

# **Untersuchung der Performanz des Invisible Internet Protocols (I2P)**

Bachelorarbeit FS2021

Studentin / Student  
Moritz Küttel

Betreuer:  
Dr. Dieter Arnold

Hochschule Luzern - Departement Informatik  
19. September 2021  
Version 1.0

## Bachelorarbeit an der Hochschule Luzern – Informatik

**Titel:** Untersuchung der Performance des Invisible Internet Protocols (I2P)

**Studentin/Student:** Moritz Küttel

**Studentin/Student:** -

**Studiengang:** BSc Informatik

**Jahr:** 2021

**Betreuungsperson:** Dr. Dieter Arnold

**Expertin/Experte:** Urs Rufer

**Auftraggeberin/Auftraggeber:** Carolyn Bächler, Konrad Bächler, DIVA.EXCHANGE

### Codierung / Klassifizierung der Arbeit:

- ☒ A: Einsicht (Normalfall)
- ☐ B: Rücksprache (Dauer:     Jahr / Jahre)
- ☐ C: Sperre (Dauer:     Jahr / Jahre)

**Eidesstattliche Erklärung** Ich erkläre hiermit, dass ich/wir die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe angefertigt haben, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben haben, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht haben, das Vertraulichkeitsinteresse des Auftraggebers wahren und die Urheberrechtsbestimmungen der Fachhochschule Zentralschweiz (siehe Merkblatt «Studentische Arbeiten» auf MyCampus) respektieren werden.

Ort / Datum, Unterschrift \_\_\_\_\_

Ort / Datum, Unterschrift \_\_\_\_\_

**Abgabe der Arbeit auf der Portfolio Datenbank:**

Bestätigungsvisum Studentin/Student

Ich bestätige, dass ich die Bachelorarbeit korrekt gemäss Merkblatt auf der Portfolio Datenbank abgelegt habe. Die Verantwortlichkeit sowie die Berechtigungen habe ich abgegeben, so dass ich keine Änderungen mehr vornehmen kann oder weitere Dateien hochladen kann.

Ort / Datum, Unterschrift \_\_\_\_\_

Ort / Datum, Unterschrift \_\_\_\_\_

**Ausschliesslich bei Abgabe in gedruckter Form:  
Eingangsvisum durch das Sekretariat auszufüllen**

Rotkreuz, den \_\_\_\_\_ Visum: \_\_\_\_\_

**Verdankung**

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei dieser Bachelorarbeit unterstützt haben.

Als allererstes möchte ich meinem Betreuer Dieter Arnold danken, der meine Bachelorarbeit betreut und begutachtet hat und stets für meine Fragen und Anliegen da war. Insbesondere möchte ich mich für seine aussergewöhnliche Unterstützung während dieser schwierigen Zeit bedanken.

Ich bedanke mich bei Konrad und Carolyn Bächler vom Verein DIVA.EXCHANGE, für diese tolle Projektidee, und für die hervorragende Arbeitsumgebung in ihrem Büro in Baar. Bedanken möchte ich mich auch für die vielen Diskussionen und Gespräche sowie Ideen und Anregungen, welche diese Arbeit in eine bessere Richtung gelenkt haben.

Ausserdem möchte ich meiner Mutter Bernadette Lüönd, sowie meinem guten Freund Moritz Köhler danken für das Korrekturlesen meiner Arbeit danken und dafür dass sie immer ein offenes Ohr für mich hatten. Ebenfalls möchte ich mich bei meiner guten Freundin Linda Hauser bedanken für ihren Rückhalt, sowie für die Hilfe beim erstellen des Pitching-Videos.

**Hinweis:** Die Bachelorarbeit wurde von keinem Dozierenden nachbearbeitet. Veröffentlichungen (auch auszugsweise) sind ohne das Einverständnis der Studiengangleitung der Hochschule Luzern – Informatik nicht erlaubt.

Copyright © 2021 Hochschule Luzern – Informatik

Alle Rechte vorbehalten. Kein Teil dieser Arbeit darf ohne die schriftliche Genehmigung der Studiengangleitung der Hochschule Luzern – Informatik in irgendeiner Form reproduziert oder in eine von Maschinen verwendete Sprache übertragen werden.

## **Zusammenfassung**

Der Verein DIVA.EXCHANGE entwickelt einen Software-Prototypen für eine Handelsplattform, um digitale Werte auszutauschen. Die Handelsplattform soll vollständig verteilt sein und es sollen digitale Werte ausgetauscht werden können, ohne dass sich die Benutzer gegenseitig kennen oder vertrauen müssen. Um dies umzusetzen, wurde auf das Anonymisierungsnetzwerk "The Invisible Internet Protocol" (I2P) als Grundstein auf der Netzwerkebene gesetzt. Jedoch hat die Anonymität, Privatsphäre und Sicherheit, die durch I2P geboten wird, ihren Preis: Performanz. In dieser Arbeit soll empirisch untersucht werden, unter welchen Umständen sich die Performanz des I2P-Netzwerks verbessert. Insbesondere wurden die Latenzzeiten von TCP-Nachrichten untersucht.

Damit die Performance-Messungen empirisch durchgeführt werden konnten, wurde ein Teststand entwickelt an dem Experimente durchgeführt wurden. In einem privaten I2P-Netzwerk konnten Latenzmessungen ohne äussere Einflüsse durchgeführt werden. Damit konnte ein Testnetzwerk bestehend aus bis zu 256 I2P-Knoten erstellt werden. Somit konnten Latenzmessungen durchgeführt werden, welche aufgezeigt haben, dass sich entgegen unserer Erwartungen die Latenz bei einem grösseren Netzwerk, worin alle Knoten jeweils dieselbe Bandbreitenlimite haben, nicht gleich bleibt, sondern verschlechtert. Grund dafür könnte die verwendete Testinfrastruktur sein, die nur horizontal skaliert.

Die gewonnenen Resultate dienen aber als Grundlage und Referenz für zukünftige Messungen und Experimente. Anhand des entwickelten Teststands können nun weitere Messungen getätigt und miteinander verglichen werden. Grundsätzlich können so auch verschiedene I2P-Konfigurationen miteinander verglichen werden. In Zukunft könnte die Testinfrastruktur so erweitert, dass nicht nur horizontal sondern auch vertikal skaliert werden kann.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Aufgabe und Problemstellung . . . . .	1
1.2	Ziel und Vision . . . . .	2
<b>2</b>	<b>Stand der Technik</b>	<b>3</b>
2.1	Einführung in die Technologie . . . . .	3
2.1.1	Privatsphäre und Anonymität . . . . .	3
2.1.2	Anonymisierungsnetzwerke . . . . .	4
2.1.3	Das Anonymitätstrilemma . . . . .	4
2.1.4	The Invisible Internet Protocol (I2P) . . . . .	5
2.2	Technische Konzepte von I2P . . . . .	5
2.2.1	I2P-Router . . . . .	5
2.2.2	Destinations . . . . .	6
2.2.3	Tunnels . . . . .	6
2.2.4	NetDb . . . . .	7
2.2.5	Floodfill-Router . . . . .	8
2.2.6	Peer-Profiling, Selektion und Tunnel-Erstellung . . . . .	9
2.2.7	Garlic-Routing . . . . .	9
2.2.8	I2P-Testnetzwerke und Bootstrapping . . . . .	10
2.2.9	Funktionsweise der Blockchain von DIVA.EXCHANGE . . . . .	10
<b>3</b>	<b>Ideen und Konzepte</b>	<b>12</b>
3.1	Reproduzierbarkeit . . . . .	12
3.2	Infrastruktur und Deployment . . . . .	12
3.3	Isolierung des Testnetzwerks . . . . .	12
3.4	Bootstrapping . . . . .	14
3.5	Konfigurierbarkeit . . . . .	14
3.6	Latenzmessung . . . . .	15
3.7	Limitieren der Bandbreite . . . . .	15
3.8	Messen der Ressourcenauslastung . . . . .	15
<b>4</b>	<b>Methode</b>	<b>16</b>
4.1	Experiment . . . . .	16
4.2	Projektmanagement . . . . .	16
4.2.1	Vorgehensmodell . . . . .	16
4.2.2	Projektorganisation . . . . .	17
4.2.3	Projektanforderungen . . . . .	18
4.2.4	Planung . . . . .	20

<b>5</b>	<b>Realisierung</b>	<b>22</b>
5.1	Systemarchitektur . . . . .	22
5.2	Komponentendesign . . . . .	22
5.2.1	Deployment der Test-VM . . . . .	24
5.2.2	Testkonfiguration . . . . .	24
5.2.3	Recompose-Skript . . . . .	24
5.2.4	i2pd-Container . . . . .	24
5.2.5	Reseed-Server . . . . .	24
5.2.6	Messskript . . . . .	25
5.3	Umsetzung . . . . .	25
5.4	Lösungsansatz: NixOS-VMs und NixOS-Container . . . . .	25
5.5	Lösung mit Docker-Compose . . . . .	26
5.5.1	Skalierung . . . . .	26
5.5.2	Bootstrapping und Reseed-Server . . . . .	28
5.5.3	Konfigurationsoptionen . . . . .	31
5.5.4	Netzwerkisolation . . . . .	32
5.5.5	Adressbuch . . . . .	32
5.5.6	TCP-Server . . . . .	33
5.5.7	TCP-Client . . . . .	34
5.5.8	Nachricht zur Latenzmessung . . . . .	35
5.5.9	Das Messskript . . . . .	35
5.6	Testing . . . . .	36
5.7	Benutzerhandbuch . . . . .	36
5.7.1	Deployment der Test-VM . . . . .	36
5.7.2	Konfiguration einer Messung . . . . .	37
5.7.3	Erstellen des Testnetzwerkes . . . . .	37
5.7.4	Ausführen einer Messung . . . . .	37
<b>6</b>	<b>Evaluation und Validation</b>	<b>38</b>
6.1	Messresultate . . . . .	38
6.1.1	Validation der Testumgebung . . . . .	38
6.1.2	Gleichbleibende Latenz bei Skalierung des Netzwerks . . . . .	39
6.1.3	Verhalten der Latenz abhängig von der Nachrichtengrösse . . . . .	39
6.2	Vergleich mit Anforderungen . . . . .	41
6.3	Technische Aspekte . . . . .	42
6.4	Vorgehen . . . . .	43
<b>7</b>	<b>Ausblick</b>	<b>44</b>
7.1	Projektfazit . . . . .	45
7.2	Persönliches Fazit . . . . .	45
<b>A</b>	<b>Aufgabenstellung</b>	<b>VI</b>
<b>B</b>	<b>Resultate</b>	<b>X</b>
<b>C</b>	<b>Journal</b>	<b>XIII</b>

# 1. Einleitung

Das Internet Protokoll (IP) ist einer der wichtigsten Grundsteine für das heutige Internet. Es wurde jedoch damals nicht unter den Gesichtspunkten wie Datenschutz, Privatsphäre und Sicherheit designt, da es wohl eher darum, überhaupt ein Protokoll zu haben, um Nachrichten in einem Netzwerk verschicken und von Knoten zu Knoten weiterleiten zu können. Mittlerweile gibt es jedoch verschiedene Anonymisierungsnetzwerke, welche in diesem Bereich Abhilfe schaffen könnten. Eines davon nennt sich "The Invisible Internet Protocol" (kurz I2P), welches in dieser Arbeit genauer untersucht wird. Die Performanz von Anonymisierungsnetzwerken ist aber immer schlechter als die Performanz des Internet Protokolls (IP). Denn einerseits verwenden die Anonymisierungsnetzwerke das Internet Protokoll (IP) als Basis und andererseits hat Anonymität in einem Netzwerk immer den Preis von höherer Netzwerkbandbreite oder höherer Latenz (siehe Abschnitt 2.1.3 "Das Anonymitätsdilemma"). Dementsprechend lässt die Performanz von Anonymisierungsnetzwerken für Benutzer oft zu wünschen übrig. Auch sonst gibt es für normale Benutzer nicht wirklich Gründe, wieso solch ein Netzwerk eingesetzt werden sollte, da es kaum Anwendungen gibt und für viele Privatsphäre nicht oberste Priorität hat.

## 1.1. Aufgabe und Problemstellung

Der Verein DIVA.EXCHANGE<sup>1</sup> entwickelt einen Softwareprototypen DIVA. Der Softwareprototyp soll aufzeigen, dass es möglich ist, eine vollständig verteilte Handelsplattform zu entwickeln, die sowohl sicher ist, als auch die Privatsphäre der Benutzer schützt. Benutzer sollen digitale Werte austauschen können, ohne sich dabei zu kennen und ohne sich gegenseitig vertrauen zu müssen. Es handelt sich hierbei um ein freies Softwareprojekt<sup>2</sup>.

Der Softwareprototyp besteht aus drei Schichten. Die Handelsplattform und die dazugehörige Verwaltungssoftware stellen die oberste Schicht dar. Darunter befindet sich eine Datenhaltungs- und Speicherschicht, basierend auf einer im Haus entwickelten Blockchain namens Divachain. Auf der untersten Ebene, der Netzwerkschicht, wird I2P verwendet. I2P soll die Basis bieten für den Softwareprototypen, um von Grund auf Anonymität und Sicherheit der Kommunikation sicherzustellen (siehe Abbildung 1.1 "Übersicht DIVA.EXCHANGE").

Es besteht die Annahme, dass die Performanz des I2P-Netzwerks nicht ausreichend ist, um für Endbenutzer schnell reagierende Applikationen auf dem Netzwerk anbieten zu können. Das Netzwerk ist verglichen mit dem TOR-Netzwerk klein und es können grössere Latenzen entstehen, um Anonymität für die Netzwerkteilnehmer zu bieten. Kurz vor Beginn dieser Arbeit dauerte das Hin- und Zurücksenden einer Nachricht durch das I2P-Netzwerk (Roundtrip) etwa drei bis acht Sekunden. Dies hat mehrere Gründe:

- Eine Nachricht wird über mehrere Hops versendet und wird mehrmals weitergeleitet bevor sie beim Empfänger ankommt.

---

<sup>1</sup>Webauftritt: <https://diva.exchange>

<sup>2</sup>Was ist "freie Software"? Siehe <https://www.gnu.org/philosophy/free-sw.de.html>





Abbildung 1.1.: Übersicht DIVA.EXCHANGE

- Fallen Hops, aus müssen die Nachrichten über eine andere Route neu übermittelt werden.
- Jede Nachricht ist mehrfach verschlüsselt und jeder Knoten muss jeweils eine Verschlüsselschicht beim Weiterleiten einer Nachricht entfernen, was viele CPU-Zyklen kostet.

Im Rahmen dieser Arbeit soll folgende Problemstellung untersucht werden:

**Hypothese 1 (H1):** *Ist eine steigende Anzahl von I2P-Knoten für die I2P-Netzwerk-Latenz (tiefer) vorteilhaft? Können Nachrichten schneller vom Sender zum Empfänger gelangen je mehr Knoten das I2P-Netzwerk hat?*

Damit Applikationen auf dem I2P-Netzwerk gut funktionieren, soll ermittelt werden, wie die Performance verbessert werden kann. Siehe auch die komplette Aufgabenstellung, die im Anhang A "Aufgabenstellung" zu finden ist.

## 1.2. Ziel und Vision

Ziel ist es, aufzuzeigen, unter welchen Umständen und Rahmenbedingungen Anwendungen auf dem I2P-Netzwerk kürzere Latenzzeiten aufweisen und somit aus Sicht eines Endbenutzers schneller reagieren. Das Niveau an Anonymität soll aber zum Schutz der Privatsphäre nicht eingeschränkt werden. Wird festgestellt, dass mehr Knoten in einem I2P-Netzwerk die Latenz verringern, könnte es unter Umständen mehr Personen dazu bewegen selber I2P-Knoten zu betreiben. Dies wiederum würde das Netzwerk stärken und diverse Netzwerkeffekte könnten auftreten. Zum Beispiel könnte dies dazu führen, dass mehr Entwickler Applikationen für das Netzwerk erstellen und es auch aus Benutzersicht attraktiver wird.

## 2. Stand der Technik

Dieses Kapitel zeigt den aufgrund der Zielsetzung und Fragestellung ermittelten Stand der Technik auf. Dies beinhaltet die Resultate der Recherche, sowie einen Beschrieb der Technologien auf welchen diese Arbeit aufbaut. Im ersten Abschnitt 2.1 "Einführung in die Technologie" wird ein Einstieg in das Thema Anonymisierungsnetzwerke geboten, wobei die Begrifflichkeiten Privatsphäre und Anonymität erklärt sowie grundsätzliche Einschränkungen von Anonymisierungsnetzwerken erläutert werden. Im zweiten Abschnitt 2.2 "Technische Konzepte von I2P" wird genauer auf die Technologie von I2P eingegangen. Darin wird die Funktionsweise von I2P genauer erläutert und relevante technische Konzepte erklärt.

### 2.1. Einführung in die Technologie

Da sich diese Arbeit um ein Anonymisierungsnetzwerk dreht und somit die Privatsphäre der Benutzer gewährleistet werden soll (siehe Abschnitt 1.2 "Ziel und Vision"), werden zunächst diese Begrifflichkeiten erklärt. Anschliessend werden grundsätzliche Beschränkungen von Anonymisierungsnetzwerken aufgezeigt. Schlussendlich wird genauer auf I2P selber eingegangen, da es sich hier um die Netzwerkschicht von DIVA.EXCHANGE handelt.

#### 2.1.1. Privatsphäre und Anonymität

Im DIVA.EXCHANGE-Netzwerk sollen digitale Güter austauscht werden können. Dies ohne sich gegenseitig zu kennen und ohne sich gegenseitig vertrauen zu müssen. Das heisst in Folge, dass die Anonymität und Privatsphäre der Benutzer gewährleistet werden müssen. Anonymität und Privatsphäre sind jedoch nicht dasselbe. Im Weiteren werden diese beiden Begriffe differenziert und verglichen. Diese Thematik wurde schon in einer vorherigen Arbeit (siehe Marić, 2020, S. 9) mit DIVA.EXCHANGE behandelt, wobei bereits eine Definition von Privatsphäre und Anonymität erarbeitet wurde.

Komplette Anonymität ist dann erreicht, wenn nicht auf die wahre Identität eines Benutzers geschlossen werden kann, das heisst, es fehlen jegliche Identitätsmerkmale, die einen Hinweis auf die reale Person geben würden. Anonymität ist jedoch auch ein Spektrum. Ein Internetbenutzer kann zum Beispiel teilweise anonym sein, indem er ein Pseudonym wie einen Benutzernamen verwendet. Dabei handelt es sich bei einem Benutzernamen auch um ein Identitätsmerkmal. Wenn dieses Pseudonym jedoch mit Identitätsmerkmalen der richtigen echten Person in Zusammenhang gebracht wird, ist die Anonymität wiederum verloren. (Marić, 2020, S. 8-9)

Mit Hilfe von Anonymität kann auch die Privatsphäre geschützt werden. Anonymität ist jedoch nicht zwingend notwendig, um die Privatsphäre der Benutzer oder den Datenschutz zu gewährleisten. Privatsphäre kann zum Beispiel nur schon durch Verwendung von Verschlüsselung erreicht werden. Auch kann die Anonymität sogar verloren gehen, versucht man die Privatsphäre zu schützen, dies zum Beispiel wenn Verschlüsselung mit einer Signatur verwendet wird, die auf die wahre Identität eines Benutzers zurückschliessen lässt. Anonymität kann durch Verschlüsselung, sowie auch durch Weiterleiten oder Durchmischen von Nachrichten erreicht werden.

### 2.1.2. Anonymisierungsnetzwerke

Es gibt eine Vielzahl von verschiedenen Anonymisierungsnetzwerken. Wobei The Onion Router (TOR) wahrscheinlich das Bekannteste und meistgenutzte ist. In einer vorherigen Arbeit wurde I2P mit TOR verglichen und es wurde festgestellt, dass I2P die Voraussetzungen für Privatsphäre und Anonymität des Software-Prototypen von DIVA.EXCHANGE (Marić, 2020, S. 28-30) erfüllt. Deshalb wird I2P nun als Grundstein für das Netzwerklayer von DIVA.EXCHANGE eingesetzt. Somit wird in dieser Arbeit lediglich das Anonymisierungsnetzwerk I2P behandelt und nicht weiter auf andere Anonymisierungsnetzwerke eingegangen.

### 2.1.3. Das Anonymitätstrilemma

Beim Design eines anonymen Kommunikationsprotokolls gibt es grundsätzliche Einschränkungen, die zu beachten sind. Es gilt die folgenden drei Aspekte abzuwägen und einen Kompromiss zwischen diesen zu finden:

- Der Anonymitätsgrad von Sender und Empfänger einer Nachricht.
- Die zur Kommunikation benötigte Netzwerkbandbreite.
- Die Latenzzeiten und Verzögerungen der zu übermittelnden Nachrichten.

Dabei ist es nicht möglich ein Protokoll zu designen, welches einen hohen Anonymitätsgrad bietet, wenig Netzwerkbandbreite benötigt und tiefe Latenzzeiten vorweist. (Das et al., 2018, S. 1) Höhere Anonymität geht also immer auf Kosten von höherer Latenzzeiten oder mehr benötigter Netzwerkbandbreite, oder auf Kosten von beidem. (siehe auch Abbildung 2.1 "Das Anonymitätstrilemma").

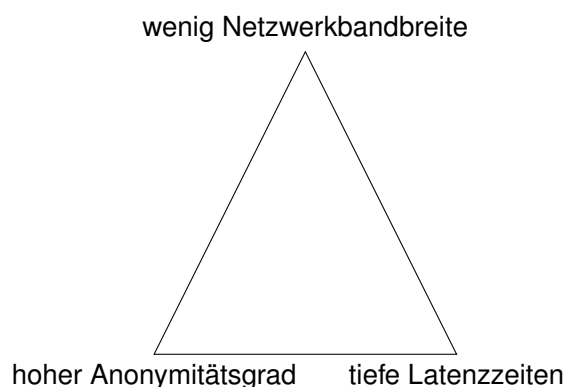


Abbildung 2.1.: Das Anonymitätstrilemma

Ein Anonymisierungsnetzwerk könnte beispielsweise auf Kosten der Netzwerkbandbreite implementiert werden, indem die Nachrichten jeweils an alle Netzwerkteilnehmer versendet werden. Somit könnte mindestens der Empfänger verschleiert werden. Auch gibt es die Möglichkeit, mehr Anonymität zu schaffen, indem der Netzwerkverkehr ergänzt wird oder vermischt wird mit anderem Netzwerkverkehr.

Umgekehrt könnte ein Anonymisierungsnetzwerk auf Kosten der Latenz implementiert werden. Werden Nachrichten über mehrere Netzwerkknoten geleitet, könnte man Empfänger und Sender verschleiern. Jedoch hätte dies höhere Latenzzeiten zur Folge, da dieselbe Nachricht mehrmals

nacheinander von verschiedenen Netzwerkknoten bearbeitet werden müsste. Zusätzlich könnte bei diesem Ansatz ein Netzwerkknoten extra ein Netzwerkpaket zwischenspeichern und zeitverzögert weiterschicken. Dies würde das Anonymisierungsnetzwerk noch besser gegen timing-basierte Deanonymisierungsangriffe und Korrelation von Netzwerkverkehr schützen. Und schliesslich gibt es Netzwerke bei welchen Anonymität keine Rolle spielt, wie beispielsweise beim üblichen “User Datagram Protocol” (UDP). Es ist dann möglich hohe Bandbreite sowie niedrige Latenzzeiten zu erreichen.

Es gilt abzuwägen, welche der Aspekte für das zu designende Protokoll wichtig sind. Es muss entschieden werden, wo man ein Schwerpunkt setzt und welche Aspekte man somit auch vernachlässigt, um einen guten Kompromiss zu finden. (Das et al., 2018, S. 1-2)

#### **2.1.4. The Invisible Internet Protocol (I2P)**

Kurz gesagt ist “The Invisible Internet Protocol” (I2P) ein dezentrales Mischnetzwerk, in dem verschlüsselte Nachrichten ausgetauscht werden können. Es handelt sich hier um ein eigenständiges Overlay-Netzwerk, welches auf den darunterliegenden Protokollen UDP und TCP aufbaut. Mit einem Mischnetzwerk ist gemeint, dass Nachrichten durch Knoten des Netzwerks weitergeleitet werden, um den Sender und den Empfänger einer Nachricht zu verschleiern. Dieser Mischvorgang wird bei I2P Garlic-Routing genannt (siehe Abschnitt 2.2.7 “Garlic-Routing”). Auch gilt es zu unterstreichen, dass *Nachrichten* ausgetauscht werden. I2P ist *nachrichtenbasiert* und verhält sich in diesem Sinne ähnlich wie UDP (Zantout & Haraty, 2011, S. 1). Es gibt dementsprechend keine Garantien, dass, wenn eine Nachricht versendet wird, die Daten in der richtigen Reihenfolge, komplett oder überhaupt ankommen. Es gibt jedoch Erweiterungen auf dem Netzwerk, welche diese Garantien (ähnlich, wie es sie bei TCP gibt) wieder herstellen können. Zudem bietet I2P eine Ende-zu-Ende-Verschlüsselung von Nachrichten. Die Nachrichten sind also vom Sender bis zum Empfänger verschlüsselt.

Während das TOR-Netzwerk ursprünglich entwickelt wurde, um auf das öffentliche Internet via Outproxies zuzugreifen, liegt bei I2P der Fokus darin anonym auf Dienste, die innerhalb des Netzwerks angeboten werden, zuzugreifen. Jedoch besteht bei TOR ebenfalls die Möglichkeit, netzwerkinterne Dienste anzubieten mittels “TOR Hidden Services”. Umgekehrt gibt es aber auch bei I2P die Möglichkeit, mit Hilfe von Outproxies, auf das normale Internet zuzugreifen (Ehlert, 2011, S. 2). Dienste, die innerhalb des Netzwerks angeboten werden, sind zum Beispiel Bit-Torrent für File-Sharing, Webseiten (sogenannte “eepsites”), oder auch Chat-Dienste wie IRC (Internet Relay Chat).

(de Boer & Breider, 2019; „I2P Compared to Freenet - I2P“, n. d., p. 3-4)

## **2.2. Technische Konzepte von I2P**

In diesem Abschnitt werden wichtige Begrifflichkeiten und technische Konzepte im Bezug auf I2P genauer erklärt und unter die Lupe genommen.

### **2.2.1. I2P-Router**

Ein I2P-Router ist ein einzelner Teilnehmer, ein Knoten oder auch ein Peer im I2P-Netzwerk. Ein Router kann Nachrichten weiterleiten und somit im Netzwerk mischen, aber auch empfangen, oder versenden. Es gibt zwei verschiedene Implementationen von I2P-Router-Software, welche

auch zusammenarbeiten können und gemeinsam ein Netzwerk bilden können. Die ursprüngliche Implementation des I2P-Routers, welche auch I2P genannt wird, wurde in Java programmiert. In dieser Arbeit wird jedoch lediglich die später entwickelte und schlankere C++-Implementation auch genannt "i2pd"<sup>1</sup> (Invisible Internet Protocol Daemon), untersucht.

### 2.2.2. Destinations

Ein I2P-Router kann eine beliebige Anzahl an Destinations (Zielorten) im I2P-Netzwerk zur Verfügung stellen. Eine Destination hat immer mindestens eine Adresse, mit welcher andere Netzwerkteilnehmer einen netzwerkinternen Service erreichen können. Wurde eine Destination mit einem Service definiert, erstellt der I2P-Router beim Aufstarten ein Schlüsselpaar. Dieses wird für die Verschlüsselte Kommunikation mit dem Service benötigt. Zudem wird aus dem Public-Key immer auch eine netzwerkinterne Adresse für den Service abgeleitet. Diese Adresse nennt sich eine b32-Adresse. Dies ist eine 52-Zeichen-lange aus Kleinbuchstaben und Zahlen bestehender Name. Zudem hat diese die Endung `.b32.i2p` damit klar ist, dass dieser Name innerhalb des Overlay-Netzwerks aufgelöst werden muss. Eine b32-Adresse sieht also zum Beispiel so aus:

`b2cdodoyqg5v6go56bkaevvw777e2wyiflqh6zyyxugy4cq1zuq.b32.i2p`. („Naming and Address Book - I2P“, n. d.)

### 2.2.3. Tunnels

Wenn man im Bereich vom Netzwerken über Tunneling redet, geht es immer darum ein Kommunikationsprotokoll in ein anderes einzubetten. („Duden | Tunneling | Rechtschreibung, Bedeutung, Definition, Herkunft“, n. d.) Die gesamte Menge an Tunnels und I2P-Router bildet somit ein Overlay-Netzwerk. Bei I2P ist ein Tunnel ein *unidirektionaler* Kommunikationsweg durch eine ausgewählte Liste an I2P-Router. Die Tunnels werden aus Anonymitätsgründen alle zehn Minuten zerstört und wieder neu aufgebaut. Ähnlich wie beim Onion-Routing wird hier die Nachricht mehrfach verschlüsselt, damit die Zwischenknoten nicht in die Nachricht hineinsehen können, jedoch trotzdem die benötigten Routing-Informationen haben, um diese an den richtigen Ort weiterzuleiten. Es wird also eine Verschlüsselungsschicht nach der anderen bei jedem Hop durch das Netzwerk abgebaut, ähnlich wie man die verschiedenen Schichten einer Zwiebel abnehmen kann.

Bei I2P gibt es verschiedene Arten von Tunnels, welche in den folgenden Abschnitten genauer differenziert werden. („Intro - I2P“, n. d.; Liu et al., 2014, S. 2)

#### Inbound- und Outbound-Tunnels

Da die Tunnels ein unidirektionalen Kommunikationsweg bieten, gibt es Inbound- und Outbound-Tunnels, wobei die Inbound-Tunnels zum Empfang und die Outbound-Tunnels zum Versand von Nachrichten dienen. Inbound-Tunnels sind also Nachrichteneingänge und Outbound-Tunnels sind Nachrichtenausgänge. Jeder I2P-Router kann eine beliebige Anzahl Inbound- und Outbound Tunnels erstellen. Dies hat zur Folge, dass wenn eine Nachricht beantwortet wird, die Antwort nicht über den gleichen Weg wieder zurückkommt. („Intro - I2P“, n. d.)

---

<sup>1</sup>Webseite: <https://i2pd.website/>

## **Exploratory-Tunnels**

Exploratory-Tunnels sind Tunnels mit tiefer Bandbreite und werden dazu verwendet, um Tunnels aufzubauen, zu verwalten, oder zu zerstören. Sie werden also nicht für sensible Operationen bezüglich Privatsphäre eingesetzt. Genauer gesagt wird mit Hilfe dieser Tunnels das I2P-Netzwerk nach weiteren I2P-Routern durchforstet, um weitere Netzwerkteilnehmer zu finden, und die NetDb (siehe 2.2.4 "NetDb") heruntergeladen. (Conrad & Shirazi, 2014, S. 4)

## **Client-Tunnels**

Client-Tunnels sind Tunnels mit grösserer Kapazität und Bandbreite. Um solche gute Tunnels zu finden und zu erstellen werden erst die Exploratory-Tunnels benötigt, welche andere I2P-Router erforschen (siehe Abschnitt 2.2.6 "Peer-Profiling, Selektion und Tunnel-Erstellung"). Sie werden einerseits zum Abfragen der LeaseSets aus der NetDb eingesetzt (siehe Abschnitt 2.2.4 "NetDb"). Andererseits sind dies auch die Tunnels, welche die Nachrichten von Applikationen und Services, die im I2P-Netzwerk angeboten werden übermitteln. (Conrad & Shirazi, 2014, S. 4; „Tunnelimplementierung - I2P“, n. d.)

## **Länge der Tunnels**

Jedes Tunnel hat eine Länge, welche von jedem I2P-Knoten selber festgelegt werden kann. Die Tunnellänge gibt an, durch wie viele Knoten eine Nachricht, die durch das Tunnel verschickt wird, geroutet wird. Für jeden Hop im Tunnel wird die Nachricht in eine zusätzliche Verschlüsselungsschicht eingehüllt, sodass wissen die Router jeweils nur Bescheid wissen, von welchem anderen Router sie eine Nachricht erhalten haben und an welchen Router die Nachricht weitergeleitet werden soll. Ein Router mitten im Tunnel weiss also schlussendlich nie, von wem und an wen eine Nachricht verschickt wurde. Die Länge eines Tunnels kann auch 0 betragen. In diesem Fall wird eine Nachricht die durch das Tunnel verschickt wird, durch keine weitere Knoten geroutet. Hierbei kann jeder Knoten selber festlegen, wie lange seine eigenen Tunnels sind. Je länger ein Tunnel ist, desto mehr Anonymität bietet er. Jedoch geht dies auf Kosten der Latenz aufgrund der zusätzlichen Hops. Die Standardlänge von Exploratory-Tunnels beträgt 2, alle anderen Tunnels haben eine Standardlänge von 3. Da der Ausgang oder das Ende eines Outbound-Tunnels eines I2P-Routers mit einem Eingang oder Anfang eines Inbound-Tunnels verbunden wird, ist die effektive Länge eines Tunnels die Länge des Inbound-Tunnels addiert mit der Länge des Outbound-Tunnels. Mehr dazu im Abschnitt 2.2.7 "Garlic-Routing".

(„I2P tunnels configuration - i2pd documentation“, n. d.)

## **2.2.4. NetDb**

Da I2P dezentral und vollständig verteilt ist, jedoch trotzdem Daten zwischen den Knoten geteilt werden müssen, wird eine Art verteilte Datenbank benötigt. Diese verteilte Netzwerkdatenbank wird NetDb genannt. Es handelt sich hier um eine Distributed Hashed Table (DHT) basierend auf dem Kademlia-Algorithmus. In dieser Netzwerkdatenbank sind einerseits die Router-Infos abgelegt, andererseits sind hier auch die LeaseSets zu finden. Nur sogenannte Floodfill-Router beantworten Anfragen an die Netzwerkdatenbank.

(Timpanaro et al., 2011, S. 5-6; „The Network Database - I2P“, n. d.)

## Router-Info

Jeder I2P-Router erstellt ein Schlüsselpaar und seine eigene RouterInfo-Datei beim Start. Zusätzlich werden die öffentlichen IP-Adressen ermittelt sowie zufällige Ports ausgewählt, über welche der I2P-Router erreicht werden kann. Eine RouterInfo-Datei beinhaltet den öffentlichen Schlüssel des Schlüsselpaars und die Netzwerkkoordinaten des I2P-Routers. Der private Schlüssel wird separat abgelegt und wird anschliessend zum Entschlüsseln von Nachrichten verwendet. Diese Informationen werden benötigt, damit ein anderer I2P-Router die Kommunikation mit demjenigen I2P-Router aufnehmen kann, der die RouterInfo-Datei erstellt hat. Jedoch muss der I2P-Router erst unter Verwendung der NetDb seine RouterInfo-Datei mit anderen Peers austauschen. Anhand der Netzwerkkoordinaten (IPv4 und IPv6 Adressen, sowie Ports) ist bekannt, wohin die Exploratory-Tunnels eine Anfrage zur Tunnelerstellung senden müssen. Mit Hilfe des öffentlichen Schlüssels kann die Anfrage zur Tunnelerstellung auch bereits verschlüsselt werden. 2.2.6 "Peer-Profiling, Selektion und Tunnel-Erstellung"

Es ist zu erwähnen, dass die öffentlichen IP-Adressen eines I2P-Knotens somit nicht geheim sind. Jeder der einen Knoten betreibt, kann einfach nur die NetDb abfragen und so die öffentlichen IP-Adressen von I2P-Knoten ausfindig machen. I2P wurde nicht mit der Absicht entwickelt, den Fakt zu verschleiern, dass I2P eingesetzt wird, sondern wurde designed, damit Teilnehmer Nachrichten innerhalb des Netzwerks austauschen können. Ein Beobachter kann also herausfinden, ob ein Rechner am I2P-Netzwerk beteiligt ist, kann jedoch nicht aussagen, ob eine Nachricht für den Knoten bestimmt ist, ob er eine Nachricht versendet, oder ob er diese lediglich weiterleitet.

## LeaseSet

Ein LeaseSet ist eine Sammlung von Tunnel-Eingangspunkten (auch Leases genannt) für eine bestimmte Destination. Wenn nun ein I2P-Router auf einen bestimmten Service oder eine Destination zugreifen will, weiss dieser nach dem Abfragen der NetDb, anhand eines LeaseSets, wo die richtigen Tunnel-Eingangspunkte für den entsprechenden Service im Netzwerk aufzufinden sind. LeaseSets werden also benötigt um ein Inbound- und Outbound-Tunnel miteinander zu verbinden. (Astolfi et al., 2015)

### 2.2.5. Floodfill-Router

Damit I2P-Router Abfragen an die verteilte NetDb machen können, muss es einige sogenannte Floodfill-Router im Netzwerk geben. Es handelt sich hier um Knoten mit hoher Kapazität und Performanz. Diese beantworten die Anfragen für LeaseSets und Router-Infos von anderen Routern. Ein LeaseSet wird beispielsweise abgefragt, wenn ein anderer Router auf eine Destination zugreifen will. Bei einem Floodfill-Router handelt es sich normalerweise um einen Router mit hoher Kapazität und Performanz, damit die Anfragen an die NetDb performant sind. Die Anzahl an Floodfill-Router im I2P Netzwerk sollte sich automatisch regulieren aufgrund des Peer-Profilings (siehe Abschnitt 2.2.6 "Peer-Profiling, Selektion und Tunnel-Erstellung"). Denn Router, die genug Bandbreite zur Verfügung haben, sollten sich automatisch zum Floodfill-Router umschalten, sofern nicht genügend (etwa 6 Prozent) Floodfill-Router im Netzwerk vorhanden sind. Dies konnte jedoch nicht bei der i2pd-Implementierung beobachtet werden. Ein Router kann auch vom Administrator manuell als Floodfill-Router konfiguriert werden. (Timpanaro et al., 2011, S. 5-6)

### 2.2.6. Peer-Profiling, Selektion und Tunnel-Erstellung

Die einfachste Möglichkeit, Nachbarknoten (oder Peers) zur Tunnelerstellung zu bestimmen, wäre zufällig Knoten auszuwählen. Jedoch wird dies aufgrund der Performanz anders gehandhabt. Jeder I2P-Knoten testet seine Nachbarknoten unter Verwendung seiner Exploratory-Tunnels anhand von drei Kriterien: Performanz, Latenzzeiten und Verfügbarkeit. Sie werden dann wiederum in drei Kategorien eingeordnet: Hohe Kapazität, Schnell und Standard. Dabei werden alle Knoten, die eine höhere Kapazität als der Median der Knoten im Netzwerk haben, unter "Hohe Kapazität" eingeordnet. Alle Knoten mit hoher Kapazität, deren Geschwindigkeit höher ist als der Median im Netzwerk, werden als "Schnell" kategorisiert. Der Rest der Knoten werden der "Standard"-Kategorie zugeordnet. Hochperformante Knoten werden bevorzugt beim Erstellen von Tunnels und es ist wahrscheinlicher dass diese automatisch als Floodfill-Router agieren. Natürlich hat dies auch einen Einfluss auf den Anonymitätsgrad. (Timpanaro et al., 2011, S. 4-5; Liu et al., 2014, S. 3)

### 2.2.7. Garlic-Routing

Inspiziert vom Onion-Routing des TOR-Netzwerks gibt es im I2P-Netzwerk der Begriff des Garlic-Routings. Ähnlich wie beim Onion-Routing wird eine Nachricht mehrfach verschlüsselt, und jeder Knoten entfernt eine Schicht. Im Gegensatz zu den TOR-Circuits, wo Nachrichten bidirektional ausgetauscht werden können, gibt es bei I2P nur unidirektionale Tunnels. Nachrichten werden übermittelt, indem ein Outbound-Tunnel mit einem Inbound-Tunnel verbunden wird. Dies handhaben die sogenannten Gateway-Knoten, die sich jeweils am Ausgang jedes Outbound-Tunnels und am Eingang jedes Inbound-Tunnels befinden. Um Nachrichten beantworten zu können, gibt es deshalb einfach zwei Tunnels ein Inbound- und ein Outbound-Tunnel. Diese Tunnels nehmen im Gegensatz zu einem TOR-Circuit aber nicht unbedingt den gleichen Weg, sondern können über andere Knoten übermittelt werden.

Die folgende Abbildung 2.2 "Garlic-Routing" veranschaulicht den Vorgang mit vier Benutzern:

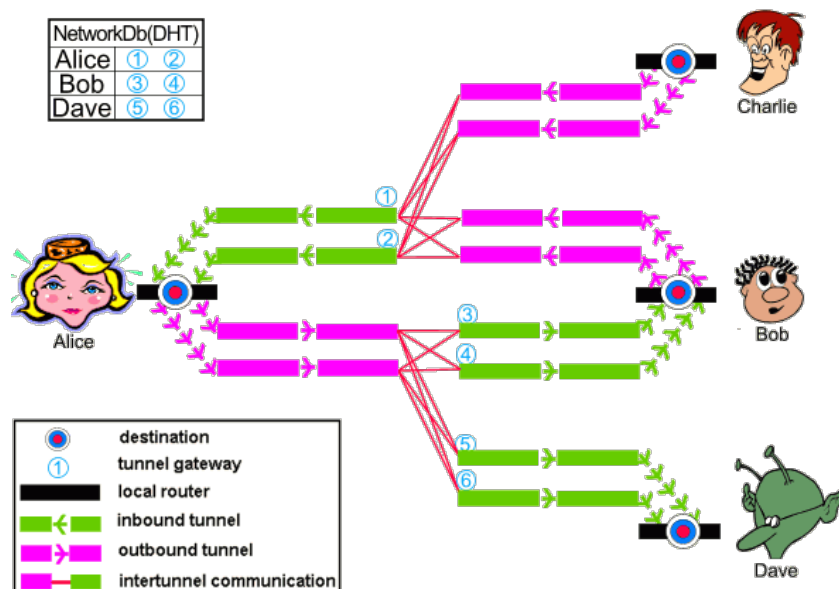


Abbildung 2.2.: Garlic-Routing



Beim Garlic-Routing wird nun eine Garlic-Nachricht wie oben gezeigt durch das Netzwerk geroutet. Eine Garlic-Nachricht kann mehrere sogenannte "Cloves" enthalten. Cloves bestehen aus einer Nachricht und zusätzlichen Routing-Informationen, wie zum Beispiel auch Verzögerungen. Das heisst eine Garlic-Nachricht kann mehrere Nachrichten von Applikationen enthalten. Der komplette Vorgang zum Versand einer Garlic-Nachricht läuft dabei wie folgt ab:

1. Die eignen Inbound- und Outbound-Tunnels werden erstellt.
2. Eine Anfrage an die NetDb wird abgesetzt um herauszufinden wo die Inbound-Tunnel für die gewünschte Destination zu finden sind.
3. Das eigene Outbound-Tunnel wird nun verwendet um die Garlic-Nachricht zu versenden.
4. Die Nachricht wird durch das Outbound-Tunnel geroutet.
5. Am Ende des Outbound-Tunnels befindet sich ein Gateway. Dieser leitet die Nachricht wiederum an einen Gateway weiter, nämlich an denjenigen Gateway für das Inbound-Tunnel der gewünschten Destination.
6. Der Gateway des Inbound-Tunnels leitet die Nachricht durch das Inbound-Tunnel zur gewünschten Destination.
7. Am Ausgang des Inbound-Tunnels kommt die Nachricht nun beim Empfänger oder der Destination an.

(Conrad & Shirazi, 2014, S. 4; Ehlert, 2011, S. 2; „Intro - I2P“, n. d.)

### **2.2.8. I2P-Testnetzwerke und Bootstrapping**

Eine Recherche hat ergeben, dass bereits in der Vergangenheit ein I2P-Testnetzwerk erstellt wurde („How to run 128 (testnet) I2P routers in multiple subnets on a single Linux system.“ 2018). Bei diesem Ansatz wurde nicht direkt eine komplette Container-Lösung verwendet, sondern direkt auf die Linux-Kernel-Funktionalitäten von Cgroups und Namespaces zurückgegriffen. Ein wichtiger Unterschied zwischen einem privaten Testnetzwerk und dem öffentlichen I2P-Netzwerk besteht darin, dass sich der Bootstrapping-Prozess erschwert. Bei einem Peer-to-Peer-Netzwerk wird ein Bootstrapping-Prozess benötigt, weil ein Knoten beim Aufstarten keine Netzwerkkoordinaten eines anderen Knotens kennt. Im öffentlichen Netzwerk wird dieses Problem einfach gelöst indem ein sogenannter Reseed-Server angefragt wird. Dieser liefert in einer sogenannten SU3-Datei eine Liste von RouterInfos zurück von Knoten denen die I2P-Entwickler vertrauen. Somit kann der neue Knoten dem Netzwerk beitreten und Abfragen an der NetDb tätigen. In einem privaten Testnetzwerk muss also auch ein Reseed-Server zur Verfügung gestellt werden, damit sich zu Beginn die Knoten gegenseitig überhaupt finden können.

### **2.2.9. Funktionsweise der Blockchain von DIVA.EXCHANGE**

Der Verein DIVA.EXCHANGE entwickelt seine eigene Blockchain namens Divachain. Diese basiert auf dem byzantinischen Konsensusalgorithmus und somit handelt es sich hier eine Proof-of-Stake basierende Blockchain. Kurz zusammengefasst stimmen beim byzantinischen Konsensusalgorithmus die einzelnen Knoten über neue Blöcke, die zur Blockchain hinzugefügt werden sollen, ab.

Stimmen mehr als  $\frac{2}{3}$  der Knoten diesem neuen Block zu, so wird dieser im Netzwerk angenommen. Damit dies funktioniert, müssen die Divachain-Knoten die neuen Blöcke auch untereinander austauschen. Um die Privatsphäre der Benutzer zu schützen, wird I2P als Netzwerkschicht verwendet. Dabei gilt es zu beachten, dass jeder Divachain-Knoten somit auch ein I2P-Knoten ist, aber viele andere I2P-Knoten nicht die Divachain-Software betreiben.

### 3. Ideen und Konzepte

Dieses Kapitel zeigt Ideen auf, nach denen die Fragestellung angegangen wurde. Es wird ein Konzept für einen Teststand vorgestellt, der den Anforderungen (siehe Abschnitt 4.2.3 “Projektanforderungen”) gerecht wird.

Grundsätzlich soll ein Teststand aufgebaut werden, an dem empirische Performance-Messungen an einem I2P-Testnetzwerk durchgeführt werden können (siehe Anforderung 2-TINF). Dabei gilt es, verschiedene Herausforderungen und Aspekte zu beachten. In den folgenden Abschnitten werden verschiedene Lösungsansätze und Ideen für verschiedene Teilprobleme und Aspekte vorgestellt.

#### 3.1. Reproduzierbarkeit

Um Messungen auf einem solchen Testnetzwerk wiederholbar und so gut wie möglich reproduzierbar zu machen (siehe Anforderung 4-TREP), sind folgende zwei Ideen wichtig. Erstens soll das Testnetzwerk isoliert werden, um äussere Einflüsse durch das Netzwerk (auch das öffentliche I2P-Netzwerk) zu vermeiden (siehe auch Abschnitt 3.3). Zweitens soll die Testinfrastruktur als Programmcode beschrieben werden (auch bekannt als Infrastructure as Code (IaC)). Somit kann dasselbe Testnetzwerk anhand der versionierten Infrastrukturdefinition später erneut wiederaufgebaut werden. Mehr dazu im Abschnitt 3.2.

#### 3.2. Infrastruktur und Deployment

Wichtig für das Deployment der Infrastruktur für das Testnetzwerk ist, dass man schnell neue Messungen starten und neue Testnetzwerke erstellen kann (siehe Anforderung 11-TVRS). mit der Möglichkeit, verschiedene Konfigurationseinstellungen zu tätigen (siehe Anforderung 5-TCNF). Wie im vorherigen Abschnitt bereits erwähnt, soll das Deployment auch reproduzierbar sein (siehe Anforderung 4-TREP). Ausserdem soll es möglich sein, verschieden grosse Testnetzwerke zu erstellen (siehe Anforderung 6-TSCL).

Um Ressourcen zu schonen, und da sie einen kleineren Overhead und kürzere Starup-Time haben, sind Container virtuellen Maschinen vorzuziehen. Zudem lassen sich Container schneller neu zu erstellen und zu deployen. Container erlauben es auch schneller Tests durchzuführen (siehe Anforderung 13-TPER). sowie Tests mit mehr Knoten zu machen, da weniger Ressourcen für einen einzelnen Knoten benötigt werden, weil der Betriebssystemkernel geteilt wird. Dies ist der Fall, weil bei Containerlösungen (oder auch OS-Virtualisierung) im Gegensatz zu virtuellen Maschinen der Betriebssystemkernel zwischen den Instanzen geteilt wird.

#### 3.3. Isolierung des Testnetzwerks

Um äussere Einflüsse zu vermeiden und eine idealisierte Testumgebung zu erschaffen muss das Testnetzwerk isoliert werden. Das Diagramm 3.1 “I2P Testnetwork” zeigt den Aufbau des Test-

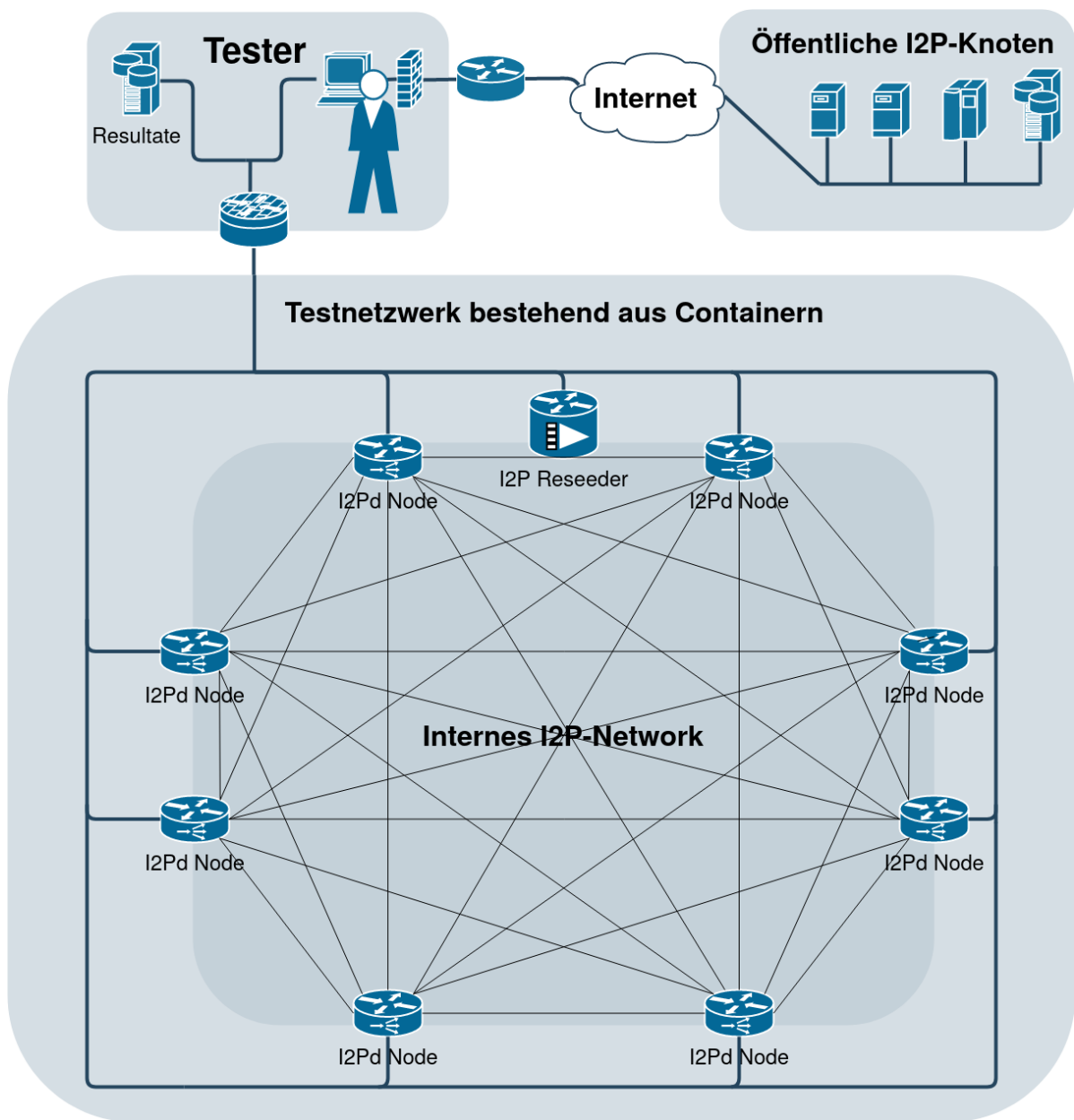


Abbildung 3.1.: I2P Testnetwork

netzwerks. Einerseits kann eine Isolation auf Ressourcenebene durch die verwendete virtuelle Maschine erreicht werden. Andererseits stellt dies auch automatisch eine Netzwerkisolation zur Verfügung, je nachdem, wie die Virtuelle Maschine mit dem Host verbunden wird. Das Testnetzwerk soll standardmässig weder mit dem Internet noch mit dem System des Testers verbunden sein, um äussere Einflüsse zu verringern und somit durch Isolation bessere Reproduzierbarkeit zu erreichen. Bei der Firewall handelt es sich entweder um das Hostsystem (VM-Network) oder um das Testnetzwerk selber, welche die I2P-Knoten hostet. Wichtig ist, dass der Tester, welcher das Netzwerk erstellt und die Tests durchführt, sich auch ausserhalb des Testnetzwerkes befindet, um ungewollte Einflüsse zu vermeiden.

Die I2P-Knoten sind als Router dargestellt und befinden sich eigentlich alle in zwei Netzwerken gleichzeitig: Im normalen IP Netzwerk und im I2P-Overlay-Netzwerk. Am Ende des Tests müssen die Testresultate an einem sicheren Ort ausserhalb des Testnetzwerks abgelegt werden, damit diese später analysiert werden können. In der i2pd-Konfiguration gibt es zusätzlich das `family`-Tag, welches die Knoten als zusammenhängend markieren würde, würden diese aus Versehen mit dem echten I2P-Netzwerk kommunizieren. Dies ist auch interessant bezüglich Tests die am echten Netzwerk durchgeführt werden sollen.

### 3.4. Bootstrapping

Wie im Abschnitt 2.2.8 "I2P-Testnetzwerke und Bootstrapping" erklärt wird im privaten Testnetzwerk ein Bootstapping-Vorgang benötigt. Ganz zu Beginn, wenn ein Knoten dem I2P Netzwerk beitrifft, muss dieser wissen, wo mindestens ein anderer Knoten ist. Um dies zu bewerkstelligen, wird ein separater Container dem Testnetzwerk hinzugefügt, der einen Reseed-Server mit sich bringt. Somit kann zu Beginn jeder Knoten eine Liste von anderen Knoten abfragen und so dem Netzwerk beitreten. Anhand dieser Liste kann auch der Verbindungsgrad der Knoten gesteuert werden. Man kann auch einen Floodfill-Router verwenden, um das Netzwerk zu bootstrappen („Bootstrapping without Reseed Servers - i2pd documentation“, n. d.). Jedoch scheint der Ansatz mit einem Reseed-Server eleganter zu sein, da sich somit nicht ein extra Knoten im Netzwerk befindet, der die Messung beeinflussen könnte.

### 3.5. Konfigurierbarkeit

Nach Absprache mit dem Auftraggeber soll mindestens Folgendes im Teststand konfigurierbar sein (5-TCNF):

- Anzahl-Knoten im Testnetzwerk
- Maximale Bandbreite der einzelnen Knoten
- Anzahl Hops (also die Länge der Tunnel)
- Ob im privaten oder öffentlichen Netzwerk getestet werden soll. (optional)

Nix als deklarative Konfigurationssprache erlaubt es, beliebige Aspekte des Setups konfigurierbar zu machen. Die meisten oben erwähnten Konfigurationsmöglichkeiten werden bereits durch verschiedene NixOS-Module (i2pd, Firewall, ... ) angeboten. Als anderer Ansatz kann eine JSON-Datei eingesetzt werden, da solche von vielen Programmiersprachen einfach eingelesen werden können.

### 3.6. Latenzmessung

Um die Latenz oder Verzögerung einer Nachricht im Testnetzwerk zu messen (siehe Anforderung 9-TLAT), könnte man jeweils den Empfangs- sowie Sendezeitpunkt einer Nachricht speichern und voneinander subtrahieren.

$$\boxed{\text{Latenz} = \text{Empfangszeitpunkt} - \text{Sendezeitpunkt}} \quad (3.1)$$

*Latenzberechnung*

Grundlage für eine saubere Latenzmessung ist zudem, dass man in keine Ressourcenengpässe gerät. Dementsprechend ist das Überwachen der benötigten Ressourcen wichtig; siehe dazu Abschnitt 3.8 “Messen der Ressourcenauslastung”.

### 3.7. Limitieren der Bandbreite

Die i2pd-Software erlaubt es, die Bandbreite eines Knotens (siehe Anforderung 10-TLIM) mittels der bandwidth-Konfigurationseinstellung (siehe Anforderung 5-TCNF) zu limitieren. Zusätzlich gäbe es auch andere Wege die Bandbreite zu limitieren. Eine Möglichkeit wäre es dies auf Netzwerkebene, mittels des “Traffic Control Tool” (kurz tc zu lösen.

### 3.8. Messen der Ressourcenauslastung

Die Messung der Ressourcenauslastung der Knoten kann aus mehreren Gründen sinnvoll sein: Gerät ein Knoten oder die Hostsysteme für das Testnetzwerk in Ressourcenengpässe, beeinflusst dies auch die Resultate. Dies ist insbesondere bei Latenzmessungen enorm wichtig, denn diese würden so fehlerhaft und nicht reproduzierbar (siehe Anforderung 4-TREP). Auch kann festgestellt werden, dass das Testnetzwerk nicht zu gross skaliert wurde, bzw. wie weit es skaliert werden kann (siehe Anforderung 6-TSCL). Zudem kann die Test-Performance so gemessen werden (siehe Anforderung 13-TPER). Folgendes könnte gemessen werden:

- CPU Gebrauch (in Prozent)
- Netzwerkbandbreite (kbits/sec)
- RAM Verwendung (in Prozent)
- Disk I/O (read/write, kbits/sec)
- Freier Speicherplatz

Es wird erwartet, dass der Workload von I2P vor allem netzwerklastig (weiterleiten über mehrere Knoten) und prozessorlastig (Verschlüsselung) ist. Da es sich hier um ein virtuelles Container-Netzwerk handelt, heisst das, dass die Last auf dem Netzwerk sich auch in der Prozessorlast zeigen wird. Um die benötigte Anzahl Container zu starten und die I2P-Knoten zu betreiben, wird auch viel Arbeitsspeicher benötigt.

## 4. Methode

Als wissenschaftliche Methode wurde ein Experiment durchgeführt. Als Vorgehensmodell für das Projektmanagement wurde KanBan ausgewählt.

### 4.1. Experiment

Es soll ein Experiment durchgeführt werden, um eine quantitative Analyse durchführen zu können. Spezifischer soll ein Laborexperiment durchgeführt werden, um die Hypothese 1 zu testen. Dabei gilt es, die drei Qualitätskriterien Messbarkeit, Wiederholbarkeit und Überprüfbarkeit zu beachten. Damit die Performance-Experimente an I2P durchgeführt werden können, muss ein Testnetzwerk aufgebaut werden sowie eine Messanlage entwickelt werden. (Helmut Balzert et al., 2017, S. 276)

### 4.2. Projektmanagement

In diesem Abschnitt wird aufgezeigt welches, Vorgehensmodell verwendet wurde und wie das Projekt abgewickelt wurde. Auch wird aufgezeigt, wie das Projekt organisiert ist, wie die Anforderungen ermittelt wurden und welche Anforderungen es gibt.

#### 4.2.1. Vorgehensmodell

In den Besprechungen hat sich ergeben, dass inkrementell und iterativ gearbeitet wird, so wie auch von der Aufgabenstellung vorgeschlagen (siehe Anhang A). Das Projektmanagement soll schlank gehalten werden und nicht viel Aufwand erzeugen, da alleine an dem Bericht gearbeitet wird. Deshalb wurde Kanban als Vorgehensmodell ausgewählt. Einige wichtige Kernpraktiken von Kanban sind hier kurz erklärt:

1. **Arbeit sichtbar machen:** Arbeit muss sichtbar gemacht werden, um zu sehen wo der Arbeitsfluss im Gesamtsystem stockt. Auch arbeitet man bei Kanban nicht nach dem Push-Prinzip sondern dem Pull-Prinzip. Das heisst es wird keine Arbeit zugewiesen, sondern neue Arbeit wird abgeholt, sobald die letzte Aufgabe (oder Issue/Task) fertiggestellt wurde.
2. **Limitiere den Work in Progress (WIP):** Es ist schwierig ständig zwischen verschiedenen Aufgaben zu wechseln. Deshalb soll nur an einer Aufgabe auf einmal gearbeitet werden. Tritt bei einer Aufgabe ein Problem auf, legt man sie auf die Seite, um sie später mit jemandem anzuschauen, oder erledigt erst eine andere Aufgabe, welche eine Voraussetzung dafür ist.
3. **Manage Flow:** Arbeitsfluss ist wichtig in Kanban. Die Aufmerksamkeit soll auf Engpässe oder Blockaden im System gelegt werden. Dabei hilft es, zu unterscheiden, was für verschiedene Arbeitstypen das Team erledigen muss, und diesen verschiedene Dringlichkeitsgrade zuzuweisen.

4. **Explizite Prozessregeln:** Die Regeln sollen offengelegt und explizit definiert werden, damit man merkt, wann diese gebrochen werden oder nicht mehr sinnvoll sind.
5. **Feedback Mechanismen:** Um die Arbeitsprozesse kontinuierlich verbessern zu können, benötigt es Feedback-Mechanismen. Tägliche Stand-Ups oder Retrospektiven sind z.B. mögliche Gelegenheiten für Feedback. Dies sind Meetings, in denen darauf abgezielt wird dazuzulernen, den Arbeitsprozess zu reflektieren und sich zu verbessern.

(Siegfried Kaltenecker und Klaus Leopold, 2013, p. 17-22)

#### 4.2.2. Projektorganisation

##### Projektteam

Die folgende Tabelle 4.1 listet alle Personen auf die an diesem Projekt beteiligt sind.

Person	Rollen
Carolyn Bächler-Schenk	Auftraggeber
Konrad Bächler	Auftraggeber
Arnold Dieter	Betreuungsperson
Urs Rufer	Experte
Moritz Küttel	Student

Tabelle 4.1.: Im Projekt involvierte Personen

##### Quellcode

Der  $\text{\LaTeX}$ -Quellcode für diesen Bericht ist auf codeberg.org in diesem Repository zu finden:  
<https://codeberg.org/mkuettel/ba>

##### Projektboard und Issue-Tracker

Die Abbildung 4.1 zeigt das Projektboard, welches für Projektmanagement und Controlling verwendet wird. Jede Karte auf dem Projektboard ist eine Aufgabe oder auch Issue aus dem folgenden Issue-Tracker: <https://codeberg.org/mkuettel/ba/issues>

Der Issue-Tracker ist zugleich ein Backlog. Issues können nach Bedarf aus dem Issue-Tracker entnommen werden und zum Projektboard hinzugefügt werden. Das Projektboard ist ein Kanban-Board und enthält drei Spalten: "To Do", "In Progress" und "Done" (siehe Abbildung 4.1). Die Issues aus dem Issue-Tracker können dann je nach Fortschritt in die Spalten einsortiert werden. Man sollte also nie mehr als einen Issue in der Spalte "In Progress" haben. Das Kanban-Board ist hier zu finden: <https://codeberg.org/mkuettel/BA/projects/125>

Ein einziger Issue sollte nie mehr Aufwand machen als acht Stunden Arbeit. Diese Regel hilft die Issues kleiner zu halten und genauere Aussagen treffen zu können, wo man im Projekt steht. Wird an einem Issue gearbeitet wird dies im Arbeitsjournal vermerkt (siehe Anhang sec:journal).



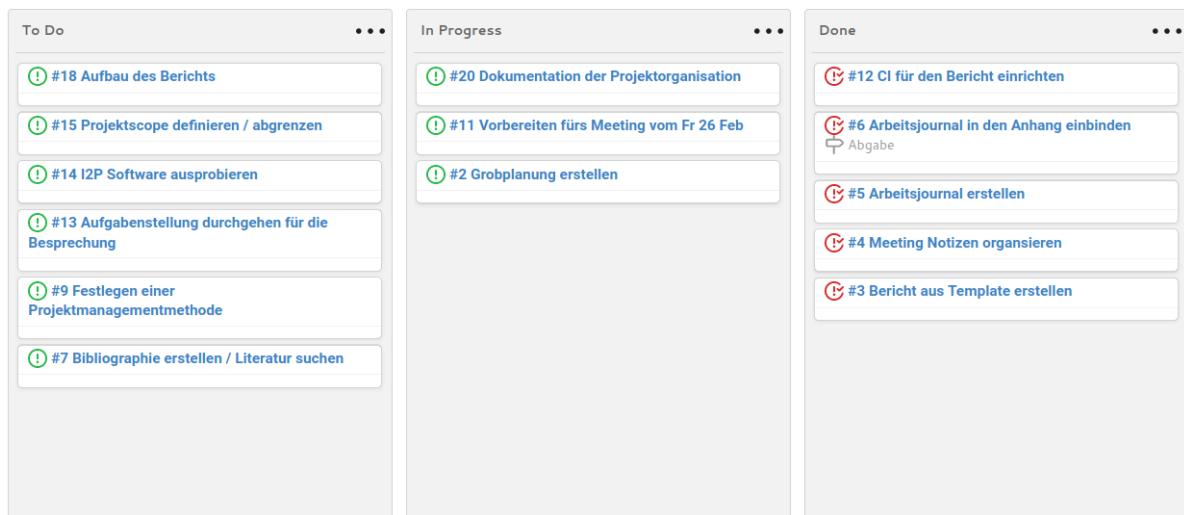


Abbildung 4.1.: CodeBerg Project Board

### 4.2.3. Projektanforderungen

Die Anforderungen an das Projekt haben sich aus der Aufgabenstellung (siehe Anhang A) sowie aus dem Requirements-Engineering durch regelmässige Meetings und Absprachen mit dem Auftraggeber ergeben. Die Vorgaben der Hochschule Luzern für Bachelorarbeiten wurden ebenfalls als Anforderungen aufgenommen.

Die folgende Tabelle 4.2 zeigt alle ermittelten Anforderungen auf. Die erste Spalte "Id" gibt jeder Anforderung einen eindeutigen Kennzeichner, um diese einfacher zu referenzieren. Die Anforderung selber ist in der Spalte "Anforderung" genauer beschrieben. Die Spalte "Art" gibt an, ob es sich bei der Anforderung um eine funktionale oder nicht-funktionale Anforderung handelt. Nicht jede Anforderung muss zwingend erfüllt werden, deshalb kennzeichnet die Spalte "Optional", ob die jeweilige Anforderung zwingend erfüllt werden muss.

Id	Anforderung	Art	Optional
1-SDTF	Der Stand der Technik/Forschung bezüglich I2P muss ermittelt und in den Bericht eingebunden werden.	Nicht-funktional	Nein
2-TINF	Ein Teststand muss aufgebaut werden, welches es erlaubt verschiedene Performance-Messungen an einem Netzwerk bestehend aus mehreren i2pd-Knoten auszuführen.	Funktional	Nein
3-TKON	Es soll ein Konzept erstellt werden für den Teststand. Dieser soll aufzeigen wie dieser aussehen soll, wie er funktioniert und was genau gemessen werden soll.	Nicht-funktional	Nein

<b>Id</b>	<b>Anforderung</b>	<b>Art</b>	<b>Optional</b>
4-TREP	Die am Teststand durchgeführten Messungen sollten so gut wie möglich reproduzierbar sein. Die gleiche Messung sollte soweit wie möglich dieselben Resultate liefern.	Nicht-funktional	Nein
5-TCNF	Gewisse Aspekte des Teststands und der i2pd-Knoten müssen konfigurierbar sein, damit verschiedene Messungen durchgeführt werden können.	Funktional	Nein
6-TSCL	Der Teststand soll zwischen 8 bis maximal 256 i2pd-Knoten unterstützen. Dies muss konfigurierbar sein, um das I2P-Testnetzwerk für verschiedene Tests zu skalieren zu können.	Funktional	Nein
7-TISO	Das I2P-Testnetzwerk soll isoliert sein vom realen I2P-Netzwerk.	Nicht-funktional	Nein
8-TNET	Das I2P-Testnetzwerk soll als Knotenfamilie auch dem öffentlichen I2P-Netzwerk beitreten können.	Funktional	Ja
9-TLAT	Die Latenz von Nachrichten die über das I2P-Testnetzwerk gesendet werden, soll gemessen werden.	Funktional	Nein
10-TLIM	Es muss möglich sein im Teststand die verfügbare Bandbreite von einzelnen i2pd-Knoten einzustellen.	Funktional	Nein
11-TVRS	Man soll schnell (innerhalb von 2h) ein neues Experiment mit neuen Einstellungen starten können.	Nicht-funktional	Nein
12-EVAL	Die am Teststand durchgeführten Messungen müssen qualitativ evaluiert und aufgearbeitet werden.	Nicht-funktional	Nein
13-TPER	Man geht davon aus, dass die Messungen lange dauern können, da jeweils ein ganzes Netzwerk und dessen Traffic erstellt und gehandhabt werden muss. Es ergibt Sinn, die Ausführungszeiten von Messungen kurz zu halten, damit mehr Messungen durchgeführt werden können.	Nicht-funktional	Ja
14-DOCS	Am Ende des Projekts muss diese Bachelorarbeit abgegeben werden. Der Bericht soll das Projekt, die verwendeten Methoden, Evaluationsprozesse, Umsetzung und Resultate beschreiben.	Nicht-funktional	Nein
15-ITER	Ein iteratives Projektmanagement wird bevorzugt für kurze Feedback-Zyklen.	Nicht-funktional	Ja
16-PRES	Es muss eine Zwischenpräsentation vorgetragen werden, welche das Projekt und das weitere Vorgehen erklärt.	Nicht-funktional	Nein

<b>Id</b>	<b>Anforderung</b>	<b>Art</b>	<b>Optional</b>
17-WEBA	Es muss ein Web-Abstract erstellt werden, der das Projekt kurz zusammenfasst. Der Web-Abstract wird veröffentlicht und muss eine Woche vor Projektende vom Betreuer abgenommen werden.	Nicht-funktional	Nein
18-PVID	Es muss ein 90 Sekunden Video erstellt werden, welches diese Bachelorarbeit kurz erklärt. Das Video wird am Start der Schlusspräsentation abgespielt.	Nicht-funktional	Nein

Tabelle 4.2.: Anforderungen

#### 4.2.4. Planung

Für diese Bachelorarbeit werden 360 Stunden Arbeit während den 15. Projektwochen geleistet. Im Schnitt bedeutet dies acht Stunden Arbeit an jeweils drei Wochentagen (meistens jeweils Mittwoch-Freitag). Neben den definierten Meilensteinen, Projektphasen und dem Projektkalender gibt es auch eine Liste an Resultaten im Anhang B, welche eine Zeitschätzung für jedes Projektergebnis beinhaltet.

##### Meilensteine

Die Meilensteine sind im Projektkalender (4.3 "Projektkalender") hellblau gekennzeichnet und hier aufgelistet.

1. **Startdatum:** Mittwoch, 24. Februar 2021, Kick-Off-Meeting
2. **Zwischenpräsentation:** 21. April 2021, 14:00
3. **Abgabe Web-Abstract:** 28. Mai 2021
4. **Abgabedatum:** Ursprünglich 4. Juni 2021, verschoben auf 20. September, 12:00
5. **Schlusspräsentation:** Ursprünglich 26. Juni 2021, 14:00, verschoben auf 29. September, 16:00

Die Meilensteine sollen ein Controlling-Mechanismus sein, um feststellen zu können, wenn das Projekt im Verzug ist. Diese Meilensteine wurden auch in den Issue-Tracker aufgenommen:

<https://codeberg.org/mkuettel/BA/milestones>

##### Phasen

Das Projekt wird in vier Phasen durchgeführt, welche in Tabelle 4.3 "Projektkalender" dargestellt sind. Dabei sind die Phasen jeweils durch Meilensteine voneinander abgegrenzt.

1. **Konzeptionsphase:** Die Konzeptionsphase beginnt mit dem Kick-Off Meeting. In dieser Phase wird das Projekt geplant, recherchiert, der Bericht wird strukturiert, die Anforderungen werden eruiert und ein Konzept für einen Teststand wird erstellt.

2. **Aufbau des Teststands:** Ein Teststand aufbauen nach Konzept um Performance-Experimente und Messungen mit der i2pd-Software in einem Netzwerk machen zu können.
3. **Auswertungsphase:** Verschiedene Performance-Experimente und Messungen werden in dieser Phase ausgeführt und ausgewertet.
4. **Abschlussphase:** Vervollständigen des Berichts, Fazit erstellen, Korrekturlesen, Erstellung des Web-Abstracts.

### Projektkalender

Die folgende Tabelle 4.3 "Projektkalender" war enorm hilfreich um die Übersicht während des Projekts zu behalten. Die Spalte "PW" gibt die Projektwoche an, die Spalte "KW" die Kalenderwoche.

PW	KW	Phase	Meilensteine	Wochentag								
				Mo	Di	Mi	Do	Fr	Sa	So		
1	8	Konzeption	Projektstart	22	23	24	25	26	27	28		
2	9				März							
3	10				1	2	3	4	5	6	7	
4	11		1. Konzept Teststand	8	9	10	11	12	13	14		
5	12	Aufbau des Teststands		15	16	17	18	19	20	21		
6	13			22	23	24	25	26	27	28		
				29	30	31						
6	13					April						
7	14						1	2	3	4		
8	15		2. Zwischenpräsentation	5	6	7	8	9	10	11		
9	16	Auswertung		12	13	14	15	16	17	18		
10	17			19	20	21	22	23	24	25		
				26	27	28	29	30				
10	17					Mai						
11	18		3. Auswertung	3	4	5	6	7	8	9		
12	19		Abschluss									
13	20	4. Web-Abstract		10	11	12	13	14	15	16		
14	21			17	18	19	20	21	22	23		
15	22			24	25	26	27	28	29	30		
				31								
15	22	5. Abgabe			1	2	3	4	5	6		

Tabelle 4.3.: Projektkalender

Es gilt zu beachten das dies eine Planung ist. Der effektive Projektverlauf ist im Issue-Tracker und im Arbeitsjournal im Anhang C ersichtlich. Aus gesundheitlichen Gründen wurden die letzten beiden Projektwochen in den September verschoben.

## 5. Realisierung

Dieses Kapitel beschreibt wie die definierten Ziele umgesetzt wurden. Zuerst wird die Systemarchitektur für die entwickelte Test-Infrastruktur und Messanlage beschrieben. Anschliessend, im darauf folgenden Abschnitt, wird genauer auf das Design der einzelnen Komponenten eingegangen. Schlussendlich wird beschrieben wie dies umgesetzt wurde, welche Lösungswege eingeschlagen wurden, und wo Schwierigkeiten aufgetreten sind.

### 5.1. Systemarchitektur

Im Kapitel 3 “Ideen und Konzepte” wurde konzeptionell ausgeführt, wie der Teststand und die Messanlage für das I2P-Testnetzwerk aussehen soll. In diesem Abschnitt wird nun die resultierende Softwarearchitektur aufgezeigt und beschrieben. Die Abbildung 5.1 “Architektur-Diagramm” zeigt in einem Komponenten-Diagramm die Systemarchitektur auf. Die entwickelte Software besteht grob aus fünf Teilen:

1. **Deployment:** Hier ist die Deployment-Konfiguration für die Test-VM zu finden. Mittels NixOps kann dieselbe Test-VM reproduzierbar erstellt werden.
2. **Test-VM:** Die Test-VM stellt eine reproduzierbare und isolierte Umgebung zur Verfügung, um darin die Messungen durchführen zu können. Die Virtuelle Maschine grenzt die Ressourcen des Testnetzwerks klar ein. Es handelt sich hier um ein Linux-System, das Docker installiert hat, um darin die benötigten Container zu erstellen.
3. **Container-Deployment:** Dieser Bestandteil ist dafür verantwortlich, anhand der Testkonfiguration das I2P-Testnetzwerk, inklusive der Shared-Volumes, mit Hilfe von Docker zu erstellen. Dazu wird das Recompose-Skript verwendet. Hierbei handelt es sich um einen Wrapper, der intern Docker-compose verwendet.
4. **I2P-Testnetzwerk:** Das Testnetzwerk besteht aus i2pd-Containern sowie aus dem Reseed-Container. Dieser wird benötigt, um das Testnetzwerk zu bootstrappen.
5. **Messanlage:** Die Messanlage besteht aus einem TCP-Server und einen TCP-Client, die zusätzlich in den i2pd-Containern angesiedelt sind. Zudem gibt es ein Messskript, welches den Nachrichtenversand vom TCP-Client zum TCP-Server anstossen kann und die Messresultate sammelt.

### 5.2. Komponentendesign

In diesem Abschnitt werden die einzelnen Komponenten, wie dargestellt der Abbildung 5.1 “Architektur-Diagramm”, genauer beschrieben.

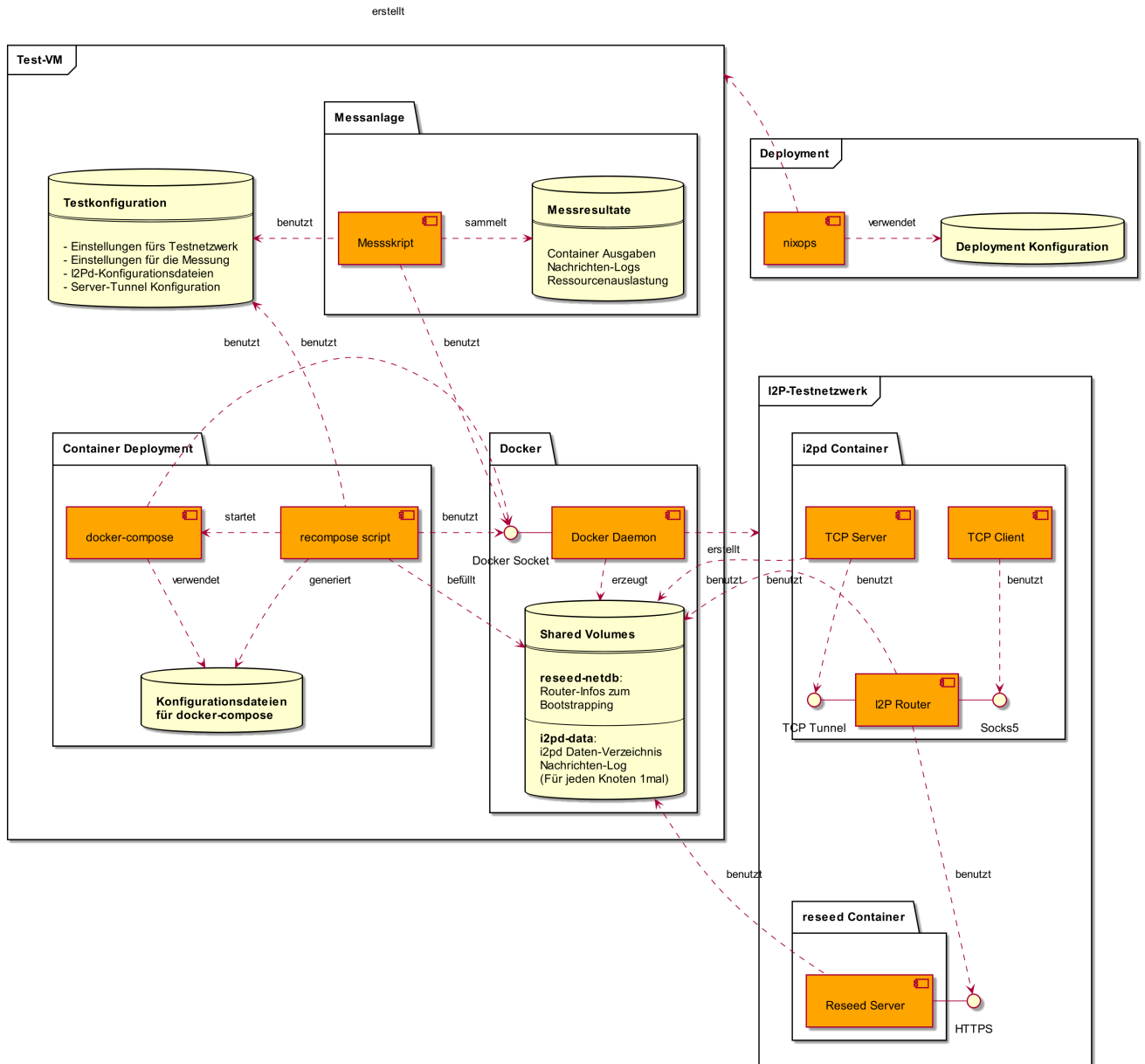


Abbildung 5.1.: Architektur-Diagramm

### 5.2.1. Deployment der Test-VM

NixOps erlaubt es, mittels einer NixOS-Systemkonfiguration ein komplettes Linux auf diverse Arten von Maschinen zu deployen. Dies kann physische Hardware oder, wie auch in diesem Falle, eine VM sein. Somit sind alle benötigten Abhängigkeiten in Code-Form abgelegt und eine Test-VM kann reproduzierbar erstellt werden.

### 5.2.2. Testkonfiguration

Die Konfiguration des Teststands kann mittels einer JSON-Konfigurationsdatei einfach angepasst werden. Da es sich hier um eine JSON-Datei handelt, kann diese falls nötig einfach von verschiedenen Programmen gelesen werden. Einerseits wird diese vom Messkript gelesen, wie aber auch vom recompose Skript. Wie diese Konfiguration genau aussieht, wird im Abschnitt 5.5.3 "Konfigurationsoptionen" genauer erläutert.

### 5.2.3. Recompose-Skript

Das Recompose-Skript ist dafür verantwortlich das Docker-Netzwerk anhand der Konfiguration aufzubauen und zu verwalten. Es ist das Kernstück der entwickelten Software. Es handelt sich hierbei um einen Wrapper um Docker-compose. Einerseits sorgt der Wrapper dafür, dass das Containernetzwerk korrekt aufgebaut wird (siehe Abschnitt 5.5.4 "Netzwerkisolation") . Andererseits generiert dieses die von Konfigurationsdateien für Docker-compose und managed die Shared-Volumes. Docker-Compose erlaubt das automatisierte Erstellen und Abbauen von Netzwerken bestehend aus mehreren Docker-Containern. Dazu liest Docker-compose jeweils eine oder mehrere YAML-Dateien. In diesen YAML-Dateien können die verschiedenen Container, Netzwerke und Volumes und diverse Optionen deklariert werden, welche das zu erstellende Containernetzwerk beschreiben. Im Benutzerhandbuch im Abschnitt 5.7.3 "Erstellen des Testnetzwerkes" wird kurz gezeigt, wie dieses Skript verwendet wird.

### 5.2.4. i2pd-Container

Hauptsächlich wird in diesem Container der i2pd-Prozess ausgeführt. Der TCP-Client und TCP-Server, welche zur Messung benötigt werden, sind ebenfalls direkt in diesem Container angesiedelt. Diese sind in den Abschnitten 5.5.6 "TCP-Server" und 5.5.7 "TCP-Client" genauer beschrieben. Dies verletzt zwar das Prinzip, dass man nur einen Prozess in einem Docker-Container ausführen sollte, jedoch ist das Ziel, hiermit Messungenauigkeiten aufgrund eines zusätzlichen Containers zu vermeiden, sowie die Netzwerkkonfiguration zu vereinfachen.

### 5.2.5. Reseed-Server

Da das Testnetzwerk privat ist und somit nicht Teil des öffentlichen Netzwerks sein sollte (siehe Anforderung 7-TISO), wird ein Bootstrapping Prozess benötigt, da die öffentlichen Reseed-Server nicht erreichbar sind. Dieser wurde so implementiert, das ein zusätzlicher Container im Testnetzwerk gestartet wird, welcher einen sogenannten Reseed-Server beinhaltet. Die i2pd-Container fragen zu Beginn den Reseed-Server über HTTPS nach einer Liste von RouterInfos. Dieser Ansatz hat den Vorteil, das so das Testnetzwerk schnell komplett neu aufgebaut und gestartet werden kann. Weitere Details zum Bootstrapping-Vorgang sind im Abschnitt 5.5.2 "Bootstrapping und Reseed-Server" zu finden.

### 5.2.6. Messskript

Das Messskript ist dafür verantwortlich, eine konfigurierte Anzahl an Nachrichten zur Latenzmessung durch das Testnetzwerk zu senden, damit Messdaten erstellt werden. Die Nachrichten zur Latenzmessung werden vom TCP-Client eines zufälligen Knotens zum TCP-Server eines anderen zufälligen Knotens verschickt. Somit kann sichergestellt werden, dass die Latenzmessungen repräsentativ sind für das gesamte Netzwerk. Um das Netzwerk nicht zu überlasten, werden die Nachrichten sequentiell versendet und es wird nur eine Nachricht auf einmal versendet. Zudem wird nach jeder versendeter Nachricht einige Sekunden (je nachdem was konfiguriert wurde) gewartet. Genaueres dazu im Abschnitt 5.5.9 "Das Messskript".

## 5.3. Umsetzung

In folgendem Abschnitt wird nun beschrieben, wie die im vorherigen Abschnitt aufgezeigte Systemarchitektur und das dazugehörige Komponentendesign effektiv umgesetzt wurde. Zudem wird der Weg zur Lösung genauer beschrieben. Zuerst wurde versucht ein I2P-Testnetzwerk unter Verwendung von NixOS-Containern zu erstellen (siehe Abschnitt 5.4 "Lösungsansatz: NixOS-VMs und NixOS-Container"). Jedoch wurde schlussendlich unter Verwendung von Docker-Compose eine andere Lösung gefunden, welche den Anforderungen gerecht wurde. Der erzeugte Quellcode zur Erstellung des I2P-Testnetzwerk's sowie der Quellcode für die Messanlage ist im folgenden Repository zu finden:

<https://codeberg.org/mkuettel/i2p-testnet>

## 5.4. Lösungsansatz: NixOS-VMs und NixOS-Container

Zu Beginn des Projekts wurde versucht, mit Hilfe von NixOps ein I2P-Testnetzwerk aufzubauen. NixOps ist ein Tool zum Deployment von Netzwerken bestehend aus NixOS-Maschinen. NixOS ist eine Linux-Distribution basierend auf dem Paket-Manager Nix, der es erlaubt es reproduzierbare Softwarepakete zu erzeugen. Er kann mit einer deklarativen und funktionalen Programmiersprache konfiguriert werden<sup>1</sup>. Dieser Ansatz wurde zu Beginn aus mehreren Gründen ausgewählt. Einerseits setze ich auch selber NixOS seit mehreren Jahren ein und NixOps ist somit das dazugehörige Deployment-Tool. Andererseits legt Nix enormen Wert auf Reproduzierbarkeit. Das gesamte Netzwerk bis hin zum OS könnte anhand einer Nix-Konfiguration wiedererstellt werden. Auch könnten Konfigurationen für I2P-Knoten wiederverwendet werden, egal ob diese nun in Container, VMs oder auf echter Hardware deployed werden. Dies ist möglich da, System-Konfigurationen oder Teile davon bei NixOS unabhängig von der Konfiguration der effektiven Infrastruktur sind. Angefangen wurde damit, dass mit NixOps ein Netzwerk bestehend aus einer einzelnen VirtualBox-VM erstellt wurde. Anschliessend wurden mehrere I2P-Knoten als NixOS-Container innerhalb dieser VM zu erstellt und es wurde versucht eine Lösung für das Bootstrapping Problem zu finden. Jedoch ist man mit diesem Ansatz immer wieder an verschiedenen Stellen auf Probleme gestossen:

- **Netzwerkkonfiguration:** Die Knoten konnten nicht korrekt untereinander kommunizieren. Es war nicht möglich einen anderen Container mittels eines ping-Befehls zu erreichen. Es hat sich herausgestellt, dass das Routing zwischen den Containern sowie die Netzwerkmasken nicht korrekt konfiguriert wurden und es keine Option gab, diese Routen innerhalb

---

<sup>1</sup> Siehe die Webseite von NixOS für weiterführende Informationen: <https://nixos.org>



der Nix-Deklaration richtig zu setzen. Aufgrund dessen wurde das NixOS-Modul für NixOS-Container erst angepasst. Somit konnte man die Routing-Probleme beheben und Optionen hinzufügen, um die Netzwerkmasken der Knoten korrekt zu setzen. Das NixOS-Container Modul so erstelltheint also nicht ausgereift zu sein.

- **Docker-Kompatibilität:** Auch hat man versucht anstatt von NixOS-Containern, Docker-Container auf NixOS zu erstellen. Dies hätte es erlaubt, die bestehenden Container von DIVA.EXCHANGE wiederzuverwenden. Jedoch schien der Support für Docker auch nicht ausgereift genug zu sein. Man konnte auf meinem Laptop keine Docker-Images bauen für das Testnetzwerk, da dies libvirt/QEMU-Support benötigt. Dies war unter gleichzeitiger Verwendung von Virtual-Box nicht möglich, da nur ein Hypervisor gleichzeitig ausgeführt werden kann. Man hat auch versucht den Hypervisor auf libvirt/QEMU zu wechseln, was aber schon am Deployment der VM scheiterte.
- **Skalierungsprobleme:** Während dem Entwickeln hat man jeweils 2 oder 8 Knoten auf einmal erstellt, um schnell ein Feedback zu erhalten. Um den Reseeder zu testen hat man jedoch versucht 103 Knoten auf einmal zu erstellen. Jedoch hat nur schon das Generieren und Auswerten der NixOS-Systemkonfigurationen für alle diese NixOS-Container extrem viel Arbeitsspeicher benötigt. Meine 16 GB Arbeitsspeicher in meinem Laptop waren nach mehreren Sekunden gefüllt. Man konnte somit die Knoten nicht einmal erstellen und deployen, da nur schon das berechnen der NixOS-Systemkonfigurationen zu ineffizient implementiert ist.

Somit war klar das dieser Lösungsansatz nicht funktionieren konnte und es musste ein grundsätzlich anderer Ansatz zur Erstellung der Container gefunden werden.

## 5.5. Lösung mit Docker-Compose

Nachdem sich herausgestellt hat, dass der Lösungsansatz mit NixOS-Containern nicht funktioniert, wurde eine Container-Orchestration Lösung gesucht. Um unnötige Komplexität zu vermeiden wurde auf das einfachere Tool Docker-compose gesetzt. Zudem hat der Auftraggeber bereits Erfahrungen mit diesem Tool gemacht und es auch schon eingesetzt um i2pd-Netzwerke zu erstellen für die Entwicklung an der DIVA-Blockchain. Anhand der `docker-compose.yml`-Datei vom Auftraggeber, konnte ein kleines öffentliches I2P-Testnetzwerk mit acht festkodierte i2pd-Container werden. Dies wurde anschliessend als Ausgangspunkt verwendet.

### 5.5.1. Skalierung

Damit verschiedene Anzahl an I2P-Knoten im privaten Testnetzwerk getestet werden können, müssen diese skaliert werden können. Anfangs wurde versucht auf die Skalierungsfunktion von Docker-compose zurückzugreifen. Mittels der `--scale`-Option des `docker-compose up` Befehls, können aus einem einzigen deklarierten Container mehrere Container instanziiert werden. Man kann diesen also nur einmal definieren, wie aufgezeigt in folgendem Listing:

```
1 # ...
2 services:
3   i2pd:
4     build:
```

```

5     context: "./docker/i2p"
6     environment:
7         ENABLE_TUNNELS: 1
8     networks:
9         i2ptestnet:
10         ipv4_address: "10.23.128.1"
11 # ...

```

---

Jedoch bin ich mit diesem Ansatz auf Netzwerkprobleme gestossen, sodass die i2pd-Container nie untereinander direkt kommunizieren konnten. Die Vermutung besteht, dass diese Skalierungsfunktion nicht für P2P-Netzwerke geeignet ist, denn im Normalfall wird sie wohl verwendet einen Service auf einem Tier zu skalieren. Beispielsweise können damit Datenbank- oder Webserverinstanzen skaliert werden, die aber natürlich normalerweise nie untereinander kommunizieren müssen. Auch scheint die `--scale`-Option veraltet zu sein und wird durch neuere Features wie `docker swarm` ersetzt. Dieses Problem wurde schlussendlich so gelöst, dass nun dynamisch für jeden zu erstellenden Knoten eine i2pd-Container-Deklaration anhand eines JSON-Templates generiert wird. Dieses JSON-Template ist in `docker/i2p-node-base.json` zu finden. Die fehlenden Werte, so wie z.B. eine statische IP, werden beim ausführen des `Recompose`-Skripts ausgefüllt. Anschliessend werden die verschiedenen Deklarationen zusammengeführt und in eine generierte `docker-compose.yml`-Datei abgelegt. Dies funktionierte sehr gut da, YAML ein Superset von JSON ist. Vereinfacht gesagt: Jede JSON-Datei ist auch eine gültige YAML-Datei.

Im folgenden Quellcodelisting sind die beiden relevanten Funktionen aus der Datei `lib/compose.bash` aufgezeigt. Die Funktion `generate_node_config` erstellt aus dem JSON-Template eine einzelne Service Konfiguration. Insbesondere werden in Zeile 69 die fehlenden Felder im JSON-Template ausgefüllt. Die Funktion `generate_all_node_configs` erstellt die Datei, die dann anschliessend von Docker-compose eingelesen werden kann.

---

```

61 generate_node_config() {
62     local id="$1"
63     local i2pd_config_file="$2"
64     local segment3=$((id / 256 + 128))
65     local segment4=$((id % 256))
66
67     local nodefile="$(mktemp)"
68
69     jq --arg ipv4_address "10.23.${segment3}.${segment4}" \
70       --arg name "\${COMPOSE_PROJECT_NAME}_i2pd_${id}" \
71       --arg volume_dir "./docker/volumes/i2pd-data-${id}:/home/i2pd/data" \
72       --arg confarg "i2pd_config_file=${i2pd_config_file}" \
73       --arg bandwidth "$(get_bandwidth_${id})" \
74       '_.container_name=_$name
75     |_.networks["i2ptestnet"]["ipv4_address"]=_$ipv4_address
76     |_.volumes_+=_[ $volume_dir ]
77     |_.build.args_+=_[ $confarg ]
78     |_.environment["BANDWIDTH"]=_$bandwidth
79     |_.environment["RESEED_IP"]=_$RESEED_IP
80     < "$base_dir"/docker/i2p-node-base.json \
81     > "$nodefile"
82
83     if [[ "$(getconf_network.private)" == "true" ]]; then
84         jq --arg reseed_wait_url "http://$RESEED_IP:8443/i2pseeds.su3" \

```

```

85         '.environment["RESEED_WAIT_URL"]=$_$reseed_wait_url' \
86         < "$nodefile" \
87         | sponge "$nodefile"
88     fi
89
90     jq --arg i $1 '{("i2pd_$_+$i):_._}' < "$nodefile"
91     rm "$nodefile"
92 }
93
94 generate_all_node_configs() {
95     local amount="$1"
96     local i2pd_config_file="$2"
97
98     (
99         for i in $(seq 1 "$amount"); do
100             generate_node_config "$i" "$i2pd_config_file"
101         done
102     ) | jq -s 'add|_{"version":_"3.8",_"services":_._}'
103 }

```

Da hiermit pro Knoten eine statische IP pro Container vergeben wird, konnte mit dieser Lösung nicht nur das Skalierungsproblem, sondern auch die Netzwerkprobleme gelöst werden. Die konfigurierte Anzahl an Containern können nun direkt miteinander kommunizieren.

Die Skalierung auf 128 Knoten auf einem einzelnen Rechner hat sich als weiteres Problem herauskristallisiert, dass die Knoten nicht miteinander kommunizieren können. Nach einer Fehleranalyse hat sich herausgestellt, dass die ARP-Neighbor-Tabelle der Test-VM überlaufen war. Dies ist geschehen, da sich die Container eine einzelne ARP-Neighbor-Tabelle teilen, nämlich die des Hosts. Als Lösung wurden die ARP-Neighbor-Tabelle des Hosts einfach vergrößert. Aus diesem Grund wurde ein zusätzliches Skript erstellt, welches Root-Rechte benötigt, da ein `sysctl`-Befehl gebraucht wird um die Linux-Kernel-Parameter umzustellen. Hier der relevante Ausschnitt aus `bin/networking-sysctls`, der die Grösse der ARP-Neighbor-Tabelle erhöht, abhängig von der konfigurierten Anzahl an Knoten:

```

9 num_nodes="$(getconf _'nodes.amount')"
10
11 arp_table_size=$((num_nodes * num_nodes))
12
13 sysctl net.ipv4.neigh.default.gc_thresh1=$arp_table_size
14 sysctl net.ipv4.neigh.default.gc_thresh2=$((arp_table_size * 2))
15 sysctl net.ipv4.neigh.default.gc_thresh3=$((arp_table_size * 2))

```

### 5.5.2. Bootstrapping und Reseed-Server

Die `i2pd`-Implementation beinhaltet selber keinen Reseed-Server, sondern es ist eine zusätzliche Software ist vonnöten. Die Go-Implementation dieses Reseed-Servers ist hier in meinem Repository zu finden:

<https://codeberg.org/mkuettel/i2p-tools>

Diese basiert auf der Implementation die auch von DIVA.EXCHANGE verwendet wird:

<https://codeberg.org/diva.exchange/i2p-tools>

Welche schlussendlich wiederum auf der folgenden Implementation auf GitHub basiert:

Die Implementation aus meinem eigenen Repository wurde insofern angepasst, dass alle verfügbaren RouterInfos verwendet werden. Standardmässig werden nur 75 % davon überhaupt verwendet und bei einer Anfrage an die Knoten abgeliefert. Der Grund dafür ist leider unbekannt, jedoch macht dies für meinen Anwendungszweck im Testnetzwerk keinen Sinn. Man braucht so nur unnötig mehr Knoten um das Netzwerk aufstarten zu können. Deshalb wurden ich auch die Anzahl an RouterInfo-Dateien und die Anzahl an verschiedenen SU3-Dateien mit einem verschiedener Liste von RouterInfo-Dateien konfigurierbar gemacht. Die Konfiguration dieser Optionen kann eine wichtige Rolle spielen, denn diese bestimmen die Liste an Peers die den I2P-Routern geliefert beim Start geliefert werden und somit den Konnektivitätsgrad der Knoten.

## Ablauf

Der Bootstrapping-Vorgang im Testnetzwerk wird wie folgt abgehandelt (vgl. Abbildung 5.2 “Der Bootstrapping-Vorgang”):

1. Erstellen der Container für den Reseed-Sever und den I2P-Knoten
2. Der Reseed-Server erstellt die nötigen Schlüssel und Zertifikate, um einerseits später den HTTPS-Reseed-Server zu starten und andererseits die SU3-Dateien zu signieren.
3. Der Reseed-Server-Container fängt an zu warten bis er alle RouterInfo's von allen I2P-Knoten erhalten hat, die im Hintergrund vom Recompose-Skript gesammelt werden.
4. Die I2P-Router werden kurzzeitig gestartet, bis sie Ihre RouterInfo Datei erstellen. Die I2P-Router werden anschliessend wieder gestoppt, da der Reseed-Server im Moment noch nicht in Betrieb ist, da ihm die RouterInfos noch fehlen.
5. Die I2P-Knoten-Container warten anschliessend, bis der HTTPS-Reseed-Server verfügbar ist.
6. Auf der Test-VM kopiert ein Hintergrund-Job alle RouterInfos vom I2P-Container in die NetDb des Reseed-Server-Containers.
7. Nach kurzer Zeit hat der Reseed-Server-Container alle benötigten RouterInfos. Nun generiert dieser die SU3-Datei aus den RouterInfos und startet anschliessend effektiv den HTTPS-Reseed-Server.
8. Die I2P-Knoten-Container können nun den HTTPS-Reseed-Server erreichen und somit wird der I2P-Router gestartet.
9. Die I2P-Router laden die SU3-Datei herunter und können so die anderen Knoten ausfindig machen und Ihre Arbeit starten.

Welche Reseed-Server ein I2P-Router anfragt, kann in der `i2pd.conf` mittels der Konfigurationsoption `ur1s` festgelegt werden.

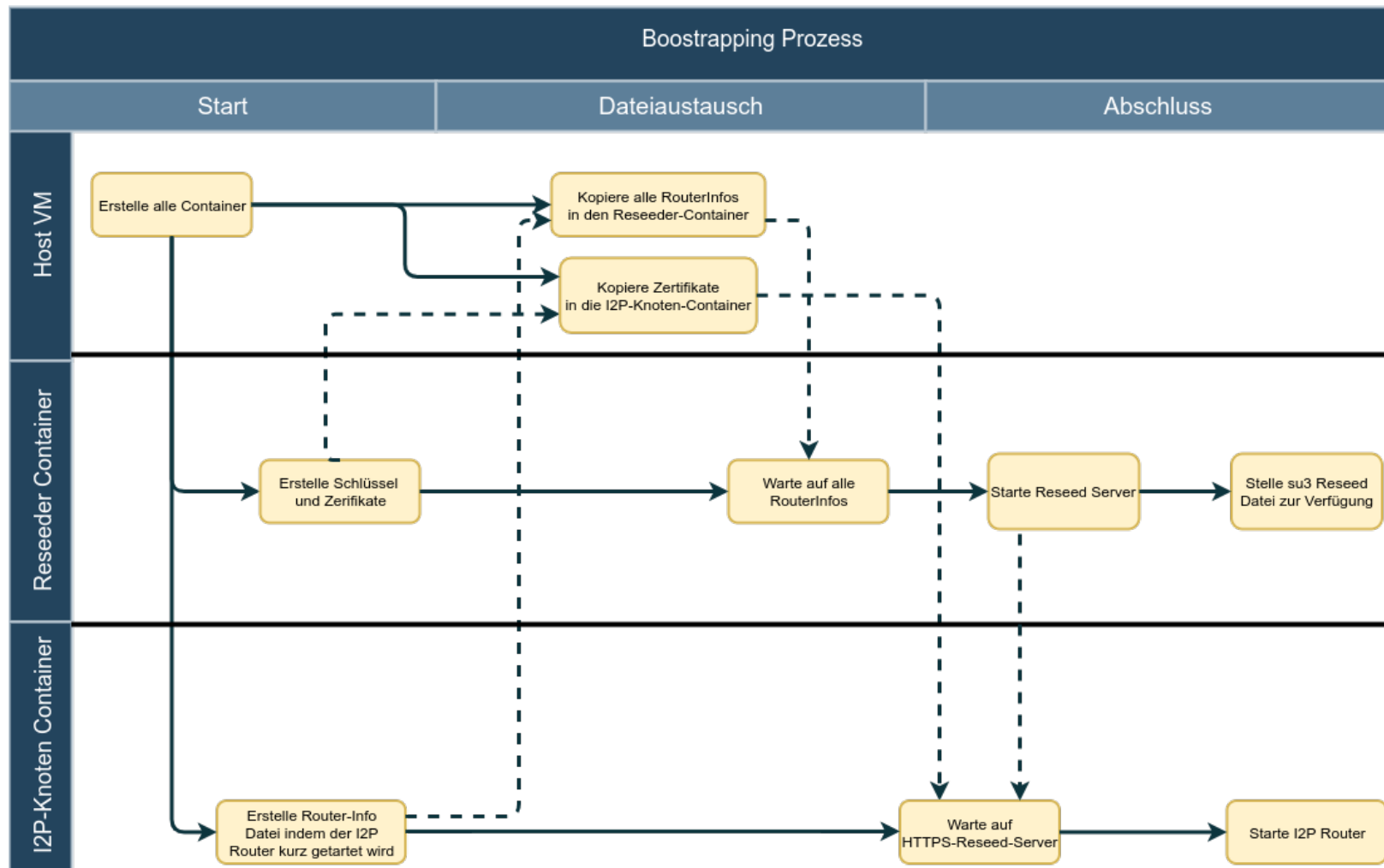


Abbildung 5.2.: Der Bootstrapping-Vorgang

### 5.5.3. Konfigurationsoptionen

Diverse Konfigurationsoptionen wurden implementiert um verschiedene Tests durchführen zu können und werden in der JSON-Datei `config.json` im I2P-Testnet Repository abgelegt. Jede der Optionen ist ein Schlüssel in der JSON-Datei und ist in der folgenden Auflistung kurz beschrieben:

- Netzwerk-Optionen (`network`):
  - `name`: Name des Docker-Netzwerks.
  - `private`: Wird auf `true` gesetzt, wenn es sich um ein privates Testnetzwerk handeln soll. Die Einstellung `false`, um dem öffentlichen I2P-Netzwerk beizutreten, wurde nicht komplett implementiert und nicht getestet.
- Reseeder-Optionen (`reseeder`):
  - `num_router_infos_per_node`: Anzahl RouterInfos pro SU3-Datei. So viele Peers kennt ein Knoten beim Start des Netzwerks.
  - `num_different_seed_files`: Anzahl Verschiedene SU3-Dateien. Der Reseed-Server wählt so viele verschiedene randomisierte Sets an RouterInfos aus.
- Knoten-Optionen (`nodes`):
  - `amount`: Anzahl Knoten im Testnetzwerk. Diese Option bestimmt wie viele i2pd-Knoten ins Testnetzwerk eingebunden werden.
  - `config_file`: Relativer Pfad zur i2pd-Konfigurationsdatei (`i2pd.conf`) für alle Knoten. Darin können alle i2pd-Optionen gesetzt werden (z.B. Tunnellänge, Netzwerk-ID, etc.).
  - `bandwidth`: Eine Liste von Zahlen. Dabei stellt jede Zahl eine Bandbreitenlimite (in kB) eines einzelnen Knotens dar. Das heisst die erste Zahl ist die Bandbreitenlimit des ersten Knotens, die zweite Zahl die Bandbreitenlimite des zweiten Knotens, usw. Sind zu wenige Zahlen in der Liste verfügbar wird wieder von Vorne angefangen. Wird nur eine Liste mit einer Zahl angegeben, so gilt dies also für alle Knoten.
- Mess-Optionen (`measurement`):
  - `message_size_kb`: Nachrichtengrösse (in kB).
  - `num_messages`: Anzahl an sequentiellen Nachrichten die versandt werden.
  - `sleep_interval`: Wartezeit in Sekunden zwischen zwei sequentiellen Nachrichten.

Eine Beispiel `config.json` könnte zum Beispiel so aussehen.

```
1 {  
2     "network": {  
3         "name": "network.i2pd.local",  
4         "private": true  
5     },  
6     "reseeder": {  
7         "router_infos_per_node": 48,  
8         "num_different_seed_files": 64  
9     },  
10    "nodes": {  
11        "amount": 64,
```

```
12     "config_file": "conf/i2pd.org.conf",
13     "bandwidth": [1000000]
14 },
15 "measurement": {
16     "message_size_kb": 16,
17     "num_messages": 1024,
18     "sleep_interval": 6
19 }
20 }
```

---

#### 5.5.4. Netzwerkisolation

Während unserer Tests wollen wir nur Traffic unserer eigenen Knoten im Netzwerk und äussere Einflüsse vermeiden. Dies, um bestmögliche und unbeeinflusste Messungen tätigen zu können. Docker-compose respektive Docker bietet eine Option, ein internes Netzwerk zu erstellen, welches einerseits vom Host nicht direkt erreichbar ist und zudem die Container nicht auf das Host-Netzwerk zugreifen können. Das folgende Quellcodelisting zeigt den Abschnitt für die Netzwerkkonfiguration wie deklariert in der docker-compose.yml. Wichtig sind hier die Optionen `driver: bridge` und `internal: true`, die ein internes Docker-Netzwerk erstellen. Die zusätzliche Option `enable_icc` stellt sicher, dass die Container jedoch untereinander kommunizieren können.

---

```
12 networks:
13   i2ptestnet:
14     name: $NETWORK_NAME
15     driver: bridge
16     internal: true
17     driver_opts:
18       com.docker.network.bridge.enable_icc: "true"
19       com.docker.network.bridge.enable_ip_masquerade: "true"
20   ipam:
21     driver: default
22     config:
23       - subnet: 10.23.0.0/16
```

---

Zusätzliche Optionen zur Abgrenzung vom öffentlichen I2P-Netzwerk können in der `i2pd.conf` spezifiziert werden. Einerseits gibt es die Option `netid`; diese ist standardmässig auf die Nummer 2 gesetzt, da dies die Netzwerk-Id des öffentlichen I2P-Netzwerks ist. Diese Option ist im Testnetzwerk jedoch zur Abgrenzung auf die Nummer 23 gesetzt. I2P-Router mit einer anderen Netzwerk-ID werden nie miteinander kommunizieren. Für Tests mit dem öffentlichen Netzwerk gäbe es zusätzlich noch die Option `family` („Family configuration - i2pd documentation“, n. d.).

#### 5.5.5. Adressbuch

Standardmässig wird jeder Destination eine b32-Adresse gegeben. Diese ist jedoch nicht einfach zu handhaben und stattdessen ist es einfacher, die Knoten direkt anhand ihrer Nummer zu adressieren. Diese Adressen werden dann automatisch vom Socks5-Proxy, der `i2pd` mitliefert aufgelöst. Damit kann das Messskript vereinfacht werden, da es dann nicht die genauen Adressen zusammensuchen muss.

Auch wird die Auswertung damit vereinfacht, weil der TCP-Client den verwendeten Namen des TCP-Servers auch in die Nachricht hineinpackt und der TCP-Server diesen als Messausgabe abspeichert. Es wird beim Startvorgang des Netzwerks für jede erstellte Destination für den TCP-Server ein Adressbucheintrag erstellt. Dieses Adressbuch ist eine einfache CSV-Datei wird ebenfalls im Bootstrappvorgang an alle Knoten verteilt. Im Falle von acht Knoten im Netzwerk sieht diese Datei beispielsweise so aus:

---

```
tcpsrv-1.i2p,b2cdodoyqg5v6go56bkaevvw777e2wyiflqh6zyyyxugy4cqlzuq
tcpsrv-2.i2p,r6axddltht76mrtp5lrxtdxncilhgp7bcajkqtsj7l7fmoebgs3a
tcpsrv-3.i2p,ddhznqpxtgac7qa6yg6uzfpd55xoa2zfc4dwg4rh4qucx7veswlq
tcpsrv-4.i2p,3tyncnh7j4akqirrbdi4ek7nlysqmuvyav3ix326vwixvgoviz3q
tcpsrv-5.i2p,xj3qduyqur54v3mjgsxn6xa56q2ojyysf5n3bonpxyfyx5trnoja
tcpsrv-6.i2p,pt64eojrsvrp5w5izgwlide7j6vuw5td6g2pfrkurwsnmb4i7q2q
tcpsrv-7.i2p,aat6wqge3o7k2nabpwxrcu2vbbiydf3mj6n5k2zrrtozkrpyfoa
tcpsrv-8.i2p,4wm4pqautqqb3a6yw3xif4zgwwu4v5tvpwv6onha27p5pkcqepq
```

---

In der ersten Spalte ist der vereinfachte Name angegeben und in der zweiten Spalte die b32-Adresse, jedoch ohne die Endung .i2p.

### 5.5.6. TCP-Server

Für Tests am Anfang wurde das Tool Netcat verwendet. Dies hat gut funktioniert, um die Konnektivität im Netzwerk zu testen, hat aber zur effektiven Messung nicht funktioniert, da es nicht geklappt hat die TCP-Verbindung nachdem eine Nachricht erhalten wurde zu schliessen, was dazu führte, dass das Messskript stecken blieb. Auch erlaubt es Netcat nicht mehrere TCP-Verbindungen anzunehmen und die Verbindungen korrekt zu schliessen. Deshalb wurde eine `epoll`-basierte C-Implementation eines TCP-Servers erstellt, um die komplette Kontrolle über den Netzwerksocket zu bekommen. Dies erlaubt es die Latenz genauer zu messen und auch den TCP-Verbindungsaufbau nicht mitzumessen, sowie die Nachrichtenübermittlung auf Socket-Ebene kontrollieren zu können. Somit konnte die TCP-Verbindung nach Erhalt einer Nachricht korrekt geschlossen werden. Auch wurde C und nicht z.B. Python gewählt, um den Arbeitsspeicher und CPU-Gebrauch der Containers so tief wie möglich zu halten. Die Implementation des TCP-Servers ist in der Datei `docker/i2p/messenger/epoll-server.c` zu finden.

### TCP-Server Tunnel

Diese Tunnel-Konfiguration führt dazu, dass I2P eine neue Destination mit einer .b32-Adresse erstellt, damit der TCP-Server innerhalb des I2P-Netzwerks erreicht werden kann. Mit der folgenden Tunnel-Konfigurations-Datei wird der `i2pd` konfiguriert, um einen TCP-Server-Tunnel zur Verfügung zu stellen:

---

```
1 # serve some tcp service to others in the network
2 [tcp-in]
3 type = server
4 host = 127.0.0.1
5 port = 2323
6 keys = tcp-in.dat
7 gzip = false
8 enableuniquelocal = true
```

---



Hierbei gilt es zu beachten, dass die Schlüssel für die Destination, angegeben mit der `key`-Option, beim Aufstarten generiert werden. Auch wurde mittels der Option `gzip` die Komprimierung deaktiviert, damit auch wirklich die richtigen Nachrichten-Größen übermittelt werden. Die Option `enableuniquelocal` wurde ursprünglich aktiviert, um die Auswertung zu vereinfachen. Denn diese führt dazu, dass der TCP-Server nicht immer dieselbe Ausgangsadresse `127.0.0.1` bekommt, sondern je nach Sender der Nachricht eine andere `127.0.0.0/8`-Adresse anhand der ersten Bytes des Public-Keys erhält. Jedoch wurde schlussendlich nicht auf diese Möglichkeit die Knoten zu identifizieren zurückgegriffen, da die Knoten einfacher mithilfe des Adressbuch identifiziert werden konnten.

### 5.5.7. TCP-Client

Damit der TCP-Client sich mit dem I2P-Overlaynetzwerk verbinden kann, muss er fähig sein mit einem Socks5-Proxy zu kommunizieren. Zudem gilt es den Socks5-Proxy richtig zu konfigurieren, denn die Namensauflösung muss auch über das I2P-Netzwerk abgehandelt werden. Nur so können die `b32`-Adressen aufgelöst werden, sofern diese im Adressbuch vorhanden ist. Anhand einer `.b32.i2p`-Adresse kann die NetDB nach einem zugehörigen LeaseSet abgefragt werden, welches Zugriff auf die gewünschte Destination bietet. Der TCP-Client ist ein kleines C-Programm, welches in der Datei `docker/i2p/messenger/client.c` zu finden ist. Zusammengefasst macht es nichts anderes als eine Verbindung mit einem Socks5-Proxy aufzubauen, und eine TCP-Nachricht der definierten Größe mit ein durch den Proxy an die richtige Destination zu versenden. Wie diese Nachricht aufgebaut ist wird im Abschnitt `sec:nachricht_` *latenzmessungen genauer erläutert. Das TCP – Client Programm bietet folgende Kommandozeilen – Optionen :*

---

```

1 Usage: messenger [-h] [-S proxy_server] [-P proxy_port] [-d destination]
   [-p dest_port]
2 Parameters:
3   -h                display command line help
4   -S proxy_server   proxy server host name or IP address
5   -P proxy_port     proxy port number
6   -d destination    the hostname or ip of the server connected through
   the proxy
7   -p dest_port      destination port number
8   -m messagesize    size of the message to send in Kb
9   -o originator      an identifier for the originator of the message
10  -v                verbose mode
11  -d                debug mode (overrides -v)
12 Description:
13 This client is used to send messages to measure the latency of the i2p
   network

```

---

### Socks5-Proxy für den TCP-Client

Dieser Socks5-Proxy ist der Gateway um mit dem I2P-Overlaynetzwerk zu kommunizieren. Der folgende Ausschnitt aus der `i2pd.conf` zeigt auf wie der Socks5-Proxy konfiguriert wurde:

---

```

1 [socksproxy]
2 ## Uncomment and set to 'false' to disable SOCKS Proxy
3 enabled = true
4 ## Address and port service will listen on, like 127.0.0.1

```

---

```
5 address = 127.0.0.1
6 port = 4445
```

---

Zusätzlich könnte in diesem Abschnitt auch die Länge der Tunnels bestimmt.

### 5.5.8. Nachricht zur Latenzmessung

Die darüberliegende Blockchain-Schicht von DIVA.EXCHANGE versendet mittels eines Gossip-Protokolls die neuen Blocks der Blockchain an alle Knoten. Diese 64KB grossen Blocks werden über eine Web-Socket-Verbindung an die Nachbarknoten übermittelt. Die Nachbarknoten leiten diese dann wiederum an ihre jeweiligen Nachbarknoten weiter. Dieser Prozess wird solange wiederholt bis alle Knoten im Netzwerk den neuen Block erhalten haben. Nach Absprache mit dem Auftraggeber wird dies im Testnetzwerk mittels versenden von TCP-Nachrichten nachgestellt. Bei Web-Sockets handelt es sich vereinfacht ausgedrückt wiederum um eine TCP-ähnliche Verbindung über HTTP. Zudem verwendet HTTP auch TCP auch als Transportschicht.

Eine Nachricht welche vom TCP-Client zum Server übermittelt wird, besteht aus folgenden Elementen:

- Zufällige Daten (standardmässig 64KB, konfigurierbar)
- Knoten Nummer des Senders
- Destination-Adresse des Empfängers vom Adressbuch (beinhaltet auch die Knoten Nummer des Empfängers)
- Grösse der versendeten Nachricht in Bytes (standardmässig 63255)
- Versandzeitpunkt in Nanosekunden seit dem 1.1.1970
- Empfangszeitpunkt in Nanosekunden seit dem 1.1.1970

Der TCP-Server legt alle erhaltenen Nachrichten im I2P-Container in einem Nachrichten-Log ab.

### 5.5.9. Das Messskript

Die Nachrichten zur Latenzmessung werden versendet unter Verwendung des docker `exec`-Befehls. Damit wird innerhalb des zufällig ausgewählten i2pd-Containers ein TCP-Client-Prozess gestartet. Dieser verbindet dann auf den Socks5-Proxy im gleichen Container. Anschliessend schickt der TCP-Client die Nachricht der gewählten Grösse ins I2P Netzwerk.

Bevor das Messskript gestartet werden kann, müssen jedoch einige Bedingungen erfüllt werden. Erstens muss mittels dem Recompose-Skript das Testnetzwerk in Betrieb genommen werden. Zweitens muss gewartet werden, bis der Bootstrapping-Vorgang im Testnetzwerk abgeschlossen wurde. Zu guter Letzt muss noch einige Zeit gewartet werden und sichergestellt werden, dass das Netzwerk bereit ist und die Exploratory-Tunnels aufgebaut sind. Dies ist um sicherzustellen, dass keine Messungenauigkeiten auftreten. Einerseits gehen so keine Messpakete verloren, andererseits könnten die Latenzmessungen verfälscht werden, wenn die Tunnels erst noch aufgebaut werden müssen. Das Messskript ist in der Repository in der Datei `bin/measure-message-latency` zu finden.

## 5.6. Testing

Da ein gesamtes Netzwerk simuliert wurde mit dynamisch vielen Containern, konnte nicht automatisch unter Verwendung von Continuous Integration getestet werden. Viele Lösungen (wie zum Beispiel das angebotene GitLab im EnterpriseLab der HSLU) verwenden eine andere Syntax zur Definition von Containern als Docker-compose und die definierten Container lassen sich nicht skalieren. Somit musste oft auf manuelles Testen zurückgegriffen werden. Jedoch hat sich während des Entwickelns herausgestellt, dass man oft dieselben Befehle auf der Kommandozeile wiederholt hatte. Deshalb hat man im Unterordner `tests/` in der Repository einige Tests geschrieben in Bash abgelegt, um gewisse Grundfunktionalitäten zu testen. Diese Tests verwenden das Bash Automated Testing System (BATS) Eine Testsuite kann zum Beispiel folgendermassen gestartet werden:

---

```
$ bats tests/networking.bats
```

---

Diese Testsuite wurde erstellt um schnell zu prüfen, ob das Netzwerk korrekt konfiguriert wurde. Das heisst, dass die I2P-Knoten das Internet nicht erreichen können, jedoch den Reseeder und die anderen Container schon. Diese Tests sind im Verzeichnis.

Dieser Bericht wurde mit Hilfe von LaTeX umgesetzt. Dabei handelt es sich bei dem Text um eine Art Programm mit einem Build-Vorgang der ein PDF erzeugt. Dieser Build-Vorgang wurde mit dem GitLab CI Service vom EnterpriseLab automatisiert.

## 5.7. Benutzerhandbuch

Dieser Abschnitt erklärt, wie ein Tester selber neue Testnetzwerke erstellen, die Tests-konfigurieren und Messungen tätigen kann.

### 5.7.1. Deployment der Test-VM

Wird NixOps verwendet, kann die VirtualBox Test-VM wie folgt erstellt werden:

---

```
$ nixops create single-vm-testnetwork.nix deployment/vbox-containers.nix  
$ nixops deploy -d vbox-i2p-testnet
```

---

Anschliessend kann mittels ssh auf die Test-VM zugegriffen werden.

---

```
$ nixops ssh vbox-i2p-testnet
```

---

Für genauere Anweisungen und weitere Features, welche durch NixOps geboten werden, siehe das "NixOps User's Guide"<sup>2</sup>. Alternativ kann auch jede andere Linux-Distribution mit oder ohne VM verwendet werden. Wichtig ist, dass folgende Software-Pakete installiert sind und mindestens die angegebenen Versionen (wo spezifiziert) verwendet werden:

- docker-compose >= 1.28.2, verwendet für Container-Orchestration
- docker >= 20.10, verwendete Container-Technologie
- bash, benötigt da das Recompose-Skript und das Messskript in Bash geschrieben sind

---

<sup>2</sup><https://releases.nixos.org/nixops/nixops-1.5/manual/manual.html>

- jq, Utility zum Verarbeiten von JSON-Dateien
- moreutils, Weitere Shell-Utilities die vom Recompose-Skript verwendet werden (z.B. sponge).
- bats, optional, wird benötigt um Selbsttests auszuführen
- sysstat, wird vom Messskript benötigt, um die Ressourcenauslastung zu messen

### 5.7.2. Konfiguration einer Messung

Nachdem die i2p-testnet-Repository in der Test-VM heruntergeladen wurde, kann bereits mit der Konfiguration der Messung gestartet werden. Die Konfiguration der Messung kann in der Datei config.json angepasst werden.

### 5.7.3. Erstellen des Testnetzwerkes

Für alle folgenden Befehle wird davon ausgegangen, dass ein Benutzer verwendet wird der Teil der docker Gruppe ist.

Das Testnetzwerk kann mit folgenden Befehl gestartet werden:

---

```
$ bin/recompose
```

---

Wird gewünscht bei einer Messung mit einem komplett frischen Netzwerk zu starten sollte folgender Befehl verwendet werden.

---

```
$ bin/recompose -c
```

---

**Achtung:** Dieser Befehl löscht alle bestehenden Docker-Volumes sowie Logs und Messresultate von laufenden Messungen welche noch nicht vom Messskript in den results/-Ordner abgelegt wurden.

### 5.7.4. Ausführen einer Messung

Wird eine Messung mit 64 Knoten oder mehr durchgeführt, muss die ARP-Neighbor-Tabelle des Hosts vergrößert werden, da der Standardwert (siehe Schluss des Abschnitts 5.5.1 "Skalierung"). Dies benötigt zwingend Root-Rechte, da Kernel-Parameter verändert werden:

---

```
$ sudo bin/networking-sysctl
```

---

Eine Messung kann wie folgt ausgeführt werden:

---

```
$ bin/measure-message-latency
```

---

Die Messresultate können anschliessend im results/-Ordner aufgefunden werden. Darin sind alle von den I2P-Knoten erstellten Daten deren sowie Docker-Logs zu finden. Zudem gibt es eine Datei resource-usage in der die Ressourcenauslastung des Testnetzwerks geloggt wurde. Darüber hinaus ist in jedem I2P-Daten Ordner eine Datei namens i2p-data-<ID>/messagelogs/msglog.csv zu finden, welche alle Nachrichten geloggt hat die der jeweilige Knoten mit erhalten hat.

## 6. Evaluation und Validation

In diesem Kapitel werden nun die Ergebnisse dieser Arbeit ausgewertet. Im ersten Abschnitt 6.1 “Messresultate” werden die Messungen ausgewertet und die Messresultate graphisch illustriert und interpretiert. Anschliessend wird im Abschnitt 6.2 “Vergleich mit Anforderungen” überprüft inwiefern die Projektanforderungen erfüllt werden konnten. Daraufhin werden im Abschnitt 6.3 “Technische Aspekte” die verwendeten Tools und die Architektur evaluiert. Zuletzt wird das Vorgehen während des Projekts beurteilt (siehe Abschnitt 6.4 “Vorgehen”).

### 6.1. Messresultate

Als Erstes wurde die Testumgebung dahingehend überprüft, ob sie überhaupt sinnvolle Resultate misst. Anschliessend wurden zwei verschiedene Arten von Messungen durchgeführt. Für weitere Messungen, vor allem für Messungen mit Bandbreitenbeschränkung, war die Zeit leider zu knapp.

Bei jeder Messung wurden 1000 Nachrichten von einem zufälligen Knoten verschickt und von einem anderen zufälligen Knoten im Testnetzwerk wieder empfangen. Zudem wurde die konfigurierte Tunnellänge bei jeder Messung auf dem Standardwert 3 belassen. Die Nachrichten wurden jeweils in einem Abstand von sechs Sekunden nacheinander verschickt, um das Netzwerk auf keinen Fall zu überlasten. Zudem wurde jeweils die CPU-Ressourcenauslastung und Netzwerkbelastung alle 5 Sekunden gemessen.

#### 6.1.1. Validation der Testumgebung

Um Aussagen über das Testnetzwerk und dessen Messresultate machen zu können, muss erst sichergestellt werden, dass die Messwerte überhaupt Sinn ergeben. Deshalb wurden erste Messungen und Tests durchgeführt, um zu prüfen, ob die Testumgebung auch richtig funktioniert.

Als Teil dieser Messung wird erst ein Netzwerk mit 8 Knoten getestet und die Latenz von 1000 Nachrichten gemessen. Alle Knoten haben in diesem Test keine Bandbreitenbeschränkung. Die konfigurierte Nachrichtengrösse wurde jeweils auf 64 kB festgelegt.

Als erstes wurde die Messumgebung auf meinem Laptop (8 Kerne, Intel Core i7-8550U CPU @ 1.80 GHz, mit 16 GB RAM) getestet mit 8 Knoten und dieselbe Messung mehrfach wiederholt. Dies hatte jeweils fast immer dasselbe Resultat zur Folge: Die Nachrichten wurden mit einer Latenzzeit von etwas mehr als fünf Sekunden empfangen. Einige Ausreisser waren jedoch zu beobachten; Bei manchen Nachrichten liess sich eine Latenzzeit von etwas mehr als sechs Sekunden oder bis hin zu einem Vielfachen von fünf Sekunden beobachten. Die Neuerstellung der Tunnels, Abfragen an der NetDb, andere Effekte wie eine kurzzeitige Überlastung oder das Scheduling des Kernels könnten eine Erklärung sein für die Ausreisser von sechs Sekunden. Die Ausreisser bei welchen sich ein Vielfaches der Latenzzeit beobachten liess, könnten sich durch die wiederholte Übermittlung von Nachrichten im Fehlerfall oder einem Fehler in der i2pd-Software erklären.

Anschliessend wurde eine virtuelle Maschine beim Anbieter Hetzner (16 Kerne, AMD EPYC CPU @ 2.495 GHz, mit 32 GB RAM) bestellt, um dort die Messungen durchzuführen. Alle weiteren Messungen wurden auf dieser Maschine durchgeführt. Dort lieferte derselbe Testaufbau auch bei

wiederholten Messungen dieselben Resultate. Die Nachrichten wurden mit einer Latenzzeit von etwas mehr als fünf Sekunden empfangen und dieselben Ausreisser (jedoch nicht ganz so häufig) von sechs Sekunden bis hin zu einem Vielfachen von fünf Sekunden konnten beobachtet werden. Somit ist sichergestellt, dass die Messungen wiederholbar und reproduzierbar sind und eine Basis einen Referenzpunkt für weitere Messungen bieten können.

### **6.1.2. Gleichbleibende Latenz bei Skalierung des Netzwerks**

Um nun die Hypothese 1 zu bearbeiten, wird als Erstes die Anzahl Knoten skaliert und die Latenz mehrmals unter gleichen Bedingungen gemessen. Die konfigurierte Nachrichtengrösse wurde jeweils auf 64 kB festgelegt. Es wird angenommen, dass sich die Latenz der Nachrichten im Netzwerk hier nicht verändert. Dies mag auf den ersten Anschein der Hypothese widersprechen, jedoch gilt es zu beachten, dass hier alle Knoten dieselbe Bandbreitenbeschränkung von 100000 kB/s haben. Dementsprechend müsste die Peer-Selektion von I2P alle Knoten gleich bewerten. Der einzige Unterschied zwischen einem kleineren Netzwerk und einem grösseren Netzwerk ist somit, dass eine grössere Auswahl an Knoten besteht, an denen eine Nachricht weitergeleitet werden kann. Die Anzahl Knoten in den Tunnels bleibt jedoch gleich, es kommen lediglich weniger Knoten zum Einsatz bei einem kleineren Netzwerk. Die Erwartung ist daher, dass die Latenz bei steigender Anzahl Knoten gleich bleibt.

Die Abbildung 6.1 "Nachrichtenlatenz bei Skalierung der Anzahl I2P-Knoten" zeigt jedoch ein anderes Bild. Während die Mindestlatenz der 1000 Nachrichten konstant auf fünf Sekunden bleibt, vergrössert sich die Median- und Durchschnittslatenz je mehr Knoten im Testnetzwerk vorhanden sind. Vor allem bei einem Netzwerk von 256 Knoten steigt diese beträchtlich an. Dies ist ein überraschendes Ergebnis und die Gründe sind unklar. Es könnte sein, dass das man hier auf gewisse implizite Limits im Linux-Kernel bei dieser Menge an Docker-Containern stösst und dass man hier auf ein Problem mit der Testinfrastruktur gestossen ist. Jedoch ist die CPU-Auslastung im gemessenen Fünf-Sekunden-Abstand nie auf mehr als 20 % angestiegen und im Schnitt betrug diese lediglich 5%.

### **6.1.3. Verhalten der Latenz abhängig von der Nachrichtengrösse**

Bei dieser Messung wurden in einem Testnetzwerk bestehend aus 64 Knoten verschieden grosse Nachrichten versandt. Auch hier haben alle Knoten dieselbe Bandbreitenbeschränkung von 100000 kB/s. Hier wurde erwartet, dass sich die Latenz unter Umständen leicht vergrössert. Dies aufgrund von Zwischenpuffergrössen, so wie auch des zusätzlichen Verschlüsselungsoverheads.

Wie in der Grafik 6.2 "Nachrichtenlatenz bei verschiedener Nachrichtengrösse" zu erkennen ist, verändert sich die Latenz abhängig von der Nachrichtengrösse nicht markant. Ein kleiner Anstieg der Latenz ist jedoch ab der Nachrichtengrösse von 64 kB ersichtlich. Jedoch macht dies keinen grossen Unterschied: Auch wenn die Nachrichtengrösse vervierfacht wird auf 256 kB, scheint das keinen grossen Einfluss auf die Latenz zu haben. Es wurde in diesem Falle sogar eine tiefere Median- und Durchschnittslatenz gemessen, als mit einer Nachrichtengrösse von 128 kB. Es wird vermutet, dass das an der Zwischenpuffergrösse liegt, welche bei I2P ebenfalls auf 64 kB gesetzt ist.

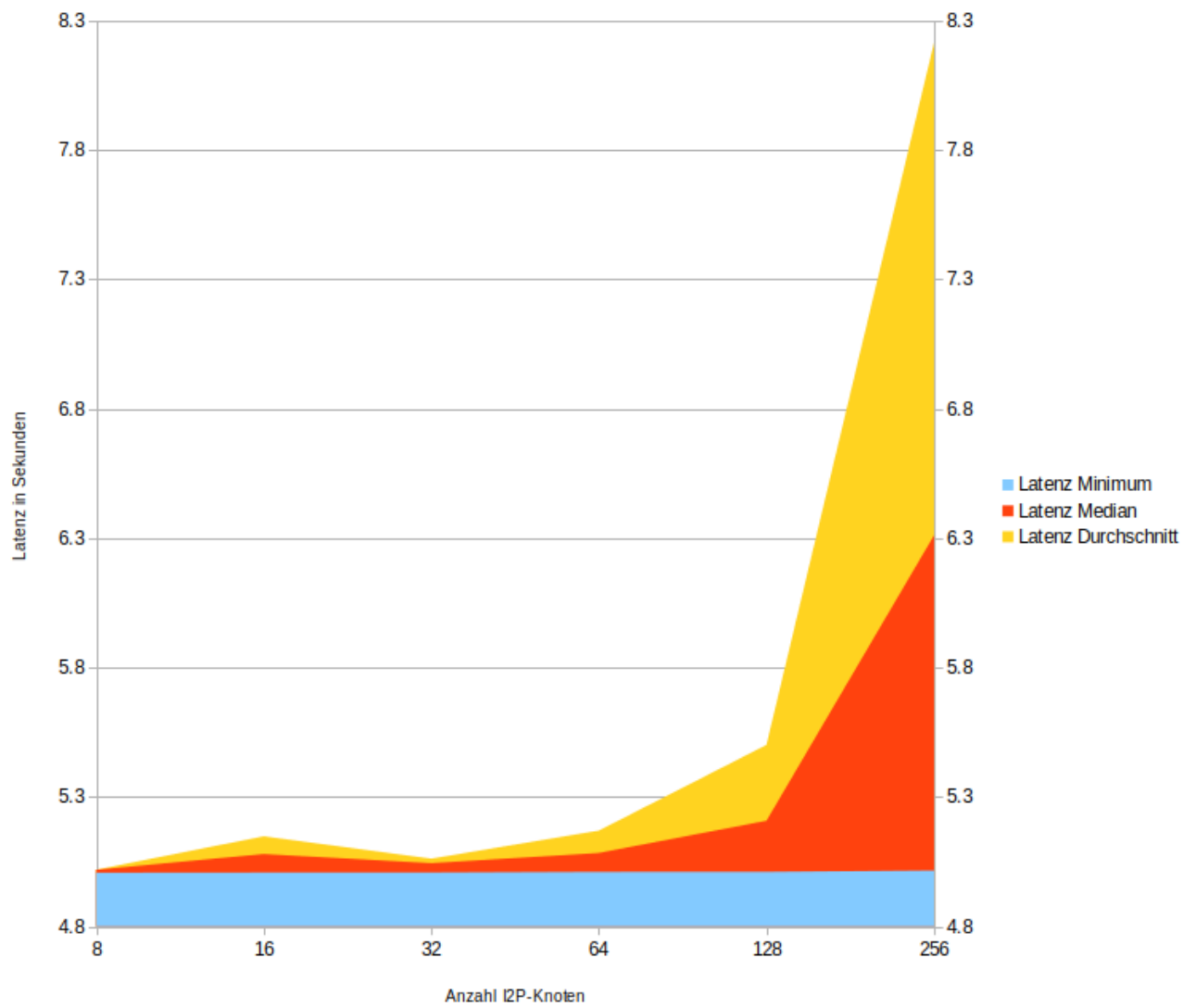


Abbildung 6.1.: Nachrichtenlatenz bei Skalierung der Anzahl I2P-Knoten

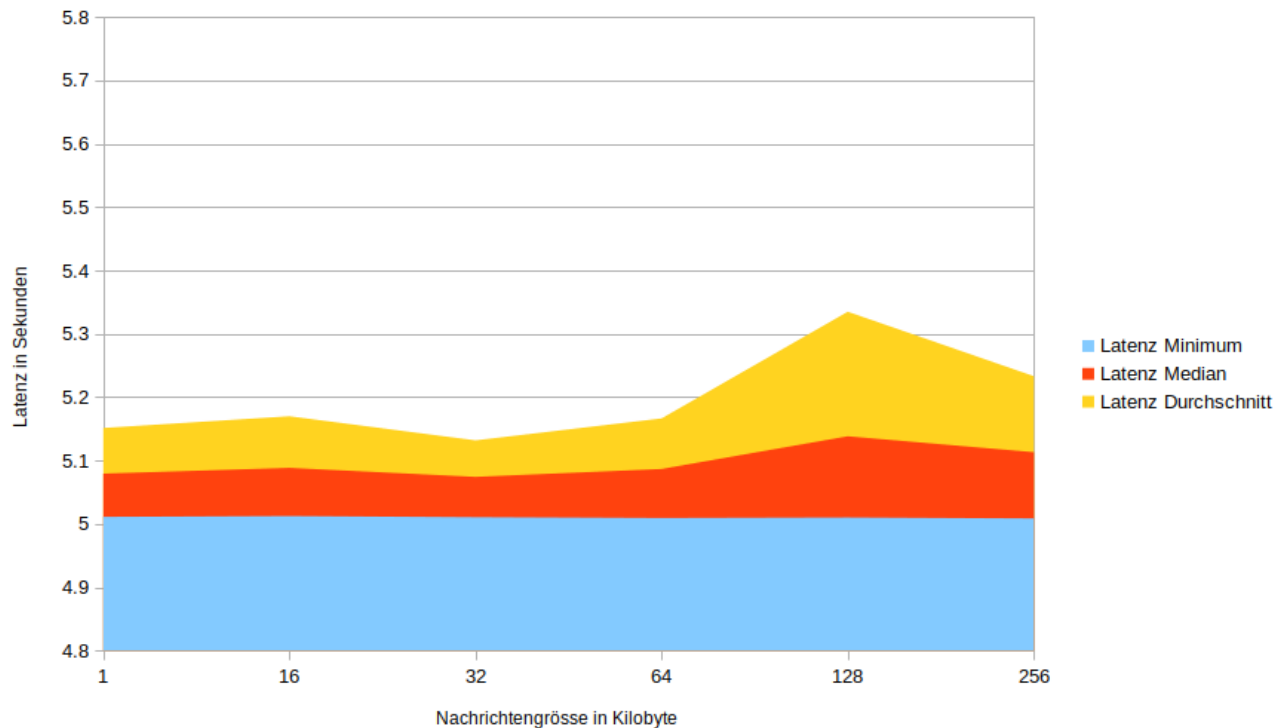


Abbildung 6.2.: Nachrichtenlatenz bei verschiedener Nachrichtengröße

## 6.2. Vergleich mit Anforderungen

Es wurde ein Stand der Technik ermittelt (siehe Anforderung 1-SDTF) im Kapitel 3 “Ideen und Konzepte”, anhand dessen ein Konzept (siehe Anforderung 3-TKON) erstellt und ein Teststand aufgebaut (siehe Anforderung 2-TINF). Es wurde darauf geprüft, ob die Messungen reproduzierbar sind (siehe Anforderung 4-TREP) im Abschnitt 6.1.1 “Validation der Testumgebung”. Es ist möglich, den Teststand und die Messungen zu konfigurieren, wie beschrieben im Abschnitt 5.5.3 “Konfigurationsoptionen” (siehe Anforderung 5-TCNF). Dabei kann man den Teststand auf bis zu 256 Knoten skalieren (siehe Anforderung 6-TSCL), wie beschrieben im Abschnitt 5.5.1 “Skalierung” und gezeigt bei der Messung im Abschnitt 6.1.3 “Verhalten der Latenz abhängig von der Nachrichtengröße”. Dabei ist das interne I2P-Netzwerk vom realen Testnetzwerk abgeschottet (siehe Anforderung 7-TISO), wie beschrieben im Abschnitt 5.5.4 “Netzwerkisolation”. Optional hätten auch Tests im öffentlichen I2P-Netzwerk durchgeführt werden können (siehe Anforderung 8-TNET). An verschiedenen Stellen im Programmcode wurde dies so vorgesehen, dass es in Zukunft möglich wäre. Jedoch wird es nicht komplett unterstützt und wurde auch nie getestet. Eine Messanlage zur Latenzmessung (siehe Anforderung 9-TLAT) wurde entwickelt. Es wurde die Möglichkeit implementiert die Bandbreite der einzelnen I2P-Knoten zu limitieren (siehe Anforderung 10-TLIM), jedoch wurde diese aus Zeitgründen nie komplett für Messungen eingesetzt; die anderen Experimente hatten Vorrang. Es ist schnell möglich, ein neues Mess-Experiment zu starten (siehe Anforderung 11-TVRS): Man muss lediglich die Konfiguration anpassen und die Knoten erstellen. Im besten Falle ist dies innerhalb von 5 Minuten möglich. Soll jedoch etwas anderes als eine Latenzmessung durchgeführt werden, müsste die Messanlage angepasst werden. Dies



würde weiteren Entwicklungsaufwand mit sich bringen. Die Evaluation und Auswertung der Messresultate (siehe Anforderung 12-EVAL) wurde im Abschnitt 6.1 “Messresultate” vorgenommen. Die Ausführungszeiten der Tests wurden so gut wie möglich kurz gehalten (siehe Anforderung 13-TPER). Eine Messung mit 1000 Nachrichten dauert fast zwei Stunden, da die Nachrichten im Abstand von sechs Sekunden verschickt werden, um das Netzwerk nicht zu überlasten. Andere Experimente könnten hier eine bessere Test-Performance mit sich bringen.

Es wurde mit Kanban ein iteratives Projektmanagement eingesetzt (siehe Anforderung 15-ITER). Oft aber hatte man zu viel Overhead da ich zu genaue Meeting-Notizen für jedes kleine Meeting erstellt habe. Zudem wurde als Bestandteil der Bachelorarbeit dieser Bericht erstellt (siehe Anforderung 14-DOCS), eine Zwischenpräsentation gehalten (siehe Anforderung 16-PRES), ein Web-Abstract erstellt (siehe Anforderung 17-WEBA) und ein dazugehöriges Pitching-Video produziert (siehe Anforderung 18-PVID). Somit konnten meisten Anforderungen erfüllt werden, inklusive den optionalen Anforderungen.

### 6.3. Technische Aspekte

Mit der Umstellung von der NixOS-basierten Lösung auf die auf Docker-compose-basierte Lösung ist die Möglichkeit, I2P-Knoten auf mehrere Maschinen zu deployen, verloren gegangen. Die Docker-Container werden mit Docker-compose so nur auf einem einzelnen Rechner erstellt, und es können nicht ohne weiteres mehrere Testnetzwerke miteinander verbunden werden. Mit diesem Ansatz kann nicht auf beliebig viele Knoten skaliert werden, da nur vertikal skaliert werden kann. Dementsprechend ist dieser Ansatz durch existierende Hardware limitiert.

Für die Messanlage wurde ein eigener TCP-Server und TCP-Client in C implementiert, um genauere Messresultate zu erhalten. Dies hat jedoch viel Zeit beansprucht und eine einfachere Lösung hätte eventuell ähnlich genaue Resultate liefern können. Die Auswertung der CSV-Dateien mit den Messresultaten mittels LibreOffice Calc (Excel-Pendant) war nicht unbedingt einfach und durchaus mühsam. Vor allem war das Erstellen der Diagramme problematisch. Jedoch war dieses Vorgehen wohl aus Zeitgründen pragmatisch.

Zur Erstellung des Berichts und der Präsentation wurde die  $\text{\LaTeX}$ -Suite eingesetzt. Dies hatte den Vorteil, dass stets in meinem Lieblingstexteditor Vim gearbeitet werden konnte. Zudem bietet mit BibiTeX eine Lösung zur Verwaltung von Quellen für ein Literaturverzeichnis und macht das Handhaben jeglicher Referenzen einfach. Weil es sich um reinen Text handelt, konnte auch die Versionsverwaltung Git für den Bericht eingesetzt werden. Nachteil dieser Lösung ist jedoch, dass die Rechtschreibprüfung von Vim nicht so gut ist wie die Rechtschreibprüfung von Word. Dies hat einen Mehraufwand verursacht. Zudem musste oft im Internet nach passenden Befehlen und Paketen gesucht werden, um gewisse Teile des Berichts zu gestalten. Jedoch konnte man einiges dazulernen und einen schön gestalteten Bericht abliefern.

## 6.4. Vorgehen

Die verwendete Projektmanagement-Methode Kanban war sicherlich hilfreich, auch wenn man sich nicht immer strikt daran halten konnte. Im Nachhinein hätte man sich den einen oder anderen Punkt mehr zu Herzen nehmen sollen, zum Beispiel, dass man nicht gleichzeitig an mehreren Sachen hätte arbeiten sollen. Oft hat man sich während der Entwicklung auch überstürzt und zu viele Änderungen auf einmal vorgenommen, sodass nicht mehr klar war, wieso etwas nun nicht mehr funktionierte. Die wöchentlichen Meetings waren extrem hilfreich, den Kurs des Projektes zu steuern und schnell Feedback einzuholen. Oft hat man sich im Detail verloren, sich zu viel vorgenommen oder überstürzt gehandelt. Dies konnte durch die Meetings jeweils etwas reguliert werden. Während die erstellten Meeting-Notizen hilfreich waren, haben sie aber auch oft zu viel Projektmanagement-Overhead geführt.

## 7. Ausblick

In dieser Grundlagenforschung hat sich herausgestellt, dass sich die Mindestlatenz bei Skalierung des Netzwerks, wenn alle Knoten dieselbe Bandbreitenbeschränkung haben, nicht markant verändert. Das Messresultat für 256 Knoten war jedoch überraschend und dies deutet auf ein Problem mit der Testinfrastruktur hin. Zukünftig könnte die Testinfrastruktur diesbezüglich verbessert werden. Anstatt das man die Knoten nur horizontal auf einer einzelnen Maschine skaliert, würde es Sinn ergeben, diese auf mehrere Maschinen zu verteilen. Jedoch bieten die getätigten Experimente eine wichtige Grundlage und Referenz für weitere Experimente und Messungen, die in Zukunft durchgeführt werden können. Es konnten aus Zeitgründen keine Tests mit Bandbreitenlimite durchgeführt werden, obwohl dieses Feature implementiert wurde. Zukünftig könnte aber ein Experiment gestartet werden, worin die Knoten jeweils eine normalverteilte Bandbreitenlimite konfiguriert hätten. Würde dies aufzeigen, dass sich die Latenz bei Verdoppeln der Knoten verbessert, könnte die Hypothese 1 bestätigt werden. Die Hypothese 1 konnte bisher weder komplett bestätigt noch widerlegt werden.

Zusätzlich gäbe es einige weitere I2P-spezifische technische Ideen, die zur Verbesserung der Latenz führen könnten:

- Die ausgewählte Tunnellänge von 3 könnte eventuell auch auf Kosten der Anonymität auf 2 verringert werden. Alle Knoten die auch Teil des DIVA.EXCHANGE Netzwerkes sind, hätten standardmässig dieselbe Einstellung. Zudem wird immer ein Inbound- und ein Outbound-Tunnel miteinander verbunden, wodurch trotzdem ein hoher Anonymitätsgrad erreicht werden kann. Jedoch könnte ein Knoten, der ein Deanonymisierungsangriff durchführen will die Tunnellänge auf 0 setzen.
- Aufgrund des Verhaltens der im Garlic-Routing verwendeten Cloves wäre es auch interessant in Zukunft gleichzeitig Nachrichten über das Testnetzwerk zu versenden. Es könnte sein, dass sich die "Grundlatenz" von 5 Sekunden bildet, weil ein Router darauf wartet, dass er mehrere Nachrichten miteinander verpacken kann oder zur Verbesserung der Anonymität Nachrichten extra verzögert.
- Neben dem Socks5-Proxy, der i2pd anbietet, gibt es weitere Möglichkeiten Nachrichten über das I2P-Netzwerk zu verschicken. Unter anderem gäbe es das SAM-Protokoll („SAM V3 - I2P“, n. d.). Es könnte möglich sein, bessere Performanz mit diesem Protokoll zu erreichen, anstatt einen Socks5 oder HTTP-Proxy zu verwenden. Der Nachteil bestände jedoch darin, dass die darüberliegende Datenhaltungsschicht dieses Protokoll implementieren müsste und fest an I2P gekoppelt wäre.

## **7.1. Projektfazit**

Aus Sicht des Auftraggebers ist das Projekt ein Erfolg: Es konnten empirische Daten gesammelt werden bezüglich der Latenzzeiten von I2P und die Testinfrastruktur kann in Zukunft wiederverwendet werden, sodass nun einfacher weitere Experimente durchgeführt werden können. Auch wenn der erst eingeschlagene Lösungsweg mit NixOS nicht wie erhofft funktioniert hat, konnte trotzdem eine passende Lösung gefunden werden. Gegen Ende des Projekts wurde jedoch zu viel Zeit in die Testinfrastruktur investiert, sodass nicht mehr viel Zeit für die Messungen und zum Fertigstellen des Berichts übrig war. Dies hätte besser geplant werden sollen und es hätten früher Abstriche gemacht werden sollen.

## **7.2. Persönliches Fazit**

Als ich diese Projektidee auf der Projektschiene der Hochschule Luzern gefunden habe, habe ich mich enorm gefreut, sie als meine Bachelorarbeit auswählen zu können. Die Thematik von Anonymisierungsnetzwerken hat mich schon immer interessiert und erfüllt mich immer wieder neu mit Faszination. Die Umsetzungsarbeit war extrem interessant, auch wenn ich manchmal mit Frustration zu kämpfen hatte und ein Lösungsweg, den ich einschlägt nicht funktioniert. Auch habe ich mich oft übernommen und habe überstürzt Ideen ausprobiert. Ich hatte Mühe mit dem fertigstellen des Berichts und der Schreibarbeit und bin auch an meine Grenzen gestossen. Auch wäre ich gerne zu mehr aussagekräftigen Resultaten gekommen. Jedoch ist dies nicht immer möglich und ich musste erkennen, dass nicht alles in der gegebenen Zeit untersucht werden kann aber dass schlussendlich jedes Resultat ein gutes Resultat ist.



# Glossar

- CI** “Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.” – Fowler, 2014 . I, XI, 36
- DHT** Eine “Distributed Hashed Table” ist eine auf verschiedenen Knoten oder Rechnern verteilte Key-Value-Datenbank. . I, 7
- I2P** Dezentrales nachrichtenorientiertes Mischnetzwerk indem anonym verschlüsselte Nachrichten ausgetauscht werden können (Zantout & Haraty, 2011, p. 1). Die Abkürzung I2P wird auch als the Invisible Internet Project ausgelegt, um das gesamte Projekt nicht nur das Protokoll zu beschreiben. . I, II, 1, 2, 4, 5
- IaC** Das Verwalten und Bereitstellen von Infrastruktur wie Netzwerke, Dienste physische und virtuelle Maschinen unter Verwendung von maschinenlesbaren Definitionen oder Programmcode. Somit kann die Infrastruktur automatisiert aufgebaut werden und Versionierungssysteme können einfach eingesetzt werden. . I, 12
- Peer-to-Peer** Bei Peer-to-Peer-Netzwerk handelt es sich um eine verteilte Architektur. Alle Peers sind dabei gleichberechtigte Teilnehmer. Peers oder Netzwerkteilnehmer stellen ein Teil ihrer Ressourcen dem Netzwerk zur Verfügung. . I, 10
- TOR** Softwarepaket für anonyme Kommunikation. Es verwendet ein globales Overlay-Netzwerk bestehend aus Relays, um die Benutzer vor Netzwerküberwachung . I, 1, 4, 9
- Vollständig verteilt** Ein vollständig verteiltes Netzwerk zeichnet sich dadurch aus, dass jeder Knoten im Netzwerk komplett selbstständig ist und seine eigene Daten verwaltet. Es gibt also im klassischen Sinne keine Server oder Clients. . I, 1, 7

# Abbildungsverzeichnis

1.1	Übersicht DIVA.EXCHANGE . . . . .	2
2.1	Das Anonymitätstrilemma . . . . .	4
2.2	Garlic-Routing . . . . .	9
3.1	I2P Testnetwork . . . . .	13
4.1	CodeBerg Project Board . . . . .	18
5.1	Architektur-Diagramm . . . . .	23
5.2	Der Bootstrapping-Vorgang . . . . .	30
6.1	Nachrichtenlatenz bei Skalierung der Anzahl I2P-Knoten . . . . .	40
6.2	Nachrichtenlatenz bei verschiedener Nachrichtengrösse . . . . .	41

# Tabellenverzeichnis

4.1	Im Projekt involvierte Personen . . . . .	17
4.2	Anforderungen . . . . .	20
4.3	Projektkalender . . . . .	21
B.1	Resultate . . . . .	XII
C.1	Journal . . . . .	.XXIV

# Formelverzeichnis

3.1	Latenzberechnung . . . . .	15
-----	----------------------------	----

# Literatur

- Astolfi, F., Kroese, J. & van Oorschot, J. (2015). I2p - the invisible internet project, 5. *Bootstrapping without Reseed Servers - i2pd documentation*. (n. d.). Verfügbar 3. März 2021 unter <https://i2pd.readthedocs.io/en/latest/tutorials/floodfill-bootstrap/>
- Conrad, B. & Shirazi, F. (2014). A Survey on Tor and I2P. *Proceedings of the 9th International Conference on Internet Monitoring and Protection (ICIMP 2014)*.
- Das, D., Meiser, S., Mohammadi, E. & Kate, A. (2018). Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. *2018 IEEE Symposium on Security and Privacy (SP)*, 108–126. <https://doi.org/10.1109/SP.2018.00011>
- de Boer, T. & Breider, V. (2019, Februar). *Invisible Internet Project(Report)* (Master's Thesis). University of Amsterdam.
- Duden | Tunneling | Rechtschreibung, Bedeutung, Definition, Herkunft. (n. d.). Verfügbar 29. April 2021 unter <https://www.duden.de/rechtschreibung/Tunneling>
- Ehlert, M. (2011). I2p usability vs. tor usability a bandwidth and latency comparison [Published: Seminar, Humboldt University of Berlin], 12.
- Family configuration - i2pd documentation. (n. d.). Verfügbar 3. März 2021 unter <https://i2pd.readthedocs.io/en/latest/user-guide/family/>
- Fowler, M. (2014). *Continuous Integration*. Verfügbar 4. März 2020 unter <https://martinfowler.com/articles/continuousIntegration.html>
- Helmut Balzert, Marion Schröder & Christian Schäfer. (2017). *Wissenschaftliches Arbeiten* (2. Aufl., Bd. 1). Springer. <file:///home/mogria/Code/BA/Literature/Wissenschaftliches%20Arbeiten%20-%20balzert.pdf>
- How to run 128 (testnet) i2p routers in multiple subnets on a single linux system. [0xcc.re]. (2018, 16. Oktober). Verfügbar 3. März 2021 unter <https://0xcc.re/2018/10/16/how-to-run-128-i2p-routers-in-multiple-subnets-on-a-single-linux-system.html>
- I2P Compared to Freenet - I2P. (n. d.). Verfügbar 3. März 2021 unter <https://geti2p.net/en/comparison/freenet>
- I2P tunnels configuration - i2pd documentation. (n. d.). Verfügbar 3. März 2021 unter <https://i2pd.readthedocs.io/en/latest/user-guide/tunnels/>
- Intro - I2P. (n. d.). Verfügbar 3. März 2021 unter <https://geti2p.net/en/about/intro>
- Liu, P., Wang, L., Tan, Q., Li, Q., Wang, X. & Shi, J. (2014). Empirical measurement and analysis of i2p routers. *Journal of Networks*, 9(9), 2269–2278. <https://doi.org/10.4304/jnw.9.9.2269-2278>
- Marić, S. (2020). *Untersuchung eines vollständig dezentralen, nicht-diskriminierenden und Privatsphäre-schützenden Handelsnetzwerk für digitale Werte ("Krypto-Anlagen")* (Diss.).
- Naming and Address Book - I2P. (n. d.). Verfügbar 16. September 2021 unter <https://geti2p.net/en/docs/naming>
- The Network Database - I2P. (n. d.). Verfügbar 25. März 2021 unter <https://geti2p.net/en/docs/how/network-database>
- SAM V3 - I2P. (n. d.). Verfügbar 26. Mai 2021 unter <https://geti2p.net/en/docs/api/samv3>
- Siegfried Kaltenecker & Klaus Leopold. (2013). *Kanban in der IT* (2. Aufl.). Hanser.



- Timpanaro, J. P., Chrisment, I. & Festor, O. (2011, Oktober). *Monitoring the I2P network* [Published: Preprint]. <http://hal.inria.fr/inria-00632259>
- Tunnelimplementierung - I2P*. (n.d.). Verfügbar 15. April 2021 unter <https://geti2p.net/de/docs/tunnels/implementation>
- Zantout, B. & Haraty, R. (2011). I2P Data Communication System [event-place: St. Maarten, The Netherlands Antilles]. *Proceedings of ICN 2011, The Tenth International Conference on Networks*.

**Definitive Aufgabenstellung: Wirtschaftsprojekt / Bachelorarbeit**

**1. Starttermin:**

spätmöglicher Starttermin: HS KW 38; FS KW 8  
**22.02.2021**

**2. Abgabetermin:**

Dauer einer BAA: max. 15/16 Kalenderwochen (Um zur ordentlichen Diplomierung im Sommer zugelassen zu werden, muss die Abgabe bis spätestens Freitag, eine Woche nach Semesterende, erfolgen)  
Dauer eines WIPRO: max. 14/15 Kalenderwochen  
**04.06.2021**

**3. Studierende:**

	Student/in 1:	Student/in 2:
Name, Vorname:	Küttel, Moritz	
Studiengang:	I.BSCI.1401	
Mobile:	079 871 67 93	
E-Mail:	<a href="mailto:Moritz.kuettel@gmail.com">Moritz.kuettel@gmail.com</a>	
	<a href="mailto:Moritz.kuettel@stud.hslu.ch">Moritz.kuettel@stud.hslu.ch</a>	
Projekt mit Arbeit-geber (bb-Studierende)	<input type="checkbox"/> Ja <input checked="" type="checkbox"/> Nein	<input type="checkbox"/> Ja <input type="checkbox"/> Nein


**4. Auftraggeber/in (Rechnungsadresse):**

Firma:	Verein DIVA.EXCHANGE
Ansprechperson:	Carolyn Bächler-Schenk
Funktion:	Gründungsmitglied und Vorstandsmitglied
Strasse:	Schochenmühlestrasse 4
PLZ / Ort:	6340 Baar
Telefon:	+41 79 423 25 48
Email:	<a href="mailto:carolyn@diva.exchange">carolyn@diva.exchange</a>
Website:	<a href="https://diva.exchange">https://diva.exchange</a>

**5. Betreuungsperson/Dozent:**

Dieter Arnold, HSLU

## 6. Aufgabenstellung

Titel:	<b>Untersuchung vom I2P-Netzwerk unter dem Gesichtspunkt der Performanz und der Verwendung der freien Banking-Technologie DIVA.</b>
Ausgangslage und Problemstellung:	<p><b>Ausgangslage:</b> Das freie Software- und Netzwerkprojekt DIVA.EXCHANGE (<a href="https://diva.exchange">https://diva.exchange</a>) entwickelt den Softwareprototypen DIVA.</p> <p>Technisch besteht diese freie und quelloffene Software aus einer Anonymisierungsschicht, einer auf einer Blockchain basierenden Datenhaltung und der darauf aufbauenden Handels- und Verwaltungssoftware. Der Softwareprototyp hat den Zweck aufzuzeigen, wie die Aufbewahrung, der Handel und der Zahlungsverkehr mit digitalen Werten ganz ohne zentrale Dienstleister funktioniert – sicher und mit kompromisslosem Schutz der Privatsphäre.</p> <p>Es handelt sich um ein langfristiges Forschungsprojekt.</p> <p>Aus welchen Komponenten die Gesamtlösung besteht, kann aktuell wie folgt dargestellt werden:</p>  <p><b>Problemstellung und Annahme:</b> Das I2P Netzwerk ist zum aktuellen Zeitpunkt «langsam».</p> <p>Im Rahmen des Projektes wurde folgende Annahme formuliert: <b>eine steigende Anzahl DIVA-Installationen ist für die I2P-Netzwerk-Performanz und somit für alle Netzwerk-Teilnehmer vorteilhaft.</b></p> <p>Es ist unklar ob, in welchem Umfang und unter welchen Bedingungen die Annahme wahr ist. Darum braucht es eine Untersuchung.</p>
Ziel der Arbeit und erwartete Resultate:	<p>Das übergeordnete Ziel der Arbeit ist aufzuzeigen mit welchen Massnahmen und unter welchen Bedingungen das I2P Netzwerk für die Anwendungen von DIVA.EXCHANGE kürzere Latenzzeit hat und somit “schneller” ist für den Benutzer.</p> <p>Das I2P Netzwerk bietet Anonymität für alle Teilnehmer. Diese Anonymität ist mit Kosten verknüpft:</p> <ul style="list-style-type: none"> <li>- Der mehrfach verschlüsselte Verkehr geht über mehrere Netzwerk Knoten, wobei jeder Knoten eine Verschlüsselungsschicht entfernt. Das kostet CPU-Zyklen.</li> </ul>

	<ul style="list-style-type: none"> <li>- Die "Hops" (die Knoten, über welche der Verkehr geleitet wird) sind nicht stabil - ein Hop kann jederzeit ausfallen. Dann müssen die Pakete nochmals über eine andere Route gesendet werden. Das erhöht die Latenz enorm (Round-Trips: 3 Sekunden - 8 Sekunden, Stand heute).</li> <li>- Der I2P Verkehr wird mit Lärm angereichert (TCP Noise Protokoll), um die tatsächlich relevanten Datenpakete im Lärm zu verbergen (einfach gesagt: damit das Filtern von I2P Verkehr richtig schwierig wird). Das kostet Bandbreite.</li> </ul> <p>All diese Kosten lassen sich am besten mit der Aussage "I2P ist langsam" zusammenfassen - denn diese "Langsamkeit" ist für Entwickler und Benutzer die hauptsächliche Erfahrung.</p> <p>Das Arbeitsresultat ist die Bachelorarbeit.</p>
Gewünschte Methoden, Vorgehen:	<p>Das Projekt hat einen wissenschaftlichen Charakter basierend auf einer konkret vorliegenden Problemstellung. Das Vorgehen kann so gegliedert werden:</p> <ol style="list-style-type: none"> <li>1. Verständnis und Analyse vom I2P Netzwerk im Zusammenhang mit dem Projekt DIVA.EXCHANGE</li> <li>2. Spezifikation von Kriterien und Zielen</li> <li>3. Festlegung von geeigneten Methoden</li> <li>4. Durchführung der festgelegten Methoden</li> <li>5. Herleitung und Beschreibung der Resultate</li> </ol>
Kreativität, Varianten, Innovation*	<p>Im Rahmen des Projektes sind alle Ideen und Varianten willkommen. Da das Projekt ein wissenschaftliches Forschungsprojekt ist, hat es nichts mit dem Tagesgeschäft zu tun und bietet viel Freiraum für Kreativität.</p> <p>Gemäss Wissensstand der Auftraggeber existiert per Januar 2021 kein vollständig dezentrales und nicht-diskriminierendes Handelssystem für digitale Werte mit einer hinreichend Privatsphäre-schützenden Architektur. DIVA.EXCHANGE ist Innovation in Reinform.</p>
Schlagwörter:	Verteiltes System, dezentrales Netzwerk, Blockchain, digitale Werte, Handelssystem, I2P, Privatsphäre, Anonymität
Wirtschaftsprojekt oder Bachelorarbeit:	<input type="checkbox"/> Wirtschaftsprojekt: 180 Stunden pro Studierenden <input checked="" type="checkbox"/> Bachelorarbeit: 360 Stunden

\* Bitte heben Sie in diesem Punkt hervor, inwiefern Ihre Projektidee **über kreativen Spielraum** verfügt. Dabei sind folgende Kriterien relevant: Die Idee erlaubt den Studierenden eigene Ideen zu entwickeln und Varianten zu erarbeiten, ist ausserhalb vom Tagesgeschäft angesiedelt, beinhaltet Neuland/Innovation und ist nicht durch Produkte & Tools getrieben.

Bitte kreuzen Sie eine Projektart und die zutreffenden Schwerpunkte an.

**Projektarten:**

- ☐ Einsatz von Standardsoftware und Services
- ☐ Software- und Produkt-Entwicklung
- ☒ Innovationsprojekte (Projekte mit Erkenntnisgewinn, Forschungsprojekte)
- ☐ IT-Infrastrukturentwicklung
- ☐ Strukturierte Analyse und Konzeption von Systemen und Abläufen

**Schwerpunkte:**

- ☐ Artificial Intelligence & Machine Learning
- ☐ Business Process Modelling
- ☐ Data Science
- ☐ Hardwarenahe Software-Erstellung
- ☐ Human Computer Interaction Design
- ☐ ICT Business Solutions
- ☒ ICT Infrastrukturen
- ☐ Internet of Things
- ☐ Mobile Systems
- ☒ Security/Privacy
- ☒ Software-Erstellung

- ☐ Visual Computing (Grafik,  
Bildverarbeitung, Vision, VR, AR)  
☒ I2P Netzwerk  
☐ \_\_\_\_\_

## 7. Zeiteinteilung

Vorschlag für die Zeiteinteilung pro Person

### WIPRO:

pro Woche: ca. 12h  
für Modulendprüfung: ca. 10h  
**Total:** 180 h

### BAA:

pro Woche: ca. 20h  
Schlusswoche: ca. 50h  
Für Modulendprüfung: ca. 10h  
**Total:** 360 h

## 8. Rechtliche Grundlagen und Reglemente

Folgende Rechtsgrundlagen und Reglemente sind für die Wirtschaftsprojekte und Bachelorarbeiten an der Hochschule Luzern – Informatik massgebend:

- Studienordnung für die Ausbildung an der Hochschule Luzern, FH Zentralschweiz ([Link](#))
- Studienreglement für die Bachelor-Ausbildung an der Hochschule Luzern - Informatik ([Link](#))

## 9. Bestätigung

Mit der Kenntnisnahme der Aufgabenstellung bestätigen Student/in und Auftraggeber/in, dass

- Sie mit der Aufgabenstellung einverstanden sind.
- der Kostenbeitrag von CHF 1'000.00 (inkl. MwSt.) pro Student aufgrund der Organisationsform von DIVA.EXCHANGE als eine «non-profit Organisation» erlassen wird.
- Betreuungspersonen und Experten uneingeschränkten Einblick in die Arbeit erhalten. Auch anlässlich von Präsentationen und Marketingaktivitäten kann die Arbeit der Öffentlichkeit gezeigt werden. Eine Zusammenfassung der Arbeit wird in jedem Fall veröffentlicht. Falls das Thema vertraulich behandelt werden soll, muss der Aufgabenstellung eine entsprechende Vertraulichkeitserklärung beiliegen.

Datum: 03.03.2021

**Die definitive Aufgabenstellung (pdf-Format) bitte per E-Mail an die Transferstelle senden, zwingend in Kopie an alle involvierten Parteien.**

Anlaufstelle für alle Informationen im Zusammenhang mit studentischen Arbeiten sowie für Entgegennahme von Projektideen & Aufgabenstellungen:

Hochschule Luzern - Informatik  
Transfer Services  
Suurstoffi 1  
6343 Rotkreuz  
T: 041 228 24 66  
E: [transfer.informatik@hslu.ch](mailto:transfer.informatik@hslu.ch)

## B. Resultate

Die folgende Tabelle B.1 "Resultate" beschreibt Resultate und Unterresultate die während dieser Bachelorarbeit erarbeitet werden sollen. Diese Tabelle wurde zu Beginn des Projekts zusammengestellt anhand der Anforderungen im Abschnitt 4.2.3. Sie wurde aus Planungsgründen erstellt, um etwa einschätzen zu können, wie viel Aufwand welche Teile der Arbeit mit sich bringen und den Umfang abzugrenzen. Jedes Resultat hat einen eindeutigen Identifier in der Spalte "Nr" aufgelistet. Die Spalte "Anforderung" bezieht sich darauf, welche Anforderungen für das jeweilige Resultat massgebend sind. In der letzten Spalte "Geschätzter Arbeitsaufwand", wurde zu Beginn des Projekts abgeschätzt, wie viel Arbeitsaufwand (in Stunden) das Resultat verursacht.

Nr	Resultat	Anforderung	Geschätzter Arbeitsaufwand
D	<b>Dokumentation</b>	14-DOCS	<b>Total 44h</b>
D.1	Dokumenten Layout		8h
D.2	Aufbau des Berichts		4h
D.3	Titelseite		2h
D.4	Zusammenfassung / Abstract		2h
D.5	Einleitung		4h
D.6	Beschreibung Motivation / Problem		2h
D.7	Beschreibung der Aufgabenstellung		2h
D.8	Beschreibung der Ziele / Vision		1h
D.9	Fragestellungen / Hypothesen		2h
D.11	Reflektion / Fazit		3h
D.12	Persönliches Projektfazit		1h
D.13	Ausblick		4h
D.14	Anhang		3h
D.15	Zwischenpräsentation	16-PRES	6h
R	<b>Recherche</b>	1-SDTF 14-DOCS	<b>Total 40h</b>
R.1	Literatur sammeln (Recherche)		12h
R.2	Bibliographie erstellen		4h
R.3	P2P Networks		2h
R.3.1	- Latenz / Bandbreite / Performanz		2h
R.4	Beschreibung I2P		10h
R.4.1	– Begrifflichkeiten		4h
R.4.2	– Funktionsweise		2h
R.4.3	– Bandbreite		2h
R.4.4	– Latenz		2h
R.5	Deployment von Testnetzwerken		4h
R.6	Beschreibung der wissenschaftlichen Methode		2h

Nr	Resultat	Anforderung	Geschätzter Arbeitsaufwand
R.7	Metriken für die Auswertung	13-TPER 7-TISO 4-TREP	6h
K	<b>Testkonzept</b>	3-TKON 14-DOCS	<b>Total 46h</b>
K.1	Beschreibung Ideen / Konzepte		4h
K.2	Anforderungen an den Teststand		4h
K.3	Teststrategie		4h
K.4	Architektur Teststand		4h
K.5	Komponentendiagramm		2h
K.6	Beschreibung was gemessen werden soll		8h
K.7.1	– Bandbreite	10-TLIM	2h
K.7.1	– Anzahl Tunnels	5-TCNF	2h
K.7.1	– Latenz von Nachrichten	9-TLAT	2h
K.7.1	– Ressourcenauslastung eines Knotens	13-TPER	2h
K.8	Beschreibung wie gemessen wird		10h
K.8.1	– Isolation des Netzwerks	7-TISO	2h
K.8.1	– Verschiedene Netzwerksegmente	7-TISO	2h
K.8.2	– Latenz	9-TLAT	2h
K.8.3	– Bandbreite	10-TLIM	2h
K.8.4	– Konfigurationsmöglichkeiten	5-TCNF	2h
K.9.5	CI	11-TVRS	6h
K.9.6	Beschreibung der Auswertungsmethode		4h
I	<b>Teststand Design und Implementation</b>	2-TINF 14-DOCS	<b>Total 124h</b>
I.1	Software Design		16h
I.1	Implementation		72h
I.2.1	– Deployment des Testnetzwerkes	11-TVRS 13-TPER	8h
I.2.2	– Netzwerksegmentierung	7-TISO	8h
I.2.3	– Konfigurationsmöglichkeiten	5-TCNF	8h
I.2.4	– Skalierung	6-TSCL	8h
I.2.5	– Bandbreitenbeschränkung	10-TLIM	8h
I.2.6	– Reproduzierbarkeit	4-TREP	8h
I.2.7	– Latenzmessung	9-TLAT	8h
I.2.8	– Messung der Ressourcenauslastung	13-TPER	8h
I.2.9	– Verschiedene Testaufbauten	11-TVRS	8h
I.3	Test des Labors		24h
I.4	Handbuch für den Teststand		12h
I.4.1	– Installation		2h
I.4.3	– Konfiguration		2h
I.4.3	– Ausführen von Messungen		2h
I.4.3	– Beschreibung der gesammelten Messdaten		4h
I.4.3	– Beispiele		2h
A	<b>Messung und Auswertung</b>	12-EVAL 14-DOCS	<b>Total 52h</b>

Nr	Resultat	Anforderung	Geschätzter Arbeitsaufwand
A.1	Sammlung an Messdaten für die Auswertung		12h
A.2	Beschreibung der Auswertungsmethode		4h
A.3	Auswertung der Messungen		20h
A.3.1	– Einfluss der Knoten auf die Latenz	9-TLAT	6h
A.3.2	– Einfluss der Anzahl Verbindungen auf die Latenz	9-TLAT 10-TLIM	6h
A.3.3	– Einfluss der Bandbreite auf Latenz	9-TLAT 10-TLIM	6h
A.3.4	– Äussere Einflüsse / Unreinheiten	4-TREP 7-TISO	2h
A.4	Verschiedene Diagramme/Grafiken		8h
A.5	Auswertung der Anforderungen an den Teststand	2-TINF	4h
A.6	Zusammenfassung der Auswertung		4h
P	<b>Project Management Dokumentation</b>	14-DOCS 15-ITER	<b>Total 54h</b>
P.1	Beschreibung Projektorganisation		1h
P.3	Projektmanagement Methode		2h
P.2	Beschreibung Projektumfang		2h
P.4	Projektplanung		8h
P.5	Liste von Requirements		4h
P.6	Liste von Resultaten		4h
P.9	Arbeitsjournal		4h
P.10	Meeting-Protokolle und Notizen		29h
<b>Geschätzter Arbeitsaufwand</b>		<b>Total</b>	<b>360h</b>

Tabelle B.1.: Resultate



## C. Journal

Die folgende Tabelle C.1 "Journal" listet die geleisteten Stunden für diese Bachelorarbeit auf. Jede Zeile beinhaltet einen Task und ist unter Umständen mit einem oder mehreren Issues auf codeberg.org verbunden.

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
24-02-2021	Homeoffice	09:30	10:30	01:00	-	Kick-Off-Meeting und aufarbeiten und sammeln von Notizen	Wichtig sind im Moment die Grobplanung und die Aufgabenstellung sowie die Projektinitialisierung sowie Methoden und Tools.
24-02-2021	Im Büro	11:30	12:30	01:00	-	Repository mit Tasks und KanBan Board aufsetzen	Gehostet auf codeberg.org und beinhaltet auch ein Projektboard.
24-02-2021	Im Büro	13:30	14:30	01:00	3	Repository füllen und Bericht anhand von Template erstellen	
24-02-2021	Im Büro	13:40	15:00	00:30	3	Tasks erstellen und im Board organisieren	
24-02-2021	Im Büro	15:00	16:30	01:30	3	CI für die Repository im EnterpriseLab erstellenstellen	
24-20-2020	Im Büro	18:00	18:30	00:30	-	Mail an die Projektteilnehmer bezüglich Issue-Tracker/Board/Repo/CI	
24-20-2020	Im Büro	18:30	19:30	01:00	4 11	Meeting notes organisieren/aufarbeiten/vorbereiten in Anhang	
24-02-2021	Im Büro	19:30	21:00	01:30	3 6	Das erste Arbeitsjournal erstellen und in den Anhang einbinden	
25-02-2021	Im Büro	14:00	14:30	00:30	11 2	Reglement lesen und Notizen machen	
25-02-2021	Im Büro	14:30	15:30	01:00	11 2	Erster Wurf für die Grobplanung / Definition der Meilensteine	
25-02-2021	Im Büro	15:30	16:30	01:00	20	Dokumentation der Projektorganisation	
25-02-2021	Im Büro	16:30	17:30	01:00	11	Fragen notieren fürs Meeting / Projektdokumente sammeln	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
25-02-2021	Homeoffice	21:00	23:00	02:00	7	Bibliographie / Recherche Wissenschaftliches arbeiten / Lesen und Sammeln	
26-02-2021	Homeoffice	09:00	09:30	00:30	11	Vorbereiten fürs Meeting	Ich habe keinen Zugriff auf die aktuelle Version der Aufgabenstellung
26-02-2021	Homeoffice	09:30	10:30	01:00	-	Meeting	
26-02-2021	Homeoffice	10:30	11:00	00:30	26	Meeting Notizen verarbeiten	
26-02-2021	Homeoffice	12:30	13:00	00:30	2	Neue Issues erstellen und bestehende in Meilensteine Kategorisieren	
26-02-2021	Homeoffice	13:00	14:30	01:30	18	Notizen machen zum Aufbau vom Bericht. Buch "Wissenschaftliches Arbeiten"Lesen	
26-02-2021	Homeoffice	15:00	15:30	00:30	7	Tools installieren / Bibliographie	
26-02-2021	Homeoffice	15:30	16:00	00:30	26	Fragen zur Aufgabenstellung aufschreiben	
26-02-2021	Homeoffice	16:00	17:00	01:00	18	Aufbau des Berichts anpassen / vorbereiten fürs nächste Meeting	
26-02-2021	Homeoffice	17:30	18:00	00:30	26	Aufarbeiten der Meeting Notes / vorbereiten fürs nächste Meeting	
26-02-2021	Homeoffice	18:00	19:00	01:00	30	Provisorische liste von Requirements erstellen	Bezgl. Requirements an den Teststand ist noch nicht viel bekannt
03-03-2021	Homeoffice	09:30	10:00	00:30	-	Meeting	
03-03-2021	Homeoffice	11:00	11:30	00:30	-	Meeting Notizen aufarbeiten	
03-03-2021	Homeoffice	11:30	12:00	00:30	30	Requirement-Engineering Methode beschreiben	
03-03-2021	Homeoffice	13:00	14:00	01:00	7	Lesen / Referenzen sammeln	
03-03-2021	Homeoffice	14:00	14:30	00:30	-	Meeting Dieter	
03-03-2021	Homeoffice	15:00	16:00	01:00	7	Referenzen sammeln	
03-03-2021	Homeoffice	16:00	16:30	00:30	7	Recherche Testnetz/Teststand / Repo Einrichten	
03-03-2021	Homeoffice	17:30	18:30	01:00	30	Recherche Testnetz/ Lesen / Quellen sammeln über I2P	
03-03-2021	Homeoffice	18:30	19:30	01:00	32	Glossar hinzufügen	
03-03-2021	Homeoffice	19:30	20:00	00:30	-	Meeting Notes vom 2. Meeting verarbeiten	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
03-03-2021	Homeoffice	20:00	20:30	00:30	-	Google-Drive Share Folder einrichten mit/-für Meeting Notes und erzeuge Dokumente	Ist dafür da zum einfachen teilen/Meetings/Dokumente welche nicht in das Source-Code-Repository gehören
03-03-2021	Homeoffice	20:30	21:30	01:00	-	Backlog Grooming / Formatierung im Bericht / Journal	
04-03-2021	Homeoffice	11:00	12:00	01:00	-	Backlog Grooming / Issues erstellen	
04-03-2021	Homeoffice	14:30	15:30	01:00	19 30 32	Aufgabenstellung Anhängen / Recherche Kanban / Glossar anpassen	
04-03-2021	Homeoffice	15:30	16:30	01:00	30 9	Kanban Projektmanagement-Methode beschreiben und begründen	
04-03-2021	Homeoffice	17:00	17:30	00:30	2 24	Planung / Resultate	
04-03-2021	Homeoffice	17:30	18:30	01:00	14	i2pd auf einem Raspberry PI aufsetzen	
04-03-2021	Homeoffice	19:00	20:00	01:00	2 21	Grobplanung / Planung / Meilensteine beschreiben	
04-03-2021	Homeoffice	18:30	19:30	01:00	33	Anforderungen an den Teststand erfassen und in den Bericht einbinden.	
04-03-2021	Homeoffice	20:00	22:00	02:00	24	Resultate auflisten	
04-03-2021	Homeoffice	22:00	23:00	01:00	31	Aufwand schätzen für die Resultate	
05-03-2021	Homeoffice	09:30	10:00	00:30	-	Meeting	
05-03-2021	Homeoffice	10:00	12:00	02:00	2 39	Wochentagplan	
05-03-2021	Homeoffice	13:00	13:30	00:30	-	Meeting-Notizen	
05-03-2021	Homeoffice	14:30	15:00	00:30	21 22 2 33	Aufbau Kapital Projektmanagement / Planung / Anforderungen / Resultate	
05-03-2021	Homeoffice	15:00	16:30	01:30	10	Aufbau Kapital Einleitung / Aufgabenstellung anhängen und referenzieren	
09-03-2021	Homeoffice	18:00	20:00	02:00	14	i2pd und Tools mit Nix von Source kompilieren / Paketieren fürs PI	Wird benötigt für Eepsites
10-03-2021	Homeoffice	09:00	09:30	00:30	-	Meeting vorbereiten	
10-03-2021	Homeoffice	09:30	10:00	00:30	-	Meeting	
10-03-2021	Homeoffice	10:00	10:30	00:30	-	Meeting Notizen	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
10-03-2021	Homeoffice	12:00	13:00	01:00	-	Backlog-Grooming / Issues erstellen / Labels erstellen und zuordnen	
10-03-2021	Homeoffice	15:00	17:00	02:00	37 33	Struktur Grobkonzept	
11-03-2021	Homeoffice	15:00	16:00	01:00	7	Bibliographie / Glossar	
11-03-2021	Homeoffice	17:00	20:00	03:00	40 38	Recherche / Referenzen für Stand der Technik	
12-03-2021	Homeoffice	10:00	12:00	02:00	37	Testkonzept	
13-03-2021	Homeoffice	10:00	12:00	02:00	37	Erstellen des Netzkwerdiagrams für Testkonzept	
15-03-2021	Homeoffice	16:00	17:30	01:30	29	Struktur Kapitel Methode / Beschreibung der Verwendeten Methode	
15-03-2021	Homeoffice	18:30	22:30	04:00	36 34 35	Beschreibung Testkonzept / Ideen und Konzepte	
15-03-2021	Homeoffice	09:30	11:30	02:00	36	Beschreibung Testkonzept / Deployment	
18-03-2021	Homeoffice	09:30	11:30	02:00	47	Konfiguration Eepsite	
18-03-2021	Homeoffice	13:30	17:30	04:00	46	Ausprobieren von I2Pd-Knoten in Containern	
18-03-2021	Homeoffice	17:30	18:30	01:00	-	Beschreibung Einleitung / Aufgabenstellung	
19-03-2021	Homeoffice	08:30	09:30	01:00	-	Meeting vorbereitung / Backlog-Grooming	
19-03-2021	Homeoffice	09:30	10:30	01:00	-	Meeting / Meeting Notizen aufarbeiten	
19-03-2021	Homeoffice	10:30	12:00	01:30	-	Teststand aufbauen / i2p Konfiguration studieren	
19-03-2021	Homeoffice	13:00	14:00	01:00	49	Teststand aufbauen / Deployment mit VirtualBox	
19-03-2021	Homeoffice	14:00	17:00	03:00	49	Teststand aufbauen / Deployment Versuche / Paketieren von i2pd	
19-03-2021	Homeoffice	17:00	19:00	02:00	50	Teststand aufbauen / Erstellen der Konfiguration	
24-03-2021	Homeoffice	09:30	10:30	01:00	-	Meeting / Meeting Notizen aufarbeiten	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
24-03-2021	Homeoffice	10:30	12:00	01:30	-	Paketieren von i2p-tools für den Teststand (reseed)	
24-03-2021	Homeoffice	13:00	15:30	02:30	-	Teststand aufbauen / Konfiguration eines einzelnen Knotens	
24-03-2021	Homeoffice	15:00	17:30	02:00	-	Teststand aufbauen / Verschiedene Deploymentstrategien (AWS/Virtualbox)	
25-03-2021	Büro	09:30	12:00	02:30	50 48	Teststand Konfiguration von Anzahl Knoten pro Testlauf	
25-03-2021	Büro	13:00	15:00	02:00	50 48	Teststand Konfiguration von Anzahl Knoten /	
25-03-2021	Büro	15:00	18:30	03:30	51	i2p-tools paketieren / Reseeder Knoten erstellen	
25-03-2021	Homeoffice	22:00	23:00	00:30	-	Meeting Vorbereitung	
25-03-2021	Büro	09:30	10:30	01:00	-	Meeting / Notizen	
25-03-2021	Büro	10:30	12:00	01:30	52	Softwarepakete konfigurieren / Stabile und kompatible Pakete auswählen	
25-03-2021	Büro	13:00	18:30	05:30	52	Netzwerkkonfiguration und Isolierung der Knoten	Mühsam IPv4 zu konfigurieren / Netzwerkkonfiguration von Containern nicht gut unterstützt
28-03-2021	Homeoffice	17:00	19:00	02:00	40 38	Stand der Technik / Beschreibung Tunnels / Anonymitäts-Trilemma	
28-03-2021	Homeoffice	19:00	21:00	02:00	59	Kapitel Einleitung überarbeiten / Besser an Aufgabenstellung anpassen	
28-03-2021	Homeoffice	21:00	22:00	01:00	37	Anforderungen / Testkonzept überarbeiten	
31-03-2021	Büro	10:30	12:00	01:30	52	Container-Netzwerkkonfiguration	Klappt nicht mit IPv6/IPv4 / Immer falsche Netzwerkmaske mit NixOS/nspawn Containern
31-03-2021	Büro	13:00	13:30	00:30	-	Meeting vorbereitung	
31-03-2021	Büro	13:30	14:30	01:00	-	Meeting	
31-03-2021	Büro	15:00	18:30	03:30	52	Container-Netzwerkkonfiguration / Ausschau nach alternativer Container-Technologie	Kubernetes? docker-compose?

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
01-04-2021	Büro	10:30	12:00	01:30	52	Container-Netzwerkkonfiguration / weitere Versuche	
01-04-2021	Büro	13:00	17:00	04:00	52	Container-Netzwerkkonfiguration / Aus-schau nach alternativer Container-Technologie	
07-04-2021	Büro	09:00	10:00	01:00	-	Pinning einer Nixpkgs Version für Repro-duzierbarkeit	
07-04-2021	Büro	10:00	11:30	01:30	52	Korrektur des NixOS-Container-Moduls bezüglich Routen	
07-04-2021	Büro	12:30	16:00	03:30	52	Korrektur des NixOS-Container-Moduls bezüglich Adressen / Routen / Netzwerk-maske	Die Container können nun endlich mitein-ander kommunizieren. Aber Nix braucht aber enorme Mengen an RAM für die Erstellung der Container-Konfigurationen (bevor dem deployment!). Dies ist nicht skalierbar auf mehr als ca. 30 Container. Ein anderer Lösungsansatz muss her.
07-04-2021	Büro	16:00	18:30	02:30	36 48	Grundsätzliche Umstellung von NixOS-Container nach docker-compose	Keine Erfahrung mit Kubernetes / wahr-scheinlich too much
08-04-2021	Büro	09:30	11:30	02:00	49 50 48 36	Skript zum automatischen Auf- und Abbau des docker-compose Netzwerks	
08-04-2021	Büro	12:00	16:00	04:00	51	Netzwerkbootstrapping / Reseeder mit Router Infos von Knoten initialisieren	
08-04-2021	Büro	16:00	17:00	01:00	51	Erstellung einer eigenen Docker-Container für den Reseeder	
08-04-2021	Büro	17:00	18:30	01:30	51	Erstellung einer eigenen Docker-Container für den I2Pd-Knoten	
09-04-2021	Büro	09:30	11:30	02:00	51	Problembhebung HTTPS-Verbindung mit Reseed-Server / Zertifikathandhabung	Erstes erfolgreiches Bootstrapping mit ei-genem Reseed Server im Testnetzwerk!
09-04-2021	Büro	12:30	16:00	03:30	51	Beschreibung und Diagram über den Bootstrapping-Vorgang	
09-04-2021	Büro	16:00	17:30	01:30	40	Einleitung / Stand der Technik	
14-04-2021	Büro	13:30	14:30	01:00	46	Umstrukturierung der Repo / entfernen von NixOS-Container	
14-04-2021	Büro	14:30	16:00	02:00	35	Überlegungen und erste Versuche für Test-nachrichten im Netzwerk	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
14-04-2021	Büro	16:00	17:00	01:00	-	Meeting	
14-04-2021	Büro	17:00	19:30	02:30	57	Beginn implementation eines TCP clients mit Socks5-Support	Socks5-Proxy-Support benötigt um TCP-Nachrichten über das I2P-Overlay-Netzwerk zu verschicken
14-04-2021	Büro	19:30	20:30	01:00	35 47 56	Beginn Implementation Servers der Latenz von Nachrichten misst und loggt	Benötigt wenn Knoten mehrere Verbindungen annehmen will
15-04-2021	Büro	10:30	13:00	01:30	56	Implementation des Servers der Latenz von Nachrichten misst und loggt	
15-04-2021	Büro	13:30	15:00	01:30	56	Implementation des Servers der Latenz von Nachrichten misst und loggt	
16-04-2021	Büro	10:30	12:30	01:30	56	Erste Implementation Servers der Latenz von Nachrichten misst und loggt	
16-04-2021	Büro	13:00	16:30	03:30	56	Erste Implementation Servers der Latenz von Nachrichten misst und loggt	
16-04-2021	Büro	19:00	20:30	01:30	52	Debugging wieso Knoten im privaten Netzwerk nicht kommunizieren können	Private Adressen werden nicht angenommen von i2pd um Tunnels aufzubauen
18-04-2021	Homeoffice	14:00	17:00	03:00	52	Abändern von i2pd das private Adressen zugelassen werden / weiteres Debugging	
18-04-2021	Homeoffice	18:00	23:00	05:00	52 48	Debugging / diverse versuche mit docker-compose Netzwerkkonfiguration	Knoten können nicht miteinander reden. Es scheint das docker-compose –scale ist broken dies bezüglich.
19-04-2021	Homeoffice	12:00	18:00	06:00	23	Zusammenstellen der Zwischenpräsentation	
20-04-2021	Homeoffice	22:00	24:00	02:00	23	Fertigstellung der Zwischenpräsentation	
21-04-2021	Büro	10:30	11:30	01:00	-	Aufbereiten Meeting Notes	
21-04-2021	Büro	12:00	13:00	01:00	39	Aufbereiten Meeting Notes / Planung	
21-04-2021	Büro	13:00	14:00	01:00	-	Vorbereitung Präsentation	
21-04-2021	Büro	14:00	15:00	01:00	-	Zwischenpräsentation	
21-04-2021	Büro	15:00	18:30	03:30	52 49 50 48	Generieren von docker-compose Konfigurationen für i2pd / Tests	Zum Lösen der Netzwerkkonfigurationsprobleme
22-04-2021	Büro	10:30	11:30	01:00	48	Fertigstellung der generierten docker-compose Konfigurationen für i2pd	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
22-04-2021	Büro	12:00	14:00	02:00	52	Erstellen von Netzwerktests / lösen der Netzwerkkonfigurationsprobleme	
22-04-2021	Büro	14:00	15:00	01:00	52	Erste Kommunikation über das private I2P-Overlaynetzwerk / Erstellen eines Tests	
22-04-2021	Büro	15:00	16:30	01:30	56 57	TCP client und server implementation / hinzufügen zum i2pd-Container	
23-04-2021	Büro	12:30	14:30	02:00	-	Backlog-Grooming / Planung	
23-04-2021	Büro	14:30	15:30	01:00	-	Beschreibung Realisierung / Netzwerkprobleme / Bootstrapping	
23-04-2021	Büro	15:30	18:30	03:00	-	Erstellung Architekturdiagramm	
28-04-2021	Büro	11:00	12:00	01:00	-	Architekturdiagramm	
28-04-2021	Büro	13:00	15:30	01:30	-	Architekturdiagramm Beschreibung Realisierung	
28-04-2021	Büro	15:30	16:30	01:00	-	Meeting	
28-04-2021	Büro	16:30	18:00	01:30	-	Architekturdiagramm / Beschreibung Architektur	
29-04-2021	Büro	10:00	11:30	01:30	-	Beschreibung Client / Server-Tunnels	
29-04-2021	Büro	12:30	18:00	05:30	-	Beschreibung I2P Stand der Technik / NetDb	
30-04-2021	Büro	10:30	11:30	01:30	-	Beschreibung I2P Stand der Technik / Router Info / Lease Sets	
30-04-2021	Büro	11:30	16:00	03:30	-	Beschreibung I2P Stand der Technik / LeaseSets / Tunnels	
05-05-2021	Büro	11:00	12:30	01:30	-	Separate Volumes für die I2Pd Container / Committen	
05-05-2021	Büro	13:00	14:00	01:00	-	Vorbereitung Meeting / Meeting	
05-05-2021	Büro	14:00	17:00	03:00	-	TCP-Server Ankunftszeit messen und in ein Logfile schreiben	
05-05-2021	Büro	17:00	20:00	03:00	-	Client Sendezeit messen und über socks5 an server senden	
06-05-2021	Büro	10:30	12:30	02:00	-	Adressbuch generieren für einfacheres Mess-Skript	



Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
06-05-2021	Büro	13:00	15:00	02:00	-	TCP Server und Client in i2pd-Container einbinden / Docker-Volume für Messdaten	Bereinigen von Berechtigungsproblemen mit dem Volume
06-05-2021	Büro	15:00	17:30	02:30	-	Anpassen des Messskripts zur Verwendung des TCP-Servers / Erste Messversuche	Probleme mit der Stabilität der Messungen
06-05-2021	Büro	17:30	18:30	02:00	-	Meeting mit Konrad / Debugging der Messprobleme	
07-05-2021	Büro	10:00	11:30	01:30	-	Stabilität von Bootstrapping mit Volumes / mehrfaches aufstarten	
07-05-2021	Büro	12:30	15:00	02:30	-	Stabilität mehrfaches Aufstarten / Erster Messvorgang starten	
07-05-2021	Büro	15:00	17:00	02:00	-	Beschreibung von Konfigurationsoptionen und Nachrichtenformat	
11-05-2021	Büro	11:00	12:30	01:30	-	Stabilität von Bootstrapping mit Volumes und erneutes Aufstarten	
11-05-2021	Büro	12:30	14:00	01:30	-	Umstrukturierung Realisierungsteil / Beschreibung TCP-Client	
11-05-2021	Büro	14:00	15:00	01:00	-	Beschreibung Adressbuch / Realisierung	
11-05-2021	Büro	15:00	17:00	02:00	-	Beschreibung Nachrichten / Realisierung	
11-05-2021	Büro	22:00	23:00	02:00	-	Ausprobieren mit 256 / 128 / 64 Knoten / Berechnung starten mit 32 Knoten	Die Knoten können nicht kommunizieren bei einem 64 Knoten Netzwerk
12-05-2021	Büro	10:30	11:30	01:00	-	Problembhebung mit mehr Anzahl Knoten / Knoten in zufällig verzögert starten	Löst das grundsätzliche Problem nicht
12-05-2021	Büro	12:30	15:30	01:00	-	Problembhebung mit mehr Anzahl Knoten / ARP-Netzwerkeinstellungen	ARP-Neighbor Table zu klein
12-05-2021	Büro	15:30	16:30	01:00	-	Meeting	
12-05-2021	Büro	16:30	18:30	02:00	-	Meeting-Notizen aufarbeiten	
13-05-2021	Büro	11:00	12:00	01:00	-	Bessere Beschreibung der Router-Infos	
13-05-2021	Büro	13:00	14:00	01:00	-	Überarbeitung der Einleitung	
13-05-2021	Büro	13:00	16:00	02:00	-	Überarbeitung Stand der Technik	
13-05-2021	Büro	16:00	18:00	02:00	-	Beschreibung des Messskripts	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
14-05-2021	Büro	11:00	12:30	01:30	-	Beschreibung Realisierung / Projektmanagement	
17-05-2021	Homeoffice	10:00	12:00	02:00	-	Konfigurationsoption für die i2pd-Konfigurationsdatei implementieren	
17-05-2021	Homeoffice	13:00	15:00	02:00	-	Konfigurationsoption für verschiedene Bandbreiten der i2pd-Knoten implementieren	
17-05-2021	Homeoffice	15:00	16:00	01:00	-	Auswertung: Beschreibung der Testschritte	
18-05-2021	Homeoffice	14:00	15:00	01:00	-	Meeting	
18-05-2021	Homeoffice	15:00	18:00	03:00	-	Setup der Finalen Mess-VM auf Hetzner	
19-05-2021	Homeoffice	15:00	18:00	03:00	-	Durchführen von Messungen	
19-05-2021	Homeoffice	15:00	18:00	03:00	-	Setup der Finalen Mess-VM auf Hetzner	
23-05-2021	Homeoffice	15:00	17:00	02:00	-	Beschreibung DIVA-Blockchain / Stand der Technik	
25-05-2021	Büro	09:00	11:30	01:30	-	Korrekturen Web-Abstract / Einleitung	
25-05-2021	Büro	12:00	14:00	02:00	-	Korrekturen Web-Abstract / Einleitung	
25-05-2021	Büro	14:00	18:00	04:00	-	Beschreibung erster Lösungsansatz NixOS-Container	
26-05-2021	Büro	09:00	11:30	01:30	-	Korrekturen Web-Abstract / Einleitung	
26-05-2021	Büro	12:00	14:00	02:00	-	Korrekturen Web-Abstract / Einleitung	
26-05-2021	Büro	14:00	18:00	04:00	-	Beschreibung erster Lösungsansatz NixOS-Container	
26-05-2021	Homeoffice	20:00	22:00	02:00	-	Überarbeitung und Korrektur Web-Abstract	
27-05-2021	Büro	09:00	13:00	01:00	-	Korrekturen Web-Abstract / Einleitung	
27-05-2021	Büro	10:00	11:30	01:30	-	Korrekturen Web-Abstract / Einleitung	
27-05-2021	Büro	12:00	15:30	03:30	-	Auswerten und Zusammenstellen der Messungen / provisorisches Diagramm erstellen	
27-05-2021	Büro	15:30	17:00	01:00	-	Meeting / Review Web Abstract / Korrektur	
27-05-2021	Homeoffice	20:30	22:00	01:30	-	Überarbeitung und Korrektur Web-Abstract	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
28-05-2021	Büro	09:00	10:00	01:00	-	Korrekturen Web-Abstract / Einleitung	
01-09-2021	Rotkreuz	09:30	10:00	01:00	-	Korrekturen Web-Abstract / Einleitung	
01-09-2021	Rotkreuz	13:30	14:30	01:00	-	Meeting zum Abschluss Vorgehen zum Abschluss der Arbeit	
01-09-2021	Rotkreuz	14:30	17:30	03:00	-	Erstellung einer Grobplanung für die letzten beiden Wochen	
02-09-2021	Rotkreuz	09:30	11:30	02:00	-	Überarbeitung Architekturdiagramm	
02-09-2021	Rotkreuz	12:30	17:30	05:00	-	Überarbeitung Architekturdiagramm	
06-09-2021	Rotkreuz	09:30	11:30	02:00	-	Überarbeitung Systemarchitektur	
06-09-2021	Rotkreuz	12:30	17:30	05:00	-	Umsetzung / Realisierung	
07-09-2021	Rotkreuz	09:30	11:30	02:00	-	Umsetzung / Realisierung / Beschrieb der docker-compose Lösung	
07-09-2021	Rotkreuz	12:30	17:30	05:00	-	Beschrieb Probleme bei der Skalierung der Knoten	
08-09-2021	Rotkreuz	09:30	11:30	02:00	-	Realisierung / Komponentendesign	
08-09-2021	Rotkreuz	12:30	17:30	05:00	-	Korrekturen Web-Abstract / Einleitung	
09-09-2021	Rotkreuz	09:30	11:30	02:00	-	Stand der Technik	
09-09-2021	Rotkreuz	12:30	15:30	03:00	-	Korrekturen Web-Abstract / Einleitung	
10-09-2021	Rotkreuz	09:30	11:30	02:00	-	Evaluation und Validation Kapitel erstellen	
10-09-2021	Rotkreuz	12:30	17:30	05:00	-	Auswerten der Messungen	
10-09-2021	Rotkreuz	19:00	22:00	03:00	-	Erstellung des Pitching Videos	
13-09-2021	Rotkreuz	09:30	11:30	02:00	-	Auswerten der Messungen	
13-09-2021	Rotkreuz	12:30	17:30	05:00	-	Auswerten der Messungen / Evaluation und Validation	
15-09-2021	Rotkreuz	09:30	11:30	02:00	-	Evaluation und Validation / Ausblick / Abstract	
15-09-2021	Rotkreuz	12:30	18:30	06:00	-	Evaluation und Validation / Ausblick	
16-09-2021	Rotkreuz	09:30	11:30	02:00	-	Diverse Korrekturen	
16-09-2021	Rotkreuz	12:30	18:30	05:00	-	Fertigstellung Stand der Technik	

Datum	Ort	Von	Bis	Stunden	Issue #	Task	Notizen
17-09-2021	Rotkreuz	10:30	12:30	02:00	-	Korrekturen Stand der Technik / Garlic-Routing	
17-09-2021	Büro ZHz	15:30	20:00	04:30	-	Korrekturen Stand der Technik / Testnetzwerke und Bootstrapping	
18-09-2021	Homeoffice	04:00	07:00	03:00	-	Korrekturen und Überarbeitung Kapitel Methode	
18-09-2021	Homeoffice	12:00	20:00	08:00	-	Letzte Korrekturen / Verdankung / Realisierung / Ideen und Konzepte	
19-09-2021	Homeoffice	13:00	18:00	05:00	-	Letzte Korrekturen und finales Layouting	
		<b>Total (Ist)</b>	<b>hours</b>	388:00			
		<b>Total (Soll)</b>	<b>hours</b>	360:00			

Tabelle C.1.: Journal