von Paketen kann UDP also nicht die Ursachen sein. Im ersten Testdurchlauf des VPNs kamen aber recht hohe Verluste vor. Daher wurden andere Quellen wie etwa zu viel Datenverkehr innerhalb des VPNs in Betracht gezogen und somit die periodischen Pings zum Erkennen der Netzwerkteilnehmer vorübergehend deaktiviert. Dies erzielte insgesamt bessere Resultate, besonders die verlorenen Pakete wurden so reduziert und auch der Durchsatz wurde ca. um den Faktor fünf erhöht. Im LAN hingegen hat dies nicht wirklich Auswirkungen gezeigt. Daher ist der Ursprung des Problems vermutlich der Flaschenhals bei der Netzwerkschnittstelle des VPNs, da diese jeden Ping prüft. Auch solche Pings die eine andere Ziel-IP-Adresse haben. Daher sollte hier ein anderes Vorgehen gewählt werden, als alle Möglichen IP-Adressen zu anzupingen. Beispielsweise könnte der Server alle aktiven IP-Adressen ermitteln und diese in einer einzigen Liste den Clients zusenden. Dafür

ist aber wiederum eine aufgebesserte Client-Server-Architektur nötig. Möglich wäre eventuell auch ein einziger Broadcast pro Client.

Minimal	Maximal	Durchschnittlich	Verlust an Datenpaketen	VPN	Broadcast aktiv	Linux
2,2	1633,2	46,8	2,00%		x	x
2,5	1566,8	61,8	2,00%			x
5,0	970,0	80,0	0,00%			
6,0	6301,1	970,6	52,00%	x	x	x
4,2	1290,6	24,8	0,00%	x		x

Abbildung 5: Performance Tests mit 100 Pings pro Zeile. Die ersten drei Spalten sind in Millisekunden angegeben. VPN gibt dabei an, ob im VPN oder LAN getestet wurde.

2.7.2 Sicherheit

Das Programm erfüllt geringe Sicherheitsstandards. Beispielsweise wird auf AES mit 128-Bit anstatt 256-Bit gesetzt. Zudem gibt es nur ein Zertifikat für alle Clients. So lässt sich bei Diebstahl des Zertifikats kein unbefugter Zugriff verhindern. Außerdem wird die Authentizität des Servers vom Client nicht überprüft.

Des Weiteren wird für die Verschlüsselung der kritischen Dateien für OpenVPN ein statischer Schlüssel benutzt, der Open-Source und somit nicht geheim ist. Dies ist eine fatale Eigenschaft, die es jedem Angreifer erlauben könnte, in kürzester Zeit das Netzwerk zu infiltrieren, wenn er denn weiß welche Daten gerade übertragen werden. Hier sollte zumindest ein dynamischer Schlüssel auf Basis der Server-IP-Adresse oder ähnlichem gewählt werden.

Außerdem wird zurzeit die Übertragung der Netzwerkkonfiguration des VPNs bis auf die von Java gewährleistete Serialisierung nicht geschützt. Sollte ein Angreifer wissen, welche Daten übertragen werden, so könnte ihm das Informationen über den Verwendungszweck des VPNs geben.

Im Zusammenhang mit Java hört man oft, dass Java unsicher sei, jedoch ist damit eher die Ausführung über den Browser gemeint, da falls unbekannte Software ausgeführt wird, ggf. Schadsoftware ausgeführt werden kann [69].

3. Abschluss

3.1 Veröffentlichung auf GitHub

Das Programm wurde nach ungefähr der Hälfte der Entwicklungszeit auf GitHub veröffentlicht [70]. Es wurde darauf geachtet den Code bestmöglich zu kommentieren und Dateien nicht über eine Größe von ca. 300 Zeilen wachsen zu lassen, um Übersicht zu gewährleisten. So ist es einfach sich in das Projekt einzuarbeiten. Zudem wurde darauf geachtet nach außen auch gut präsentiert zu sein, indem die Readme-Datei einen Überblick über den aktuellen Stand der Features und des Entwicklungsfortschritts gibt.

Auf GitHub gibt es diverse Projekte, an denen sich viele Entwickler beteiligen, jedoch eine Mehrzahl wo Bedarf herrscht. Da diese Anwendung in ihren Zügen aber meiner Einschätzung nach einzigartig ist steigert dies ggf. das Interesse.

Es wurden nicht alle Features von GitHub bis jetzt genutzt. Viele machen auch erst Sinn wenn es um die Kooperation einer größeren Gruppe geht. Zudem sind einige Features auch erst sinnvoll sobald verschiedene Fehler parallel behoben werden müssen.

Zum Abschluss dieser Arbeit umfasst das Programm ca. 4350 Lines of Code [71]. Dabei sind viele Features noch nicht vollständig umgesetzt. Die Anzahl Zeilen gibt nicht unbedingt ein Feedback über die erbrachte Leistung, jedoch wurde in dem Projekt weitestgehend Code-Duplizierung vermieden. Die Kommentare hingegen blähen den Code wiederum auf.

Ein Bild der Hauptansicht ist in der unteren Abbildung zu sehen.

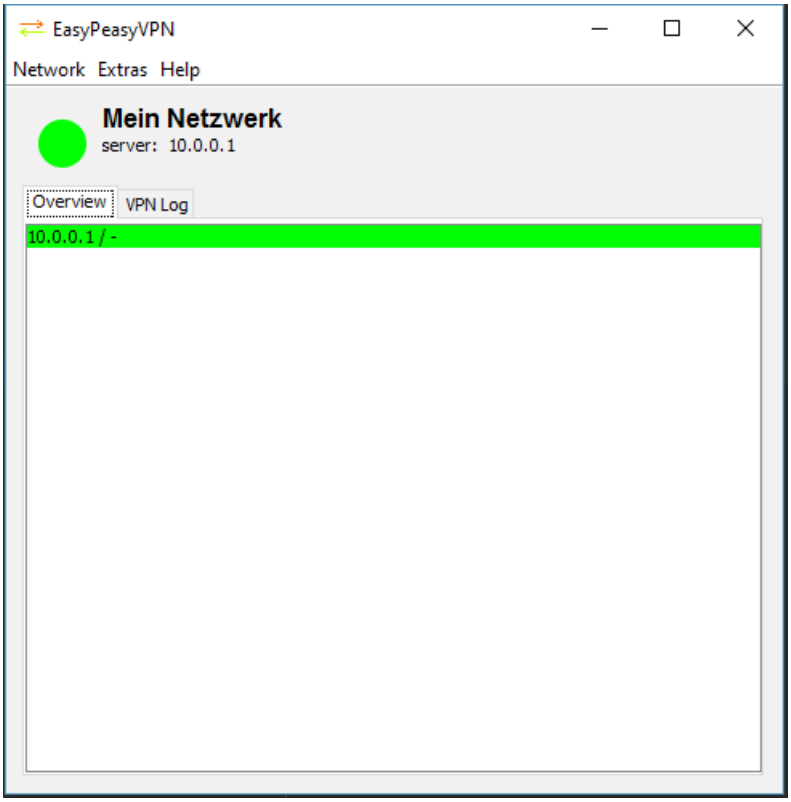


Abbildung 6: Das Hauptfenster der Anwendung. Alle Funktionen sind über die Menüleiste zu erreichen. Aktuell besteht eine Verbindung zum Netzwerk "Mein Netzwerk" mit der Rolle "Server" mit der IP "10.0.0.1", dies ist der Einzige Teilnehmer.

3.2 Dokumentation

Das Programm wurde entworfen, um möglichst intuitiv und leicht ein VPN erstellen zu können. Daher sollte es auch keine umfassende Dokumentation oder Handbuch erfordern. Alle benötigten Informationen werden im GitHub-Wiki veröffentlicht. Auf dieses wird auch aus der Anwendung selbst verwiesen. Für zusätzliche Hintergrundinformationen wird dieses Dokument voraussichtlich auch auf GitHub zur Verfügung gestellt. Falls Fragen offen bleiben sollten, bietet GitHub eine Foren ähnliche Funktion, die Issues. Hier könnten bei Problemen oder Fragen eben diese geklärt werden.

Grundsätzlich ist es primär notwendig den Aufbau einer Verbindung aus Client- und Server-Sicht zu erklären.

3.3 Ausblick

Die Anwendung ist aktuell noch in einem sehr frühen Stadium. Es gibt noch einige Probleme zu beheben und zu verbessern. Es gibt auch noch viele Features die nicht

umgesetzt oder abgeschlossen sind. Außerdem bietet die Anwendung noch immenses Erweiterungspotential. Darunter eine Vielzahl an zusätzlichen Features, wie etwa einen integrierten Chat der über das VPN funktioniert und somit direkt verschlüsselt ist. Außerdem wäre es sinnvoll die Funktionen für Hole-Punching sowohl für UDP als auch TCP funktionsfähig zu gestalten. Auch die Schritt-für-Schritt Konfiguration wäre ohne großen Aufwand umsetzbar. Weiterhin ist es sinnvoll mehrere VPN-Adapter, wie etwa SoftEther VPN, zu unterstützten.

Da die Software kein wirtschaftliches Interesse wecken kann, da sie bereits Open-Source ist ist es unwahrscheinlich das in nächster Zeit große Fortschritte kommen werden. Außerdem gibt es bereits schon ausgereifte Konkurrenzprodukte, daher wird es sobald auch keine größere Nachfrage geben. Auch ich selbst werde weniger Zeit haben mich mit dieser Anwendung auseinanderzusetzen. Daher ist es wichtig das alle Probleme und aktuellen Aufgaben festgehalten werden, sodass egal zu welchem Zeitpunkt ein Ein- und weiterarbeiten möglich ist. Daher wurde eine Funktion des IntelliJ genutzt. Alle Dinge, die es zu prüfen und erarbeiten gilt, sind im Code mit „TODO“ markiert. Aktuell sind das ca. 87 offene Punkte in 21 Dateien.

Aktuell sollte die Anwendung nur zu Vorführungs- und Entwicklungsgründen genutzt werden, da zunächst die Sicherheitsmängel beseitigt werden sollten.

3.4 Kritische Auseinandersetzung und Fazit

In dieser Arbeit wurden viele Konzepte zum Thema Netzwerktechnik und Anwendungsentwicklung betrachtet. Durch den Umfang wurde mehr auf grundlegende Dinge eingegangen und auf zielführende Thematiken beschränkt.

Es wurde erfolgreich eine Anwendung entwickelt die die zuvor behandelten Grundlagen umsetzt. So lässt sich via GUI einfach ein VPN starten und für die Verwendung sind eigentlich keine Kenntnisse nötig. Ein großes Feature wurde trotz viel Zeit und Mühe aber dennoch nicht fertig. Das Hole-Punching für Kommunikation über NAT hinweg wäre eine sehr nützliche Kernfunktion gewesen. Ohne dies muss in der Praxis auf Portforwarding zurückgegriffen werden. Dies setzt mehr Kenntnisse voraus als erhofft.

Außerdem sind aktuell noch viele Punkte offen und gerade die Sicherheit lässt noch zu wünschen übrig. Zudem wurde das Programm bisher nur im Rahmen von sehr

geringer Nutzerzahl betrachtet und erfordert nach Fertigstellung eines ausgereiften Prototypen ausgiebiges Testen.

Dafür steht aber zunächst das Grundgerüst für eine Anwendung mit viel Potential. Das Ergebnis ist zufriedenstellend, auch wenn zu Beginn ein größeres Ziel angestrebt worden ist.

Glossar

Client: Zu Deutsch Klient, Teilnehmer.

Server: Dienstleister der es Teilnehmern ermöglicht den Dienst zu nutzen.

Host: Teilnehmer eines Netzes der Betreiber eines Servers.

Hosten: Betreiben eines Servers oder ähnlichem Service.

Intranet: Das lokale Netzwerk bis zum Gateway

Extranet: Alles was nicht zum Intranet gehört

Repository: Lager, hier im Sinne von Quellcode Lager

Broadcast: Ausstrahlen, hier auch für eine ungerichtete Anfrage an alle Teilnehmer

IDE: Zu Deutsch Integrierte Entwicklungsumgebung ist eine einzelne Anwendung die alle Grundlegenden Features für die jeweilige Entwicklung bereitstellt

Branching-Modell: Ein Modell das den Arbeitsablauf für verschiedene Entwicklungszweige der selben Software bereitstellt.

Controller: Steuereinheit, etwas das die grundlegende Kommunikation zwischen Komponenten übernimmt.

Kommandozeile: Anwendung die fast jedes Betriebssystem bereitstellt und es mittels Texteingaben ermöglicht das System zu bedienen.

Packages: Packete, Ordner für Java Klassen Struktur.

Delay / Latenz :Verzögerungszeit, hier die Zeit die für die erfolgreiche Übertragung zwischen zwei Clienten oder zum Server benötigt wird.

Panel: Hier ein Abschnitt der grafischen Oberfläche.

Feature: Merkmal, hier insbesondere Software Funktionen und Eigenschaften.

Portforwarding: Portweiterleitung, der Router leitet Anfragen auf bestimmte Ports zum Zielrechner auf einem anderem Port weiter.

Code-Review: Überarbeitung und Kontrolle des Codes durch einen anderen Entwickler

Reflection: Das Betrachten und Anlaysieren von Code durch Code.

Glossar

Assertion: Das Überprüfen eines Codeabschnittes auf Vor- und Nachbedingungen.

Readme: Eine Textdatei die zusätzliche Informationen enthält und als erstes betrachtet werden sollte.

AES: Ein Verschlüsselungsalgorithmus der weit verbreitet ist.

AES-256-CBC: AES mit 256-Bit Verschlüsselug im CBC Modus.

JAR-Archiv: Eine Datei die auf der JVM ausgeführt werden kann.

JVM: Virtuelle Maschine auf der der Java-Bytecode interpretiert wird.

Abkürzungsverzeichnis

LAN: Local Area Network

W-Lan: Wireless LAN.

VPN: Virtuelles privates Netzwerk.

IP: Internet Protokoll, manchmal auch ein Synonym für IP-Adresse

NAT: Network Adress Translation

IDE: Integrated Developement Environment zu Deutsch etwa. Entwicklungsumgebung

UDP: User Datagram Protocol

TCP: Transmission Control Protocol

AES: Advanced Encryption Standard:

JVM: Java Virtual Machine

Literaturverzeichnis

1: Dirk H. Traeger und Andreas Volk, Netzarten, Topologien und Zugriffsverfahren, 2001, https://link.springer.com/chapter/10.1007/978-3-322-96795-4_3

2: Duden, Router, 2017, http://www.duden.de/rechtschreibung/Router_Vermittler_Vorrichtung

3: Axel Wagner, Kommunikation in Rechnernetzen, 2009, http://www.saar.de/~awa/data/Rechnernetze.pdf

4: Patrick Schnabel, IP-Routing, 2016, https://www.elektronik-kompendium.de/sites/net/0903151.htm

5: Patrick Schnabel, DHCP - Dynamic Host Configuration Protocol, 2016, https://www.elektronik-kompendium.de/sites/net/0812221.htm

6: Lars Schlageter, WAS IST DAS ISO/OSI MODELL?, -, http://www.it-zeugs.de/was-ist-das-iso-osi-modell.html

7: Patrick Schnabel, VPN - Virtual Private Network, 2015, https://www.elektronik-kompendium.de/sites/net/0512041.htm

8: Berliner Beauftragter für Datenschutz und Informationsfreiheit, Verfügbarkeit, Integrität, Vertraulichkeit, Authentizität, 2008, https://web.archive.org/web/20151003040039/http://www.datenschutz-berlin.de:80/content/technik/begriffsbestimmungen/verfuegbarkeit-integritaet-vertraulichkeit-authentizitaet

9: Michael Kostka, VPN Arten in der Übersicht, 2014, https://www.foxplex.com/sites/vpn-arten-in-der-uebersicht/

10: Patrick Schnabel, Tunneling-Protokolle (VPN), 2015, https://www.elektronik-kompendium.de/sites/net/1410141.htm

11: Patrick Scnabel, PPTP - Point-to-Point Tunneling Protocol, -, https://www.elektronik-kompendium.de/sites/net/0906141.htm

12: Jürgen Schmidt, Der Todesstoß für PPTP, 2012, https://www.heise.de/security/artikel/Der-Todesstoss-fuer-PPTP-1701365.html

13: Patrick Schnabel, L2TP - Layer-2-Tunneling-Protocol, -, https://www.elektronik-kompendium.de/sites/net/0906131.htm

14: Patrick Schnabel, IPsec - Security Architecture for IP, -, https://www.elektronik-kompendium.de/sites/net/0906191.htm

15: athukral, How Does NAT-T work with IPSec?, 2017, https://supportforums.cisco.com/t5/security-documents/how-does-nat-t-work-with-ipsec/ta-p/3119442

16: Bryan Ford Pyda Srisuresh und Dan Kegel, Peer-to-Peer Communication Across Network Address Translators, 2005, http://www.brynosaurus.com/pub/net/p2pnat/

17: Douglas Crawford, OpenVPN over TCP vs. UDP: what is the difference, and which should I choose?, 2013

18: Dusan Zivadinovic, Boom der virtuellen Netze, 2006, https://www.heise-gruppe.de/presse/Boom-der-virtuellen-Netze-1610723.html

19: http://vpn.net

20: -, LogMeIn Hamachi Security, -, https://www.vpn.net/security

21: LogMeIn, Inc., Hamachi Security, 2015, https://secure.logmein.com/welcome/documentation/EN/pdf/Hamachi/LogMeIn_Hamachi_SecurityWhitepaper.pdf

22: SANDOR PALFY, CHANGES TO HAMACHI ON NOVEMBER 19TH, 2012, https://blog.logmein.com/products/changes-to-hamachi-on-november-19th

23: Patrick Schnabel, Hamachi, -, http://www.elektronik-kompendium.de/sites/net/1706191.htm

24: https://www.tunngle.net/

25: -, Verschlüsselung, -, https://www.tunngle.net/community/s/?l=&sec=tfree&sl=strong-encryption

26: Mike Harper, LINUX Tunngle Client, 2013

27: https://openvpn.net

28: -, Security Overview, 2017, https://openvpn.net/index.php/open-source/documentation/security-overview.html

29: Marcel Peters, IPSec vs. OpenVPN - ein Vergleich, 2016, http://praxistipps.chip.de/ipsec-vs-openvpn-ein-vergleich_50905

30: -, OpenVPN vs. IPSec - Pros and Cons, what to use?, 2014, https://serverfault.com/questions/202917/openvpn-vs-ipsec-pros-and-cons-what-to-use

31: -, Comparison of VPN protocols, -, https://www.ivpn.net/pptp-vs-l2tp-vs-openvpn

32: https://www.docker.com

33: https://github.com

34: -, How developers work, -, https://github.com/features

35: https://about.gitlab.com

36: Achmed T., Vorteile/Nachteile von Open Source Software, 2014, https://wiki.opensourceecology.de/Vorteile/Nachteile_von_Open_Source_Software

37: Julius;MScharwies;Apsel;Woodfighter, Grundlagen/HTTPS und TLS, 2017, https://wiki.selfhtml.org/wiki/Grundlagen/HTTPS_und_TLS

38: Roland Bless, Hans-Joachim Hof, Michael Conrad, Kendy Kutzner, Stefan Mink, Sichere Netzwerkkommunikation - Grundlagen, Protokolle und Architekturen, 2005

39: T. Dierks;E. Rescorla, The Transport Layer Security (TLS) Protocol, 2008, https://tools.ietf.org/pdf/rfc5246.pdf

40: Holly Lynne McKinley, SSL and TLS: A Beginners' Guide, 2003, https://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029

41: http://www.nord-com.net/h-g.mekelburg/krypto/mod-asym.htm#rsa

42: Hartmut Härtl, Asymmetrische Verschlüsselungüber so genannte „Public-Key-Verfahren", 2006 , http://www.oszhandel.de/gymnasium/faecher/informatik/krypto/rsa.htm#_ftn2

43: https://www.test.de/Internetsicherheit-Yubikey-kleiner-Schluessel-fuer-grossen-Schutz-4807972-4807984/

44: Benjamin Schischka, Wichtige Dienste per Zwei-Faktor-Authentifizierung schützen, 2017, https://www.pcwelt.de/ratgeber/Wichtige_Dienste_per_Zwei-Faktor-Authentifizierung_schuetzen-Sicherheit-8679969.html

45: -, M 4.133 Geeignete Auswahl von Authentikationsmechanismen, 2013, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m04/m04133.html

46: Patrick Schnabel, SSL-VPN, -, https://www.elektronik-kompendium.de/sites/net/1410151.htm

47: Python Software Foundation , What is Python? Executive Summary, 2017, https://www.python.org/doc/essays/blurb/

48: Python Software Foundation, What is Python good for?, 2017, https://docs.python.org/3/faq/general.html#what-is-python-good-for

49: Lutz Prechelt, An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl, 2000, https://www.nsl.com/papers/phone/jccpprtTR.pdf

50: SoftEther Project, SoftEther VPN Open Source, 2015, https://www.softether.org

51: https://github.com/SoftEtherVPN/SoftEtherVPN

52: https://github.com/OpenVPN/openvpn/

53: Eugene Toporov, IntelliJ IDEA is the base for Android Studio, the new IDE for Android developers, 2013, https://blog.jetbrains.com/blog/2013/05/15/intellij-idea-is-the-base-for-android-studio-the-new-ide-for-android-developers/

54: bluesource Expertenteam, Das Branching-Modell Git-Flow, 2014, http://www.app-entwicklung.info/2014/12/das-branching-modell-git-flow/

55: https://www.raspbian.org

56: https://www.cubietruck.com

57: http://cubian.org

58: , , , https://github.com/Hatzen/EasyPeasyVPN/tree/d2e31e5fd643ee5b597f83429f2907b200226370/src/de/hartz/vpn/main/installation

59: , , , https://github.com/Hatzen/EasyPeasyVPN/blob/d2e31e5fd643ee5b597f83429f2907b200226370/src/de/hartz/vpn/main/server/ConfigState.java

60: Jörg Czeschla, Was versteht man unter Model View Controller (MVC)?auto, 2016, https://javabeginners.de/Design_Patterns/Model-View-Controller.php

61: https://stackoverflow.com/questions/2056/what-are-mvp-and-mvc-and-what-is-the-difference

62: Ulrich Frank & Sören Halter, Delegation: Eine sinnvolle Ergänzung gängigerobjektorientierter Modellierungskonzepte, 1996, https://www.wi-inf.uni-due.de/FGFrank/documents/Zeitschriftenartikel/Mobis96.pdf

63: Joseph Bergin, Patterns for Java Events, 2000, http://csis.pace.edu/~bergin/patterns/event.html

64: https://wiki.ubuntuusers.de/APT/

65: , , , https://github.com/Hatzen/EasyPeasyVPN/blob/master/src/de/hartz/vpn/main/server/AuthenticateUser.java

66: https://stackoverflow.com/questions/43487340/java-client-php-server-udp-hole-punching-example-code/45909458#45909458

67: https://stackoverflow.com/questions/34400249/java-de-serialization-backward-compatibility

68: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

69: Thomas Bär & Frank-Michael Schlede , Warum Java solch ein Risiko darstellt, 2014, https://www.computerwoche.de/a/warum-java-solch-ein-risiko-darstellt,2538751

Literaturverzeichnis

70: https://github.com/Hatzen/EasyPeasyVPN

71: , , ,
https://github.com/Hatzen/EasyPeasyVPN/commit/7c1f2ecc84a09ea3cc3192ab84cd37b2e837e329

Abbildungsverzeichnis

Abbildung 1.....10

Abbildung 2.....17

Abbildung 3.....19

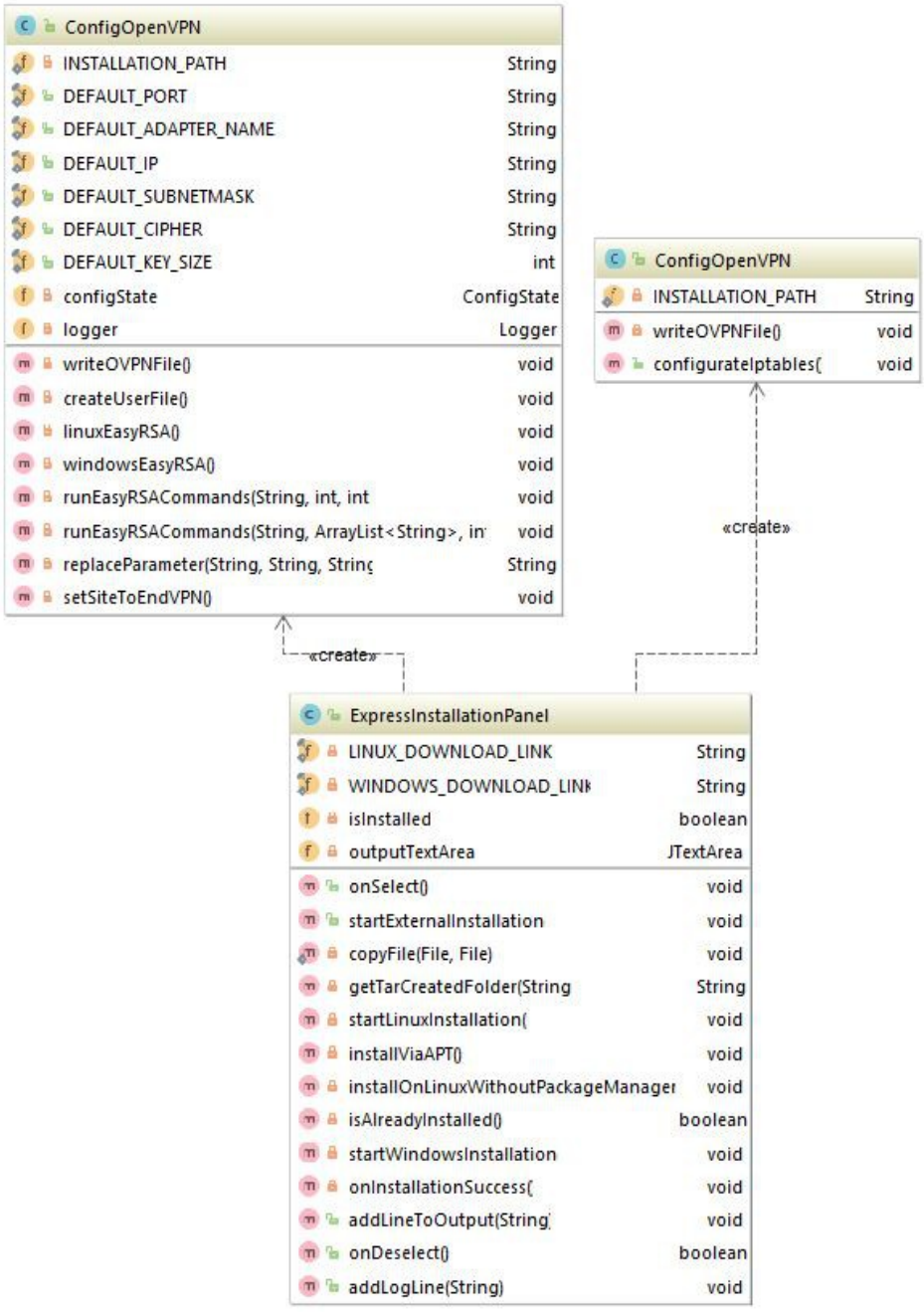
Abbildung 4.....33

Abbildung 5.....39

Abbildung 6.....41

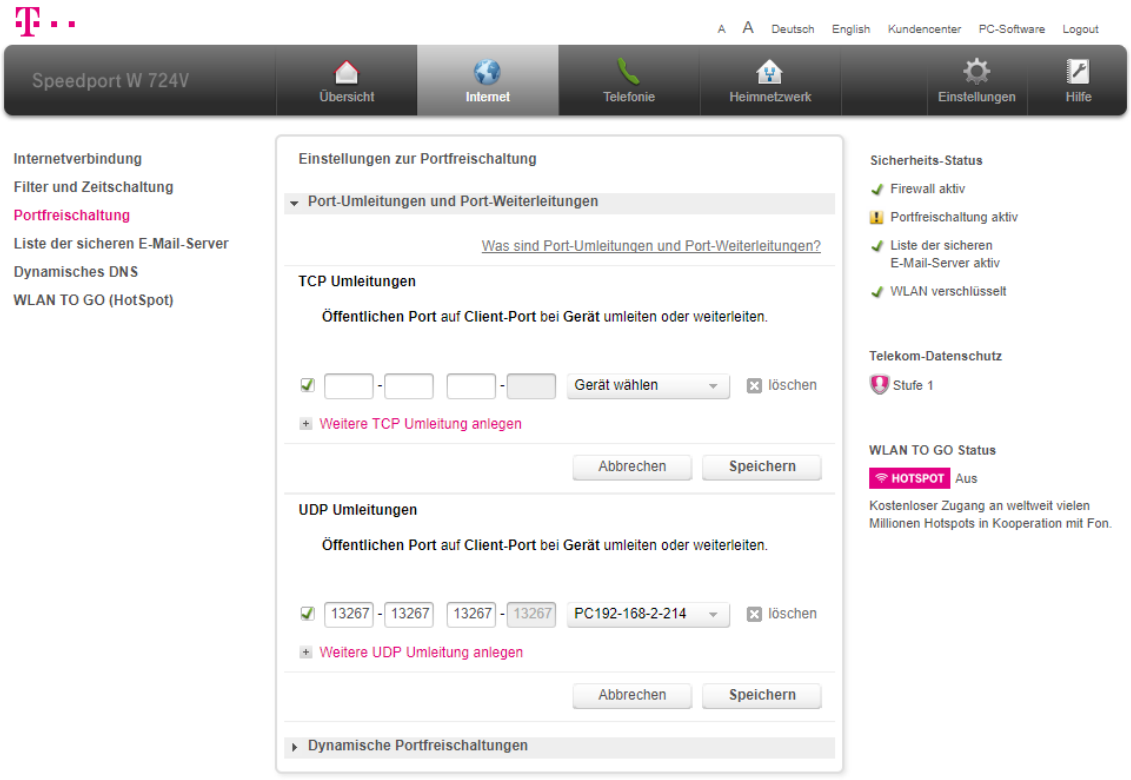
Anhang

1 Klassendiagramm VPN Konfiguration



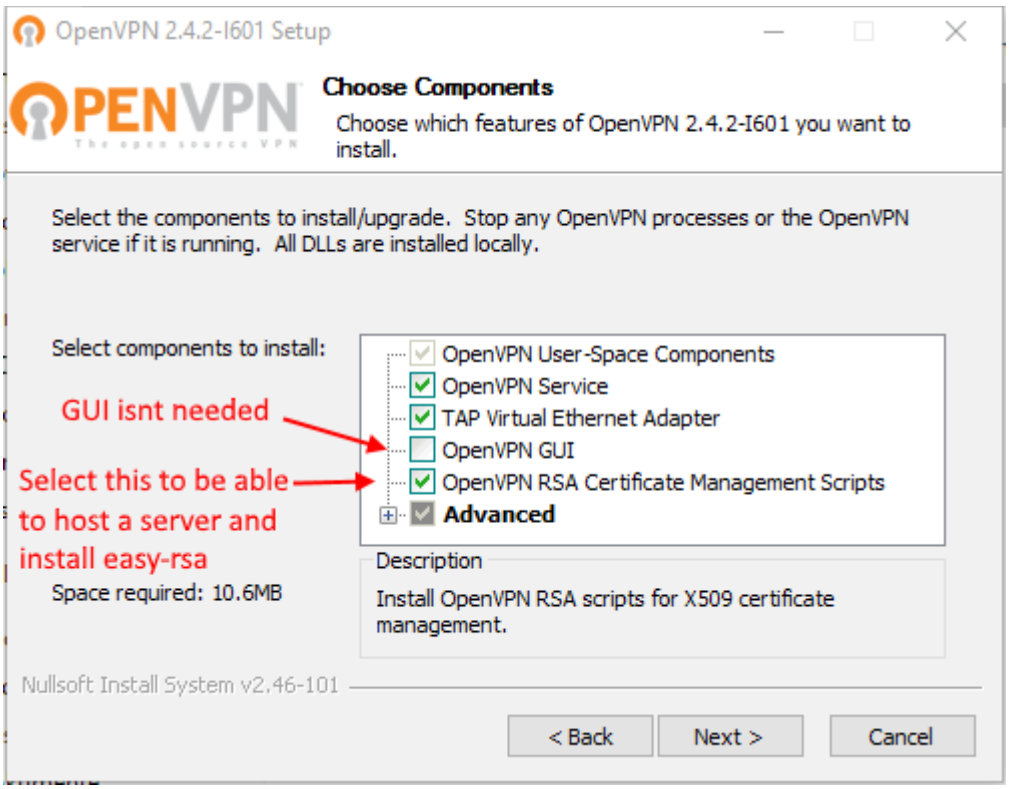
Angelehnt an ein Klassendiagramm. Die Darstellung der Attribute und Methoden der Konfiguration für den Server (Links) und den Client (Rechts). Der Server benötigt bei weitem mehr Schritte.

2 Beispielkonfiguration Router Portforwarding



Exemplarisches Beispiel zur Einrichtung einer Portweiterleitung. Hier mit dem Telekom Router Speedport W724V.

3 OpenVPN Windows Installation



Installationshinweis: Für den VPN-Server wird diese Konfiguration unter Windows benötigt.

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit über

A Tool For Creating Ad-Hoc-VPNs

selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommenen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind

Kai Hartz, Münster, 18.09.17