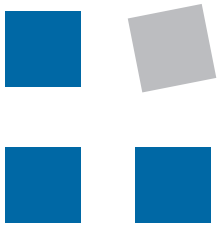

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Studienarbeit

Multipath over Wireless Networks für mobiles WiFi

Autoren:
Nils Caspar
Simon Huber

Betreuer:
Prof. Beat Stettler

19. Dezember 2013

Erklärung der Eigenständigkeit

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Rapperswil, 19. Dezember 2013

Nils Caspar

Simon Huber

Inhaltsverzeichnis

1 Abstract

Unternehmen erkennen zunehmend das Bedürfnis der Gesellschaft, auch unterwegs auf das Internet zugreifen zu können. So bieten zum Beispiel verschiedene Transportunternehmen ihren Kunden bereits kostenlosen Internetzugang via WLAN an.

Bislang kamen für die Anbindung ans Internet zwar häufig mehrere Mobilfunkanbieter zum Einsatz (Multihoming), die dynamische Verteilung des Datenverkehrs auf verschiedene Pfade (Multipathing) war aber nur beschränkt möglich.

Seit Oktober 2009 ist eine Arbeitsgruppe der Internet Engineering Task Force (IETF) damit beschäftigt, einen offenen Standard zu spezifizieren, der genau dieses Problem lösen soll: Die Verteilung eines Datenstroms über mehrere Pfade mittels MultiPath TCP (MPTCP).

Im Rahmen dieser Arbeit wurden verschiedene Multipathing-Lösungen untersucht und evaluiert. Basierend auf den Auswertungen dieser Evaluation wurde ein System entworfen, welches auf Basis von MPTCP die Nutzung aller zur Verfügung stehender Pfade ermöglicht.

Die Lösung wurde als Prototyp implementiert und ausführlich getestet. Mit dem Prototyp konnte die Ausfallsicherheit erhöht und die theoretisch zur Verfügung stehende Bandbreite zu 90% ausgenutzt werden. Um zu erreichen, dass die Lösung für den Endbenutzer vollständig transparent ist, wurde eine Multipath-fähige Middlebox eingesetzt.

Eine Schwierigkeit bestand darin, sämtlichen Traffic (TCP, UDP, ICMP etc.) über mehrere Internet-Uplinks zu übertragen. Dies war besonders deshalb eine Herausforderung, weil TCP-over-TCP Performanceeinbussen mit sich bringt. Durch den Einsatz einer Traffic-Weiche, welche den TCP-Traffic via SOCKS-Proxy und den restlichen Traffic via VPN-Tunnel weiterleitet, konnte dieses Problem gelöst werden.

2 Management Summary

2.1 Ausgangslage

Bereits heute bietet die PostAuto Schweiz AG kostenlosen Internetzugang via Wireless-LAN (WLAN) in einem Grossteil ihrer Fahrzeuge an. Die eigentliche Datenübertragung findet dabei über das Mobilfunknetz mittels UMTS-Aussenantennen statt. Funklöcher und überlastete Zellen oder auch Netzwechsel führen zu einem Unterbruch und beeinträchtigen so die Nutzung. Um diese Nachteile zu vermindern und dadurch letztendlich das Benutzer-Erlebnis zu verbessern, soll der Verkehr dynamisch auf verschiedene Mobilfunknetze aufgeteilt werden. Konkret bedeutete das, dass mehrere Mobilfunknetze gleichzeitig verwendet werden. Da selten alle Provider am selben Ort Funklöcher haben, erhöht sich dadurch die Ausfallsicherheit erheblich. Ausserdem würde sich die Geschwindigkeit vervielfachen, da die Bandbreiten der verschiedenen Mobilfunkprovider gleichzeitig genutzt und somit summiert werden können.

2.2 Vorgehen, Technologien

In einem ersten Schritt mussten die Schwierigkeiten beim Multipathing, also dem gleichzeitigen Nutzen von Netzen verschiedener Provider, verstanden werden. Weiter wurden existierende Technologien und Lösungsansätze analysiert und mit den Anforderungen abgeglichen und ihre Tauglichkeit bewertet. Für den experimentellen Standard MPTCP wurden weitere Analysen vorgenommen und Abklärungen getroffen. Das MPTCP-Projekt ist zwar noch ziemlich jung, scheint die gestellten Anforderungen aber am besten zu erfüllen.

MPTCP muss grundsätzlich von beiden Kommunikationsteilnehmern (Endgerät und Server) unterstützt werden. Da der Standard sich erst in der Entwicklung befindet, ist das heute noch nicht der Fall.
Der in dieser Arbeit konzipierte Ansatz beruht stattdessen darauf, dass eine Umleitung

über einen Relay-Server gemacht wird, der die besagte Technologie beherrscht und als Vermittler zwischen den beiden Parteien auftritt. Somit kann der Luftweg über MPTCP zurückgelegt werden, der Weg vom Relay-Server bis zum Webserver erfolgt wieder über einen einzigen, stabilen Pfad. Für den Benutzer ist diese Umleitung transparent, also nicht bemerkbar.

2.3 Ergebnisse

Auf Basis der Analysen und Untersuchungen wurde ein funktionierender Prototyp implementiert. Der Fokus dieser Arbeit lag auf der Untersuchung und Erarbeitung eines grundlegenden Konzepts. Deshalb wurde für die Beispiel-Implementation eine Vereinfachung der Hardware vorgenommen: Der Prototyp besteht aus einem Laptop mit mehreren USB-Modems (entspricht den UMTS-Aussenantennen), welche mit verschiedenen Mobilfunkprovidern verbunden sind.

Tests bestätigten die Performance-Steigerung sowie unterbrechungsfreie Übergänge bei Ausfällen einzelner Mobilfunkverbindungen. Die theoretisch zur Verfügung stehende Bandbreite konnte zu ca. 90% ausgenutzt werden.

2.4 Ausblick

Die Arbeit kann als Grundlage für eine Implementation im produktiven Einsatz genutzt werden. Eine Angleichung der Konfiguration an das Produktiv-System wäre notwendig. So müsste beispielsweise die Methode zur Ermittlung der Signalqualität an die Schnittstelle des Produktiv-Systems angepasst werden. Das grundlegende Konzept könnte aber übernommen werden.

Die Ziele der Arbeit wurden erreicht. Zusätzlich wurden Pläne für die Skalierung, beispielsweise durch den Einsatz von Cluster-Lösungen, formuliert.

Ein Ausbau, wie beispielsweise die Umschaltung gemäss Signalqualität, könnte mit wenig Aufwand vorgenommen werden, da die entsprechenden Schnittstellen dokumentiert und die Ideen ausformuliert wurden.

Bei der Implementation des Prototyps fielen noch einige Schwierigkeiten und Software-Fehler in der MPTCP-Implementation auf. Eine ausgereiftere Version von MPTCP wäre für ein Produktiv-System vonnöten.

Teil I

Technischer Bericht

3 Anforderungsspezifikation

3.1 Grossräumige Zielsetzung, Motivation

In modernen öffentlichen Verkehrsmitteln (z.B. Postautos) wird oft eine Internetverbindung via WLAN angeboten. Üblicherweise wird der Internetverkehr dabei über Aussenantennen an einen Mobilfunkbetreiber weitergeleitet. Die Nutzleistung wird jedoch durch Funklöcher und Kapazitätsengpässe eingeschränkt. Es wäre deshalb wünschenswert, die Netze verschiedener Mobilfunkbetreiber gleichzeitig nutzen zu können und den Verkehr dynamisch auf diese aufzuteilen.

3.1.1 Dynamische Verteilung des Verkehrs

Der Verkehr soll dynamisch auf verschiedene Mobilfunknetze aufgeteilt werden. Eine gute, stabile, schnelle Verbindung soll bevorzugt werden. Bei der Bestimmung der Verbindungsqualität soll möglichst wenig Overhead generiert werden.

3.1.2 Aufrechterhaltung

Auch bei Netzwechsel müssen die Verbindungen aufrechterhalten bleiben. Der Benutzer soll durch allfällige Mobilfunknetz- und dadurch bedingte IP-Wechsel nicht beeinträgtigt werden.

3.1.3 Skalierbarkeit

Die Lösung muss skalierbar sein. Wird der Traffic beispielsweise über eine zentrale Stelle geleitet, müsste die Lösung eine Form von Loadbalancing vorsehen.

3.1.4 Network Address and Port Translation

Es muss beachtet werden, dass die Mobilfunkbetreiber Network Address and Port Translation (NAPT) einsetzen. Es werden üblicherweise keine öffentlichen IP-Adressen vergeben. Den via WLAN verbundenen Endgeräten wird auch wiederum eine IP aus einem (anderen) privaten Subnet vergeben.

3.2 Systemgrenzen

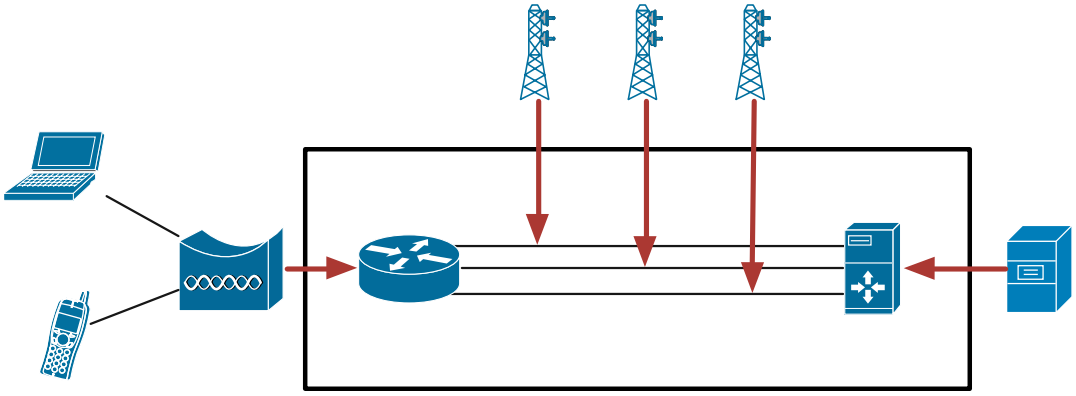

Abbildung 3.1: Systemgrenzen unter Verwendung eines Relay-Server

Das in Abbildung 3.1 dargestellte Gesamtsystem definiert sich aus Komponenten, die in- oder ausserhalb der Systemgrenze liegen, je nachdem ob sie konfigurierter Teil der Arbeit sind oder nicht. Damit werden die Grenzen dieser Arbeit festgelegt. Alle Komponenten haben eine Wechselwirkung mit dem System, d.h. sie agieren bzw. reagieren auf das System.

Mit "konfigurieren" ist in diesem Kontext gemeint, dass sie aktiv von der Arbeit verändert werden. Beispielsweise das Gerät des Endbenutzers oder die Mobilfunknetzantennen bleiben unberührt, d.h. es wird keine Konfiguration auf diesen Komponenten vorgenommen oder Software installiert. Im Gegensatz dazu sind die Komponenten innerhalb der Systemgrenze von der Arbeit veränderbare Bestandteile.

Der Endbenutzer, welcher mit einem mobilen Endgerät das zur Verfügung gestellte WLAN benutzt, liegt ausserhalb der Systemgrenze. Der Benutzer beeinflusst unser System mit Traffic, den er verursacht und wird vom System mit der Antwort beeinflusst.

Er kommuniziert über WLAN mit dem Multifunktions-Wi-Fi-Router. Nebst dem ei-

gentlichen Router beinhaltet das Gerät beispielsweise einen DHCP-Server sowie eine Wireless-Bridge. Die Systemgrenze befindet sich in diesem Gerät, das heisst innerhalb des Embedded-Systems. Auch die Wireless-Bridge, Mobilfunk-Antennen sowie der ganze Verkehr der Mobilfunkanbieter liegen ausserhalb der Systemgrenze.

Da nicht erwartet werden kann, dass alle Zielserver die angestrebte Multipath-Fähigkeit unterstützen, muss ein Relay-Server eingesetzt werden. Der Relay-Server gemäss Abbildung 3.1 beherrscht diese Multipath-Fähigkeit. Er tritt als Vermittler zwischen dem Multifunktions-Wi-Fi-Router und dem Zielserver auf.

Dabei werden nur der Router und der Relay-Server zu unserem System und damit dem Wirkungsbereich dieser Arbeit gezählt.

3.3 Aktoren

Das System hat zwei Aktoren:

- Traffic

  Das System wird durch den Traffic beeinflusst. Die End-User verursachen eine gewisse Menge Traffic die verarbeitet werden muss.

- Mobilfunknetz

  Je nach Standort sind verschiedene Links vorhanden. Dies hängt von der Abdeckung der Mobilfunknetze und deren Technologien ab.

3.4 User Stories

1. Als Passagier möchte ich während meiner Reise Skype mit guter Qualität nutzen, um mit meinem Geschäftspartner ein wichtiges Anliegen zu besprechen.
2. Als Passagier möchte ich eine grössere Datei via FTP herunterladen.
3. Als Passagier möchte ich eine stabile VPN-Verbindung, um geschäftliche Arbeiten erledigen.
4. Als Passagier möchte ich eine stabile Verbindung und von allfälligen Interface-Wechseln nichts merken.
5. Als Passagier möchte ich auch eine gute Verbindung haben, wenn viele Passagiere gleichzeitig surfen.
6. Als Passagier möchte unabhängig vom Standort die optimale Verbindung haben.
7. Als Anbieter möchte ich eine möglichst skalierbare Lösung.
8. Als Anbieter möchte ich Passagiere, die mit der Internetverbindung möglichst zufrieden sind.

3.5 Anwendungsfälle

3.5.1 UC1: Traffic optimal auf die verfügbaren Links aufteilen

Primary Actor:	• Traffic
Stakeholders and Interests:	• Endbenutzer möchte den optimalen Throughput • Anbieter: Möchte alle verfügbaren Links optimal nutzen
Preconditions:	• mindestens ein Link verfügbar • es ist Traffic vorhanden • die Werte aus "UC2: Erkennen von Links" sind vorhanden
Postconditions:	• Die verfügbaren Links werden optimal ausgelastet

Main Success Scenario:

1. Traffic trifft beim System ein
2. System verteilt den Traffic auf alle verfügbaren Links
3. System sendet bevorzugt über schnelle Links
4. Nach einer gewissen Zeit ist nach "best effort" der maximale Throughput erreicht.

Alternate Flow:

4.a System verteilt den Traffic nach Messungen

[Multipath-Technologie kann den Throughput nicht direkt auf dem Paket auslesen]

1. System misst den Throughput auf jedem Link
2. Die Messung wird dem System zur Verfügung gestellt
3. Traffic wird proportional zu Throughput gemäss Messungen auf die verfügbaren Links verteilt

4.b System verteilt den Traffic nach Priorität

[Mobilfunknetz-Antennen-Adapter-Treiber kann den verwendeten Mobilfunkstandard auslesen]

1. System liest über den Treiber des Mobilfunknetz-Antennen-Adapters die verwendete Interface-Technologie (Enhanced Data Rates for GSM Evolution (EDGE), Universal Mobile Telecommunications System (UMTS), Long Term Evolution (LTE), WLAN) und die Link-Signalstärke aus
2. Traffic wird bevorzugt auf schnelle Interfaces geschickt, sobald die Messungen verfügbar sind.

3.5.2 UC2: Erkennen von Links

Primary Actor:	• Mobilfunknetz
Stakeholders and Interests:	• Endbenutzer möchte den optimalen Throughput • Anbieter: Möchte zu jeder Zeit wissen, welcher Link vorhanden ist
Postconditions:	• System kennt die verfügbaren Links

Main Success Scenario:

1. Auslesen der verfügbaren Links zum aktuellen Zeitpunkt
2. System kennt die verfügbaren Links

3.6 Nicht-funktionale Anforderungen

3.6.1 Übertragbarkeit

Router

Die Lösung muss mit dem Router kompatibel sein, der bereits in den Postautos vorhanden ist. Es handelt sich um einen Multifunktions-Wi-Fi-Router.

Skalierbarkeit

Im Endausbau ist mit ca. 1'500 Fahrzeugen zu rechnen. Diese mindestens zwei Verbindungen. Der Relay-Server muss demnach mindestens 3'000 Verbindungen behandeln können. Pro Fahrzeug wird mit maximal 30 mit dem Multifunktions-Wi-Fi-Router verbundenen Endgeräten gerechnet. Geht man von durchschnittlich 3 Verbindungen pro Gerät aus, entspricht das maximal 90 gleichzeitigen Verbindungen. Generell folgt die Anforderung, dass der Server skalierbar sein soll, damit die Verwendung der Lösung auch für andere Bereiche mit noch mehr Verbindungen eingesetzt werden könnte.

3.6.2 Effizienz

Traffic-Overhead

Der Traffic-Overhead soll so gering wie möglich gehalten werden. Idealerweise kann die Messung der Verbindungsgeschwindigkeit mittels Analyse des sowieso vorhandenen Datenverkehrs vorgenommen werden.

3.6.3 Zuverlässigkeit

Redundanz

Fällt ein Link aus, so soll der Traffic auf die verbleibenden Links aufgeteilt werden. Der Benutzer soll den Ausfall nicht spüren, d.h. die bestehenden Flows sollen nicht unterbrochen werden.

Belastbarkeit

Die Lösung muss bei jeder Auslastung, auch bei vollständiger Belastung der Leitung einwandfrei funktionieren.

4 Analyse

4.1 Komponenten

Der vorgegebene Multifunktions-Wi-Fi-Router (vgl. Abschnitt 3.6.1) besteht aus den folgenden Komponenten: *Wireless-Bridge*, *Switch*, *Router*, *Server* und dazugehörigen Diensten (Dynamic Host Configuration Protocol (DHCP), möglicherweise VPN etc.) sowie mehreren *Modems* (LTE, UMTS, General Packet Radio Service (GPRS)).

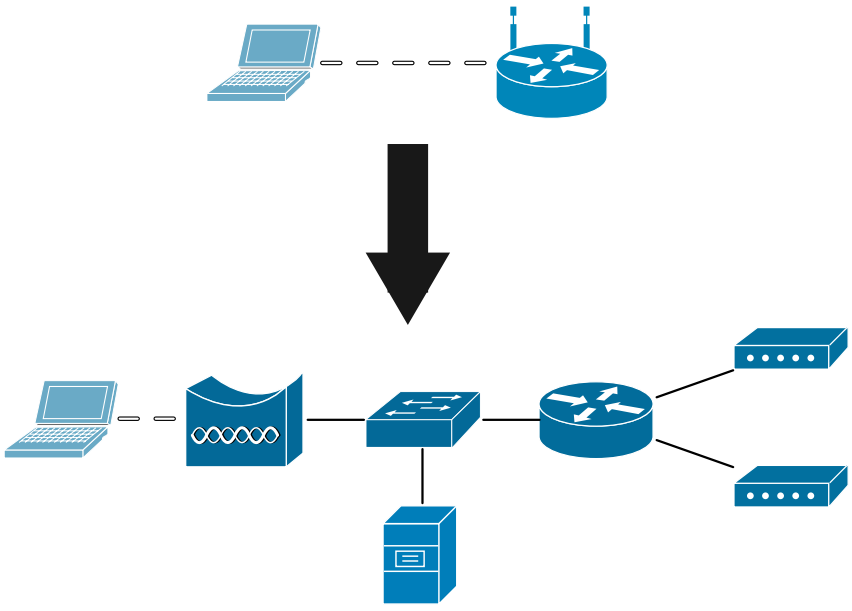

Abbildung 4.1: Multifunktions-Wi-Fi-Router: Abstrahierte Darstellung (oben) und tatsächliche Komponenten (unten)

4.2 Link-Qualität

Unabhängig von der schlussendlichen Lösung stellt sich die Frage, wie die Qualität eines Links beurteilt werden kann.

Die wichtigsten Faktoren in diesem Zusammenhang stellen der Packet-Loss, die Round-Trip Time (RTT) sowie Bandbereitenlimitierungen durch Traffic-Shaping oder physikalische Einschränkungen dar.

4.2.1 Packet-Loss

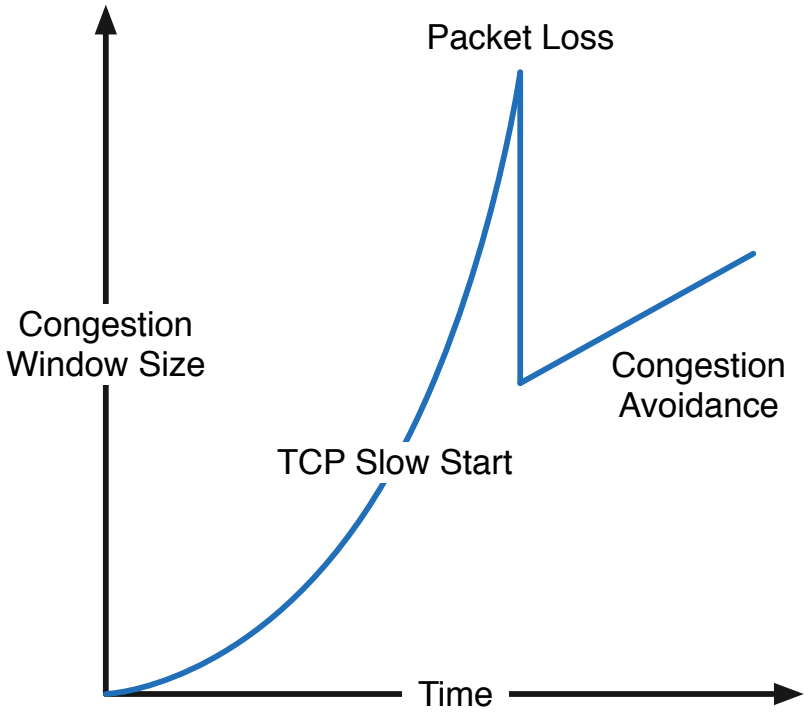

Abbildung 4.2: Congestion-Window

Ein Bestandteil des Transmission Control Protocol (TCP) ist der Congestion-Avoidance-Algorithmus, welcher bei jedem Packet-Loss die Sendegeschwindigkeit durch Verkleinerung des Congestion-Windows verlangsamt. Dies führt in der Folge zu einem schlechteren Throughput[2].

$$\text{Throughput} = \frac{s}{RTT * \sqrt{\frac{2p}{3}} + tRTO * (3 * \sqrt{\frac{3p}{8}}) * p * (1 + 32p^2)}$$

T = Throughput in Bytes pro Sekunde

s = Paketgrösse in Bytes

RTT = Round-Trip Time in Sekunden

tRTO = Retransmission-Timeout in Sekunden

p = Wahrscheinlichkeit Packet-Loss-Rate

Steigt die Wahrscheinlichkeit eines Packet-Loss, wird der Nenner in obiger Formel grösser und entsprechend sinkt der daraus resultierende Throughput.

4.2.2 Hohe RTT

Ist die Latenz sehr gross, verzögern sich die Acknowledgements (ACKs). Dies verhindert einen grossen Throughput, da der Sender mit dem Senden weiterer Pakete wartet, oder möglicherweise sogar eine erneute Übertragung des verloren geglaubten Pakets startet.

Der Throughput eines Uplinks wird folgendermassen durch dessen Latenz limitiert:

$$\text{Throughput} \leq \frac{RWIN}{RTT}$$

Der Durchsatz entspricht also maximal dem TCP Receive Window (Size) (RWin) dividiert durch die RTT.

4.3 Messverfahren

Grundsätzlich können auf Basis verschiedener Operationen Metriken gewonnen werden, um Aussagen über die Link-Qualität treffen zu können. Dies wird in der Abbildung 4.3 veranschaulicht.

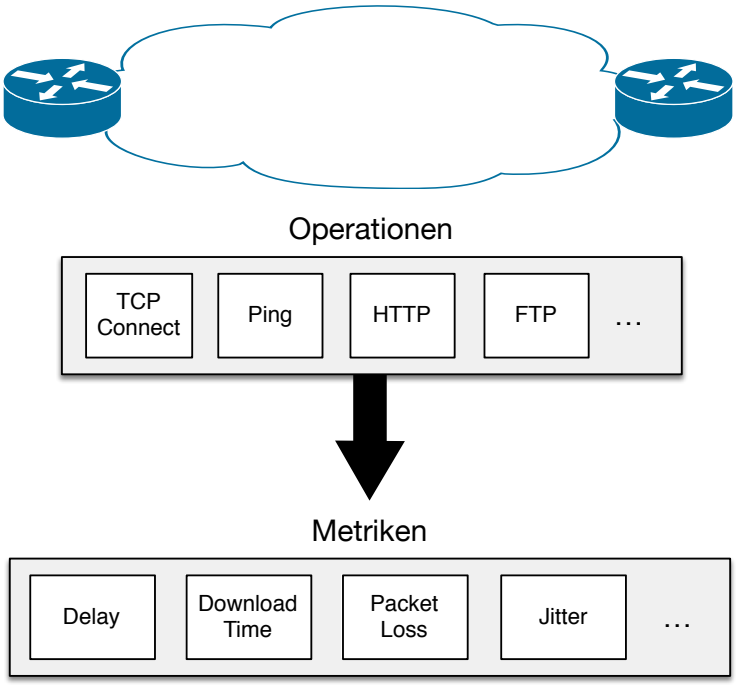

Abbildung 4.3: Messmethoden und daraus resultierende Metriken

4.3.1 Internet Protocol Service Level Agreement (IP SLA)

Ciscos IP SLA ist ein Bestandteil des Cisco Internetwork Operating System (IOS) um in Echtzeit Informationen über die Netzwerk-Performance zu gewinnen. Dabei wertet IP SLA auf dem Router verschiedene Parameter der Verbindungen aus, die diesen Router passieren.

Kontinuierlich werden Metriken wie zum Beispiel Antwortzeiten (Response-Time), Jitter und Packet-Loss gesammelt. Auf Basis der Metriken kann dann beispielsweise die Umschaltung auf einen alternativen Pfad ausgelöst werden. Somit können Administratoren Quality-of-Service gewährleisten und Probleme umgehen, bevor sich diese für den Endbenutzer bemerkbar machen.

Im vorhandenen Anwendungsfall könnten mittels IP SLA anhand der Response-Time oder dem Packet-Loss Funklöcher identifiziert werden, respektive jeweils das performanteste Interface gewählt werden.

Nachteil dieser Methode ist, dass ein Cisco Router verwendet werden müsste, was in entsprechenden Hardwarekosten resultieren würde.

4.3.2 Regelmässige Pings

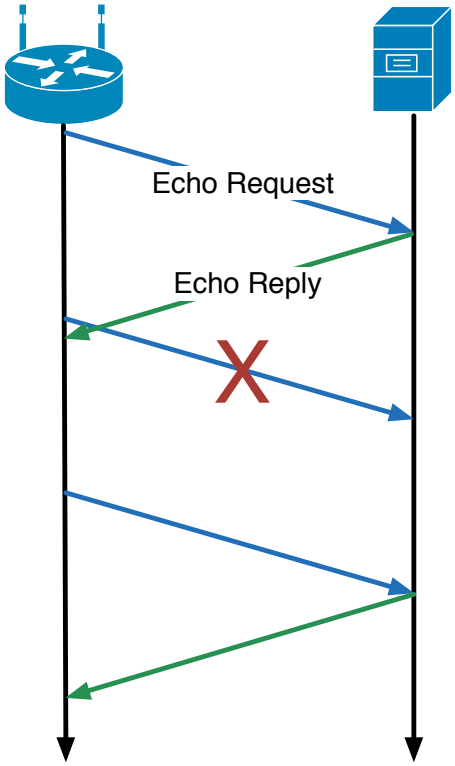

Abbildung 4.4: Ping Request mit Packet Loss

Mit Pings könnte die RTT bestimmt werden, wobei jeweils die kürzeste RTT die optimalste Latenzzeit, d.h. am schnellsten antwortende Internetverbindung darstellt. Ebenfalls wäre ein allfälliger Packet-Loss erkennbar.

Nachteil dieses Messverfahrens ist allerdings, dass durch das Versenden der Internet Control Message Protocol (ICMP) Pakete zusätzlicher Traffic generiert würde. Dieser wäre aber vernachlässigbar klein: Gehen wir von einer Grösse von 60 Bytes, und einem Interval von einer Sekunde, würde das im 24-Stunden-Betrieb in einem Traffic von ca. 10 Megabytes resultieren.

4.4 Realisierbarkeit auf OSI-Layern

Die Multipath-Fähigkeit kann auf verschiedenen Layern des OSI Models realisiert werden. Die Ansätze haben dabei jeweils unterschiedliche Vorteile und Einschränkungen,

die nachfolgend analysiert werden.

Für die Verbindungs-Aggregation auf dem Data Link Layer (Layer-2) gibt es mehrere Lösungen, wie z.B. Multilink Point-to-Point Protocol (PPP) (MLPPP), das Link Aggregation Control Protocol (LACP) oder das Port Aggregation Protocol (PAgP). Diese sind für Lösungen innerhalb einer Layer-2 Domain konzipiert. Für eine Umgebung mit Uplinks unterschiedlicher Providern sind diese nicht anwendbar, weil in diesem Fall keine Kommunikation auf dem Data Link Layer möglich ist. Die beiden Router sind wiederum durch Router getrennt und müssen deshalb mindestens über Layer-3 miteinander kommunizieren.

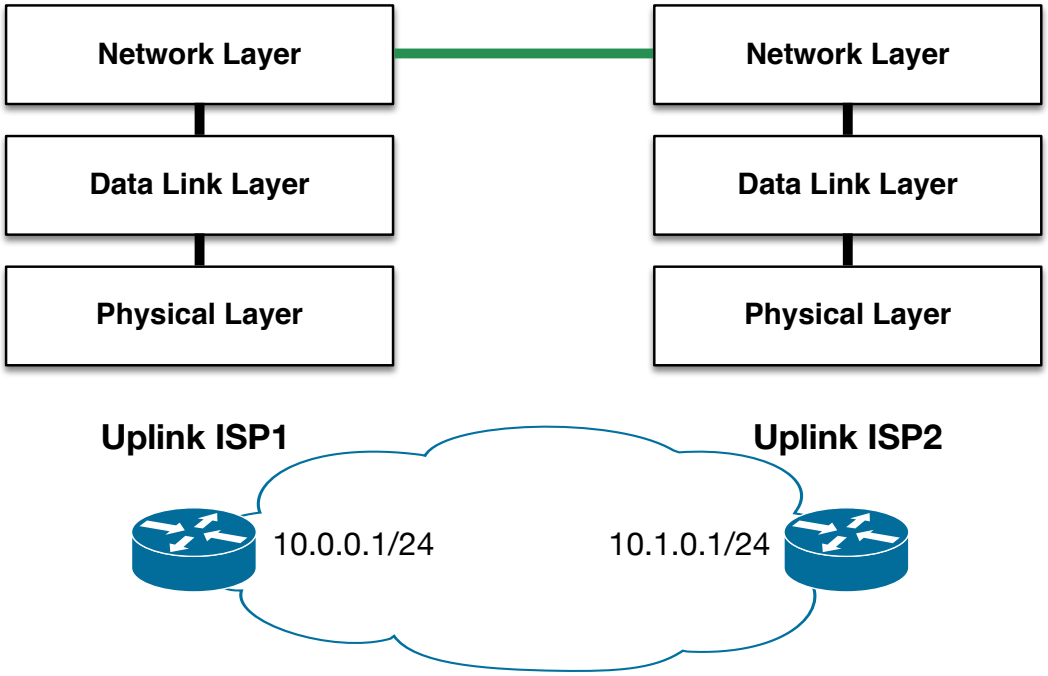

Abbildung 4.5: Konnektivität auf Layer-3

Eine Lösung auf dem Application Layer (Layer-7) wäre grundsätzlich denkbar, für die verbundenen Clients allerdings nicht transparent und damit nicht mit der Aufgabenstellung kompatibel. Die Applikationen müssten die entsprechende Multipath-Lösung unterstützen. Weiter würde die Lösung anwendungs- und protokollspezifische Lösungsansätze erfordern und wäre entsprechend wenig flexibel und aufwändig in der Implementation.

Deshalb werden nachfolgend nur Lösungen auf dem Network Layer (Layer-3) und dem Transport Layer (Layer-4) analysiert.

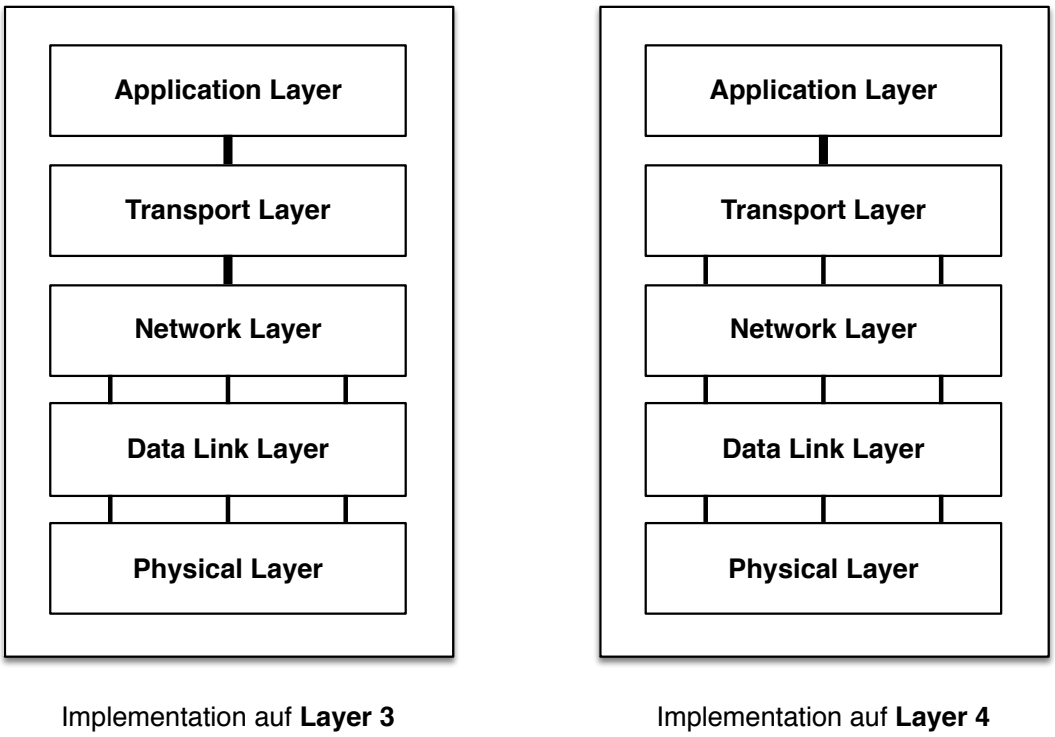

Abbildung 4.6: Netzwerkprotokoll-Stacks für Multipath-Übertragungen

4.5 Multipathing auf dem Network-Layer (Layer-3)

4.5.1 Multipath Routing

Auf dem Network Layer (Layer-3) lässt sich eine Multipath-Lösung mit Linux-Boardmitteln relativ einfach umsetzen. Der Router hat jeweils auf beiden Interfaces eine Internet Protocol (IP) und dazu passende Routen.

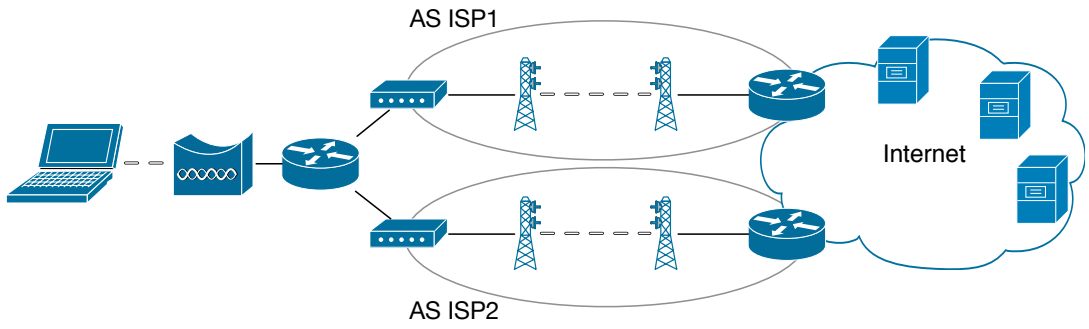

Abbildung 4.7: Multipath Routing

Als Next-Hop werden beide Routen konfiguriert, wobei diese jeweils unterschiedlich priorisiert werden können. Die Gewichtung könnte durch ein Script vorgenommen werden, welches jeweils gemäss den im Abschnitt 4.3 definierten Messverfahren die Qualität der beiden Uplinks bestimmen könnte. Ist beispielsweise der Link 1 viel schneller als Link 2, wird ersteres stärker gewichtet.

Sobald eine neue ausgehende Verbindung erstellt wird, wird ein Uplink gemäss Gewichtung ausgewählt. Ist diese Zuordnung allerdings einmal vorgenommen, ist keine Umschaltung auf einen anderen Link möglich. Fällt während einer Datenübertragung der genutzte Uplink aus, ist ein Unterbruch die Folge. Erst wenn eine neue Verbindung geöffnet wird, kann die alternative Route zum Tragen kommen.

Der Relay-Server nutzt dabei NAT um die IPs der unterschiedlichen Provider auf eine einzige IPs zu mappen. Dadurch wird sichergestellt, dass die für den Zielserver ersichtliche Source-IPs nicht ändert. Ausserdem kann so besser auf Verbindungsausfälle bei laufender Transaktion reagiert werden.

4.5.2 Hot Standby Router Protocol

Es wäre alternativ auch eine Lösung auf Hot Standby Router Protocol (HSRP) denkbar. Die beiden Uplinks wären dabei je mit einem anderen Router verbunden, die gegenüber dem Endgerät gemeinsam eine IP teilen.

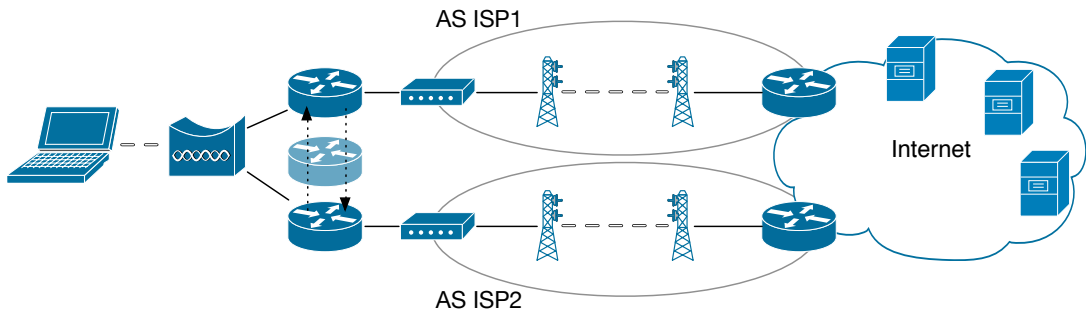

Abbildung 4.8: Multipath durch Nutzung von HSRP

Das Endgerät verbindet sich dabei via Wireless-Bridge auf eine virtuelle IPs, die zwischen den beiden Routern wechselt, was für das Endgerät ist nicht ersichtlich ist. Man spricht deshalb auch von einem *virtuellen Router* (in Abbildung 4.8 leicht transparent dargestellt). Es wird detektiert, welcher der beiden Router die stabilere Verbindung hat, um einen entsprechenden Failover, d.h einen Wechsel der virtuellen IPs auf den nicht-aktiven Router, auszulösen.

Eine solche Lösung könnte mittels Ciscos IP SLA umgesetzt werden (vgl. Abschnitt 4.3.1). IP SLA kann bei einer Verletzung der gestellten Anforderung die Umschaltung auf den alternativen Pfad veranlassen. Dafür sind keine zusätzliche Verbindungen notwendig, es werden die Parameter der vorhandenen Uplinks überwacht. Ergo gibt es keinerlei Traffic-Overhead bei dieser Lösung.

Ein Vorteil dieser Lösung ist das einfache Deployment: Die Lösung ist in sich geschlossen und nicht auf externe Server angewiesen. Es ist kein zentraler Relay-Server erforderlich. Die Daten werden jeweils direkt mit dem Zielserver ausgetauscht, was einer geringen Latenz zuträglich ist. Weiter wird eine dynamischen Verteilung gemäss Abschnitt 3.1.1 erreicht.

Der beachtliche Nachteil dieser Lösung ist, dass bei einem Failover alle bestehenden TCP-Sessions unterbrochen werden. Beim Surfen im Web würde dies kaum auffallen. Wird jedoch beispielsweise eine grosse Datei heruntergeladen oder ein Stream konsumiert, wären ärgerliche Verbindungsunterbrüche die Folge. Die im Abschnitt 3.1.2 gestellte Anforderung der Aufrechterhaltung wäre damit nicht erfüllt. Ausserdem wäre immer nur eine Verbindung aktiv. Es würde folglich je nach Situation nicht das theoretische Maximum der zur Verfügung stehenden Kapazität genutzt.

4.5.3 Fazit

Eine Umsetzung auf Layer-3 wäre grundsätzlich möglich. Da auf Layer-3 die Paketreihenfolge nicht bekannt ist, kann sie die Anforderungen nicht erfüllen. Eine unterschiedliche Paketumlaufzeit ist bei mehreren Pfaden sehr wahrscheinlich. Es muss mit entsprechenden Verschiebungen in der Paketreihenfolge gerechnet werden. Auch Paketverluste, die bei mobilen Verbindungen zu erwarten sind, können auf Layer-3 nicht erkannt werden.

4.6 Multipath auf dem Transport-Layer (Layer-4)

4.6.1 MPTCP

Das MPTCP ist eine von einer IETF-Arbeitsgruppe vorgeschlagene Erweiterung des klassischen TCP. Im Januar 2013 wurde der experimentelle Standard RFC 6824[5] veröffentlicht, in welchem MPTCP spezifiziert ist. MPTCP ist auf dem Layer-4, spätestens seit der Nutzung von Apples iPhone-Betriebssystem iOS[1], der am weitesten verbreitetste Multipath-Ansatz.

Zu den angedachten Anwendungsfällen von MPTCP gehört auch die parallele Nutzung mehrerer Wireless-Technologien — der Standard scheint ziemlich genau den untersuchten Anwendungsfall abzudecken. Mit dem Linux kernel MultiPath TCP project[1] bereits eine stabile Implementation des MPTCP Standards für Linux.

Gegenüber den oberen Schichten bietet MPTCP das gleiche Interface wie TCP. Die Applikationen müssen nicht angepasst werden, um MPTCP zu unterstützen. Das Betriebssystem aktiviert die Erweiterung beim Verbindungsaufbau automatisch, sofern dies beide Kommunikationspartner unterstützen.

MPTCP kommt erst nach dem initialen Handshake zum Einsatz. Für den Verbindungsaufbau wird der systemweite Default-Gateway genutzt. Es muss jederzeit ein (via einem verlässlichen Pfad) erreichbarer Next-Hop bekannt und konfiguriert sein. Dieses Verhalten wurde so auch entsprechend in der Mailinglist[2] des MPTCP Projektes bestätigt[3].

Das MPTCP-Projekt bietet modifizierte Versionen der *net-tools*-Collection an. So können beispielsweise mittels `netstat -m` alle aktiven MPTCP-Links angezeigt werden. Weiter steht zur Auswertung des Netzwerkverkehrs eine Version von *tcpdump* mit MPTCP-Unterstützung bereit. Auch *Wireshark* unterstützt die MPTCP-Optionen (seit Version 1.7.1[9]).

[1] Linux kernel MultiPath TCP project: `http://mptcp.info.ucl.ac.be/`
[2] MPTCP-Mailinglist: `https://listes-2.sipr.ucl.ac.be/sympa/info/mptcp-dev`

MPTCP ermöglicht mit dem Modus `multipath off`, einen Link beim Multipathing nicht zu verwenden. Auf diese Weise könnte ein sehr schlechter Link von Anfang an deaktiviert werden, damit MPTCP einen schnelleren Throughput erreicht.
Weiter ist es mittels dem Befehl `multipath backup` möglich, einen Link ausschliesslich als Backup-Pfad zu verwenden. Nur wenn keine Konnektivität über andere Links vorhanden ist, kommt dieser zum Einsatz. Während einer Verbindung kann die eine Seite eine Prioritätsänderung über das `MP_PRIO` Signal anfordern[4]. Damit könnten kostenpflichtige oder langsame Links benachteiligt werden und dadurch eine gewisse Priorisierung der Links erreicht werden.

Es sei an dieser Stelle darauf hingewiesen, dass gewisse Sicherheitsprodukte unbekannte TCP-Optionen löschen bzw. durch *NOOP* Werte ersetzen[13]. Dadurch wird die `MP_CAPABLE`-Option entfernt und eine Multipath-Verbindung in der Folge verunmöglicht. Will man den TCP-Flow analysieren, macht diese Option durchaus Sinn, im Normalfall ist dieses Verhalten allerdings unerwünscht.

4.6.2 Aufbau

Weil der MPTCP-Standard von den wenigstens Servern unterstützt wird, müsste auch bei dieser Variante ein Relay-Server eingesetzt werden. Dieser wäre via MPTCP über mehrere Pfade mit dem Testgerät verbunden. Der Relay-Server vermittelt zwischen dem eigentlichen Zielserver und dem Testgerät: Für die unzuverlässige Mobilfunkübertragung kommt MPTCP zum Einsatz, während für die stabile Verbindungs zwischen dem Relay-Server und dem Zielserver das ursprüngliche Protokoll vernwedet wird.

VPN

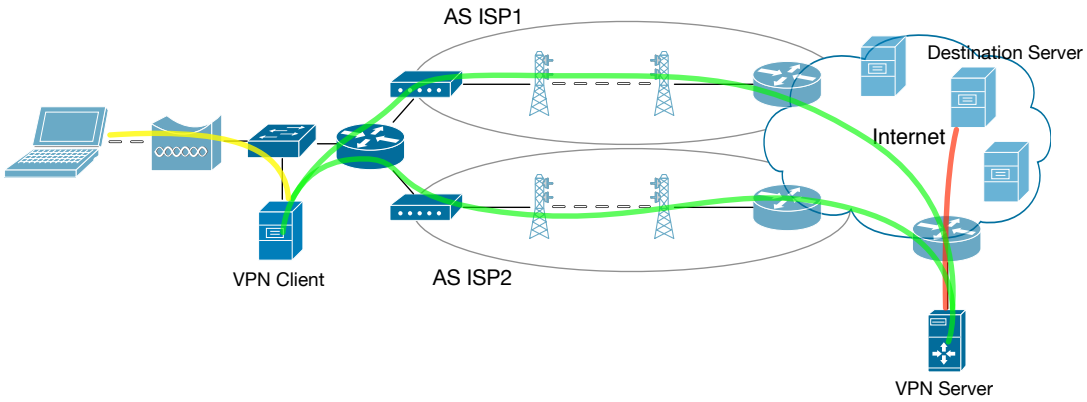

Abbildung 4.9: Traffic-Routing via VPN-Tunnel auf Basis von MPTCP

Ein solcher Relay-Server liesse sich beispielsweise auf Basis von *OpenVPN*[3] implementieren. *OpenVPN* kann mit geringem Aufwand so konfiguriert werden, dass das TCP (statt dem standardmässigen User Datagram Protocol (UDP)) zur Kommunikation genutzt wird. Weil beide Systeme MPTCP-fähig sind, wird diese Erweiterung automatisch durch das Betriebssystem beim Verbindungsaufbau aktiviert. Die Pakete werden beim diesen Ansatz durch einen MPTCP-fähigen Tunnel geroutet (vgl. Abbildung 4.10).

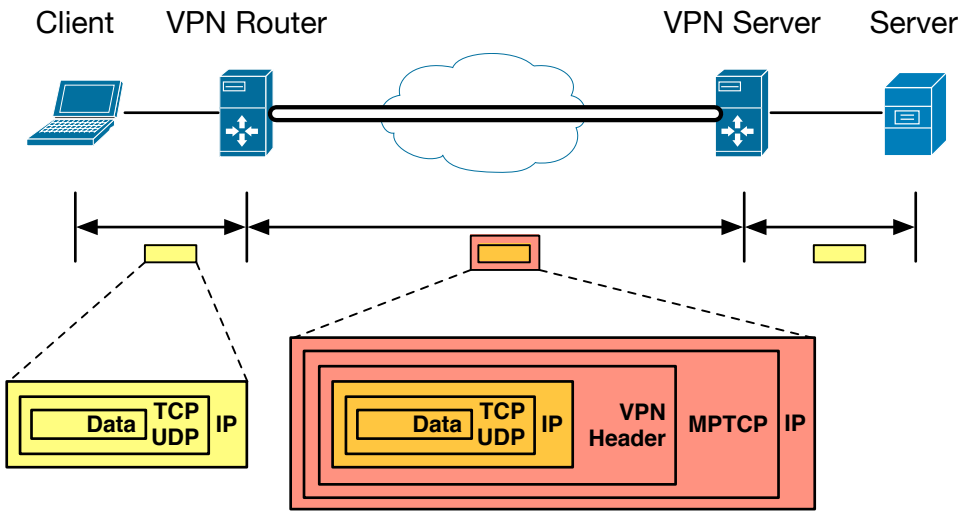

Abbildung 4.10: Packet-Stack beim Durchlaufen des Tunnels.

[3]OpenVPN - Open Source VPN: `http://openvpn.net/`

Der Stack eines Pakets, das über den VPN-Tunnel geschickt wird, ist in Abbildung 4.10 dargestellt. Das ursprüngliche Paket wird durch VPN verpackt. Falls als Transportmethode TCP gewählt wurde und sowohl Client wie auch Server MPTCP-fähig sind, wird das Paket via MPTCP verschickt werden.

Durch das Konfigurieren eines sogenannten *Redirect-Gateways* auf dem Access-Point wird sämtlicher Client-Traffic (in Abbildung 4.9 gelb eingezeichnet) via MPTCP-VPN-Tunnel geroutet (grün). Der OpenVPN-Server nutzt NAPT um die entsprechende Anfrage auszuführen (rot). Der Zielserver sieht als Source-Adresse die IP-Adresse des *OpenVPN*-Servers und schickt den Traffic an letzteren zurück. Aufgrund von NAPT weiss der *OpenVPN*-Server, wohin die Antwort geroutet werden muss.

Alternativ wäre auch ein Einsatz anderer VPN-Technologien, wie z.B. *IPsec* (über *strongSwan*[4]) denkbar.

Die Idee, TCP in TCP zu verpacken wird kontrovers diskutiert. Olaf Titz beschreibt in seinem Paper *Why TCP Over TCP Is A Bad Idea*[12] beispielsweise einen sogenannten *Meltdown Effect*, der eintreten kann, sobald es zu Paketverlusten kommt.

> Imagine what happens when [...] the base connection starts losing packets. The lower layer TCP queues up a retransmission and increases its timeouts. Since the connection is blocked for this amount of time, the upper layer (i.e. payload) TCP won't get a timely ACK, and will also queue a retransmission. Because the timeout is still less than the lower layer timeout, the upper layer will queue up more retransmissions faster than the lower layer can process them. This makes the upper layer connection stall very quickly and every retransmission just adds to the problem - an internal meltdown effect.

Dieses Problem wurde 2007 in einem Paper aus Japan[6] nachgewiesen.

Weiter gibt es beim *OpenVPN*-Projekt einen Bug-Report[4], der die schlechte Performance beim Einsatz von TCP als Transportprotokoll bemängelt.

[4]strongSwan - IPsec for Linux: `http://www.strongswan.org`

Socket Secure (SOCKS)

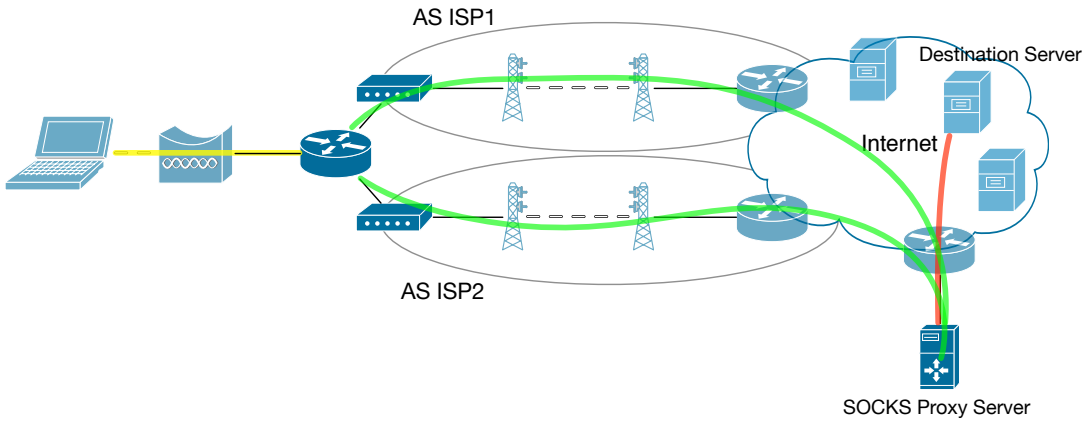

Abbildung 4.11: Traffic-Routing via SOCKS-Proxy Server auf Basis von MPTCP

Der Aufbau würde sich kaum von der VPN-Variante unterscheiden. Statt den Traffic via VPN-Server weiterzuleiten, würde der Traffic durch den im Client eingetragenen Default-Gateway mittels eines transparenten SOCKS-Redirector zum entfernten SOCKS-Server weitergeleitet.

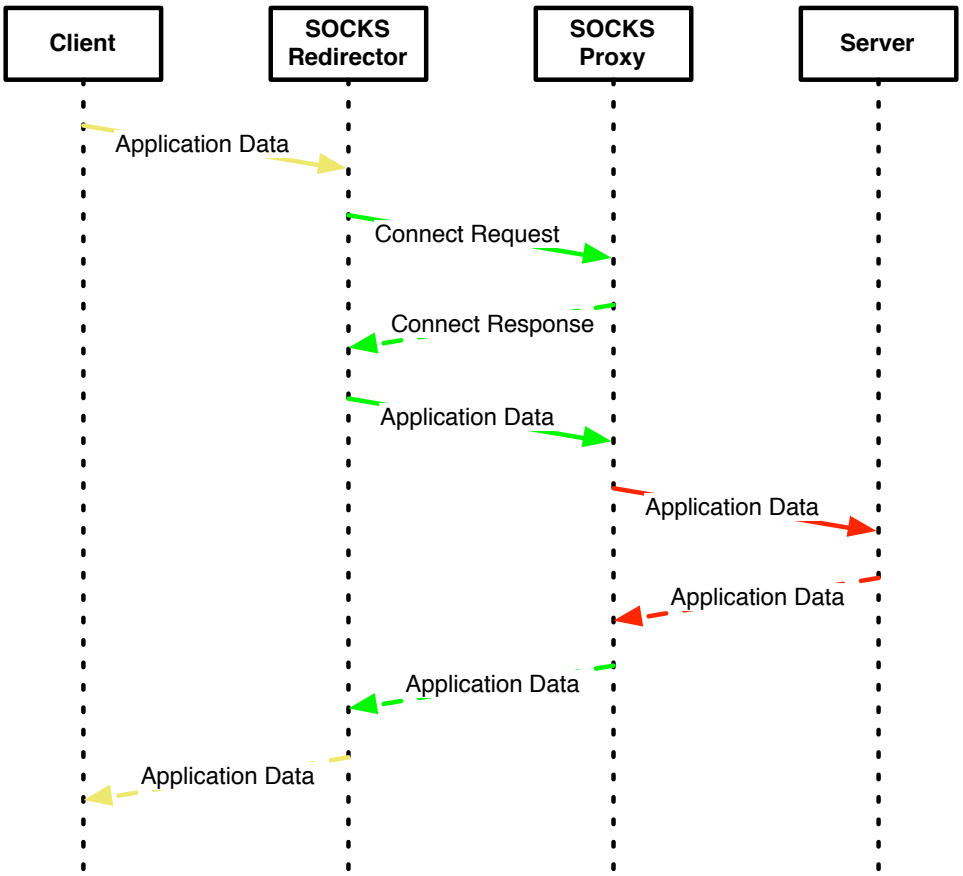

Abbildung 4.12: Sequenzdiagramm des Verbindungsaufbaus und Datenaustauschs.

Pro Session baut SOCKS in beide Richtungen je eine Socket-Verbindung zwischen Redirector und Proxy auf. Nach dem Aufbau der Session wird dem SOCKS-Proxy-Server die Ziel-IP sowie der Ziel-Port bekannt gegeben. Das hat den Vorteil, dass die ursprünglichen Angaben nicht im Daten-Payload der nachfolgenden Pakete aufgeführt werden müssen. Nach dem Verbindungsaufbau können die Datenpakete ausgetauscht werden. Der SOCKS-Redirector übernimmt jeweils die Daten aus der Anwendungsschicht (OSI Layer 5+) und vermittelt diese zwischen den beiden Sockets (vgl. Abbildung 4.12). Sind sowohl Redirector wie auch Proxy-Server MPTCP-fähig, wird die MPTCP-Erweiterung automatisch aktiviert. Um zu verdeutlichen, wie SOCKS die IPs und Ports ersetzt, ist in Abbildung 4.13 ein Beispiel eines Packet-Flows aufgezeigt:
Ein Paket wird vom Client, resp. dem Endbenutzer (IP: A, Port 49200), zum angefragten Server (IP: D, Port 80) gesendet (Nr. 1-3). Der Server antwortet anschliessend dem Client (Nr. 4-6).

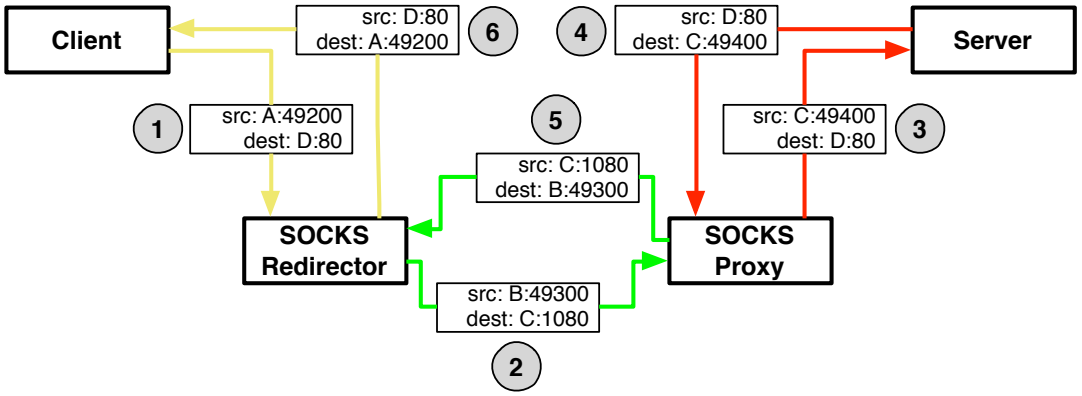

Abbildung 4.13: Packet-Flow beim Einsatz eines transparenten SOCKS-Proxy-Servers

Weil der Ansatz von SOCKS nur bei TCP-Traffic angewendet werden kann, funktioniert diese Lösung nicht, um Nicht-TCP-Traffic (z.B. UDP, ICMP, ...) über mehrere Pfade zu übertragen.

Im Unterschied zu VPN wird ein Paket nicht in ein anderes "verpackt" (*encapsulated*), sondern nur die unter der Anwendungsschicht liegenden Layer ausgetauscht. Es kann im Vergleich mit VPN deshalb etwas Traffic-Overhead eingespart werden und man umgeht die angesprochene TCP-over-TCP-Problematik.

4.7 Zwischenfazit

Multipath-Ansätze müssen unter anderem die folgenden zwei Probleme berücksichtigen:

- Die Paketreihenfolge kann bei Pfaden mit unterschiedlicher Latenz verloren gehen.
- Wird ein Pfad unterbrochen, geht ein Teil der Pakete verloren.

Eine Lösung auf dem Transport-Layer (Layer-4) hat den Vorteil, dass hier die genannten Probleme transparent gelöst werden können. Den unteren Schichten stehen nicht genügend Informationen zur Verfügung, um beispielsweise verloren gegangene Pakete nochmals anzufordern oder eine falsche Paketreihenfolge zu erkennen und zu korrigieren. MPTCP nutzt dieses Wissen und erreicht dadurch eine besser Performance bei geringerem Traffic-Overhead.

Eine Implementation auf Layer-4 scheint deshalb am besten geeignet und wird entsprechend auch zur Umsetzung empfohlen.

VPN hat eine Schwäche mit TCP over TCP, resp. in diesem Fall TCP over MPTCP. Im Gegenzug bietet SOCKS keine Lösung, um den Nicht-TCP-Traffic über MPTCP zu versenden. Es bietet sich eine Mischvariante an: Den Nicht-TCP-Traffic mittels VPN und den TCP-Traffic über einen transparenten SOCKS-Proxy zu versenden.

4.8 Linux Link-Management

Eine zentrale Frage ist, wie der Router erkennen kann, wann ein Link vorhanden ist, resp. welche Links gerade verfügbar sind. Dafür ist das im Linux-Kernel enthaltene Netzwerk-Subsystem zuständig[11].

Wird ein neues Netzwerk-Interface hinzugefügt und mittels der Funktion `register_netdev(struct net_device *dev)` registriert, wird eine `net_device`-Struktur initialisiert und in die globale Liste der Netzwerk-Geräte eingetragen. Beim Entfernen wird das Interface aus der globalen Liste gelöscht.

Die Netzwerk-Treiber-Schicht führt also für jedes erkannte Interface eine `net_device`-Struktur. Darüber können mittels der `get_stats`-Funktion statistische Daten abgefragt werden, die auch verfügbar sind, wenn das Interface deaktiviert ist. Diese Funktion wird beispielsweise auch durch `ifconfig` genutzt, um die Paket-Zähler auszulesen.

Da Netzwerkverbindungen per Definition instabil und unbeständig sind, stellt das Netzwerk-Subsystem Schnittstellen für etwaige Statusänderungen zur Verfügung. Die meisten Netzwerktechnologien mit physikalischer Verbindung kennen einen sogenannten Carrier-Zustand: Ist der Carrier anwesend, heisst das, dass die Hardware existent und betriebsbereit ist. Gibt es einen Verbindungsunterbruch, verschwindet der Carrier und der Treiber meldet dem Kernel dies. Die Verbindung gilt dann als inaktiv.

Der Treiber verwendet dazu folgende Funktionen des Netzwerk-Subsystems:

```
1 // Carrier auf "off" stellen
2 void netif_carrier_off(struct net_device *dev);
3
4 // Carrier auf "on" stellen
5 void netif_carrier_on(struct net_device *dev);
6
7 // Abfrage des Carrier-Stands
8 int netif_carrier_ok(struct net_device *dev);
```

Listing 4.1: Gemäss `net/sched/sch_generic.c` aus dem Linux-Quellcode-Repository[5]

Daraus ergibt sich, dass Linux den Zustand eines Links nur in binärer Form beschreibt: Entweder es besteht eine Verbindung oder nicht. Bei Mobilfunktverbindungen wäre aber zusätzlich interessant, welche Mobilfunkverbindung beispielsweise das bessere Signal aufweist, damit die die Interfaces entsprechend priorisiert werden können.

4.9 Modem details

Um den Traffic aufgrund von Mobilfunk-Technologie oder Signalstärke zu verteilen, muss der Treiber entsprechende Information zurückgeben können. Der `ModemManager`-Service und die dazugehörige Library stellen in Linux dafür via Desktop-Bus (D-Bus) Interface Funktionen zur Verfügung.

Mit dem Command Line Interface (CLI) Tool `gdbus`[6] steht ein Werkzeug zur Verfügung, dass das Arbeiten mit D-Bus Objekten erlaubt. Mittels `call` kann beispielsweise eine Funktion aufgerufen werden und mithilfe von `introspect` können alle Eigenschaften (Properties) eines Objekts aufgelistet werden.

Die Liste der Modems ist lässt sich mittels `introspect` auf die `Modems`-Node (vgl. Listing 4.2) abfragen.

```
gdbus introspect --system --dest
   org.freedesktop.ModemManager --object-path
   /org/freedesktop/ModemManager/Modems
```

Listing 4.2: Befehl: Introspection der `Modems` Node

```
1 node /org/freedesktop/ModemManager/Modems {
2   node 0 {
3   };
4   node 1 {
5   };
6 };
```

Listing 4.3: Rückgabe aller Modems nach dem Befehl aus Listing 4.2

Jedes Modem wird über eine Nummer identifiziert. Beispielsweise kann das Modem mit

[5] `https://github.com/torvalds/linux/blob/a6cc0cfa72e0b6d9f2c8fd858aacc32313c4f272/net/sched/sch_generic.c`

[6] gdbus - Tool for working with D-Bus: `https://developer.gnome.org/gio/stable/gdbus.html`

der ID 3 über folgenden Aufruf angesteuert werden, um eine beliebige Funktion des Modems auszuführen:
`/org/freedesktop/ModemManager/Modems/3`

Für den Prototypen sind zwei Methoden des Modems interessant:

- `org.freedesktop.ModemManager.Modem.Gsm.Network.GetSignalQuality` gibt die Signalstärke in Prozent zurück.
- Via der Eigenschaft `AccessTechnology` des Modems lässt sich der Mobilfunkstandard auslesen. Der Rückgabewert (eine Zahl) lässt sich mittels dem Aufzählungstyp `MM_MODEM_GSM_ACCESS_TECH` (vgl. Listing 4.6) interpretieren.

```
OBJECT="/org/freedesktop/ModemManager/Modems/3"
METHOD="org.freedesktop.ModemManager.Modem.\␣
   Gsm.Network.GetSignalQuality"
gdbus call --system --dest org.freedesktop.ModemManager
   --object-path $OBJECT --method $METHOD
```

Listing 4.4: Aufrufen der Methode `GetSignalQuality()` auf dem Modem 3 mittels `gdbus`

```
OBJECT="/org/freedesktop/ModemManager/Modems/3"
gdbus introspect --system --dest
   org.freedesktop.ModemManager --object-path $OBJECT |
   grep "AccessTechnology"
```

Listing 4.5: Abfragen des Mobilfunkstandards von Modem 3 mittels `gdbus`

```
1  /* MM_MODEM_GSM_ACCESS_TECH enum values */
2  typedef enum {
3      MM_MODEM_GSM_ACCESS_TECH_UNKNOWN = 0,
4      MM_MODEM_GSM_ACCESS_TECH_GSM = 1,
5      MM_MODEM_GSM_ACCESS_TECH_GSM_COMPACT = 2,
6      MM_MODEM_GSM_ACCESS_TECH_GPRS = 3,
7      MM_MODEM_GSM_ACCESS_TECH_EDGE = 4,
8      MM_MODEM_GSM_ACCESS_TECH_UMTS = 5,
9      MM_MODEM_GSM_ACCESS_TECH_HSDPA = 6,
10     MM_MODEM_GSM_ACCESS_TECH_HSUPA = 7,
11     MM_MODEM_GSM_ACCESS_TECH_HSPA = 8,
```

```
12     MM_MODEM_GSM_ACCESS_TECH_HSPA_PLUS = 9,
13     MM_MODEM_GSM_ACCESS_TECH_LTE = 10,
14 } MMModemGsmAccessTech;
```

Listing 4.6: enum MM_MODEM_GSM_ACCESS_TECH

4.10 Evaluation

4.10.1 VPN Server

Es soll geprüft werden, ob die angedachte Lösung (vgl. Abschnitt 4.7) auch in der Praxis funktioniert. Als VPN-Server kommt dafür *OpenVPN* zum Einsatz. Dies ist dem Umstand geschuldet, dass das Projektteam bereits über einen entsprechende Wissensfundus zur Konfiguration und dem Einrichten eines entsprechenden Tunnels verfügt.

Server

Das Installieren und Konfigurieren eines *OpenVPN*-Servers gestaltet sich dank dem *puppet*-Modul *puppet-openvpn*[7] relativ einfach:

```
1 openvpn::server { 'example':
2   country      => 'CH',
3   province     => 'ZH',
4   city         => 'Zurich',
5   organization => 'example.com',
6   email        => 'info@example.com',
7   server       => '10.10.10.0 255.255.255.0',
8   push         => ['redirect-gateway def1', 'dhcp-option
       DNS 8.8.8.8'],
9   proto        => 'tcp'
10 }
```

[7] R. Schmid, Oktober 2013: *OpenVPN* module for puppet including client config/cert creation: `https://github.com/luxflux/puppet-openvpn`

Clients können mit der folgenden Definition hinzugefügt werden:

```
1 openvpn::client { 'testgeraet.example.com':
2   server => 'example'
3 }
```

Listing 4.7: Client-Definition im Puppet-Rezept

Das Puppet-Modul sorgt automatisch für das Generieren und Konfigurieren der benötigten Zertifikate und Schlüssel. Die Client-Konfiguration ist danach auf dem Server unter `/etc/openvpn/myopenvpn/download-configs/testgeraet.myopenvpn.example.com.tar.gz` abgelegt. It just works™.

Client

Die generierte Konfiguration wird auf dem Client unter `/etc/openvpn` abgelegt. Wird der *OpenVPN*-Dienst nun gestartet, verbindet sich dieser mit dem Server und konfiguriert ein Tunnel-Interface. Sämtlicher Traffic wird nun über dieses Tunnel-Interface umgeleitet.

Tests und Fazit

Mittels *Wireshark* liess sich bestätigen, dass MPTCP und damit mehrere Pfade verwendet werden. *OpenVPN* lässt sich also mit wenig Aufwand so konfigurieren, dass eine Verbindung via TCP bzw. MPTCP erfolgt.

4.10.2 SOCKS Redirector

Um die TCP-Pakete transparent via SOCKS umzuleiten, ist ein SOCKS-Redirector vonnöten. Die Wahl ist hier auf *redsocks*[8] gefallen, weil es im Gegensatz zu *TranSocks*[9] SOCKS 5 sowie Authentifizierung unterstützt. Weiter ist *redsocks* in den Software-Repositories von vielen gängigen Linux Distributionen verfügbar.

[8] redsocks — transparent socks redirector: `http://darkk.net.ru/redsocks/`
[9] TranSocks — Transparent SOCKSifying Proxy: `http://transocks.sourceforge.net/`

4.10.3 SOCKS Server

Auf Serverseite kommt *Dante*[10] als SOCKS Server zum Einsatz. *Dante* unterstützt Authentifizierung und glänzt mit hoher Performance. Im Gegensatz zu *OpenSSH* kann es ohne Verschlüsselung genutzt werden und erlaubt dadurch den praktisch Traffic-Overhead-losen Betrieb.

[10] Dante — A free SOCKS server: `http://www.inet.no/dante/`

5 Design

5.1 Physische Architektur / Netwerktopologie

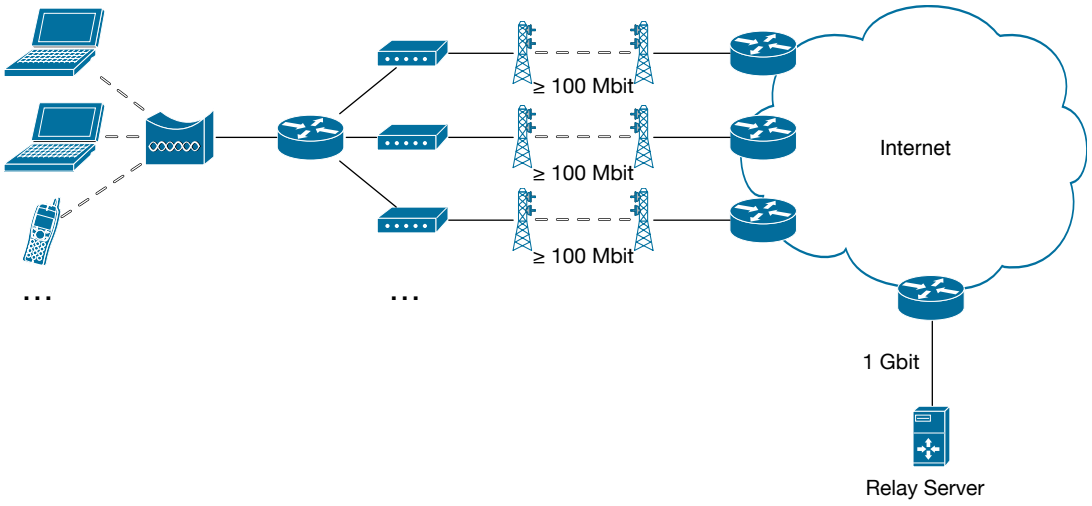

Abbildung 5.1: Netzwerktopologie

In der einfachsten Ausführung besteht das Setup aus dem Multifunktions-Wi-Fi-Router, der über mehrere Mobilfunknetze sowie evt. über WLAN mit dem Internet verbunden ist. Da der Relay-Server in der Theorie dem doppelten der verfügbaren Anbindung des Multifunktions-Wi-Fi-Router ausgesetzt sein könnte, muss eine entsprechend gute Anbindung eingeplant werden: Findet beispielsweise ein Download statt, bedeutet das aus der Sicht des Relay-Servers sowohl das Herunterladen vom Zielserver, als auch das Hochladen zum Multifunktions-Wi-Fi-Router.

Weiter sollte der Anbieter des für den Relay-Server zur Verfügung stehenden Uplinks über die entsprechenden Peering-Agreements verfügen, um die Latenz möglichst klein zu halten.

5.2 Logische Architektur

Aufgrund der erzielten Resultaten in der Evaluation (vgl. Abschnitt 4.10.1), wird die Kombination von SOCKS und VPN gewählt. TCP Pakete werden via transparentem SOCKS Proxy umgeleitet. Der restliche Traffic wird via VPN geroutet und dadurch in TCP eingekapselt. Mit diesem Verfahren kann der gesamte Verkehr über mehrere Pfade transferiert werden.

5.2.1 Routing

Die Idee, für nicht-TCP Pakete einen anderen Gateway zu nutzen, erfordert eine Analyse der auf dem Router eintreffenden Pakete. Für die Analyse von Paketen sowie weitere typische Firewall-Aufgaben ist unter Linux `iptables` zuständig.

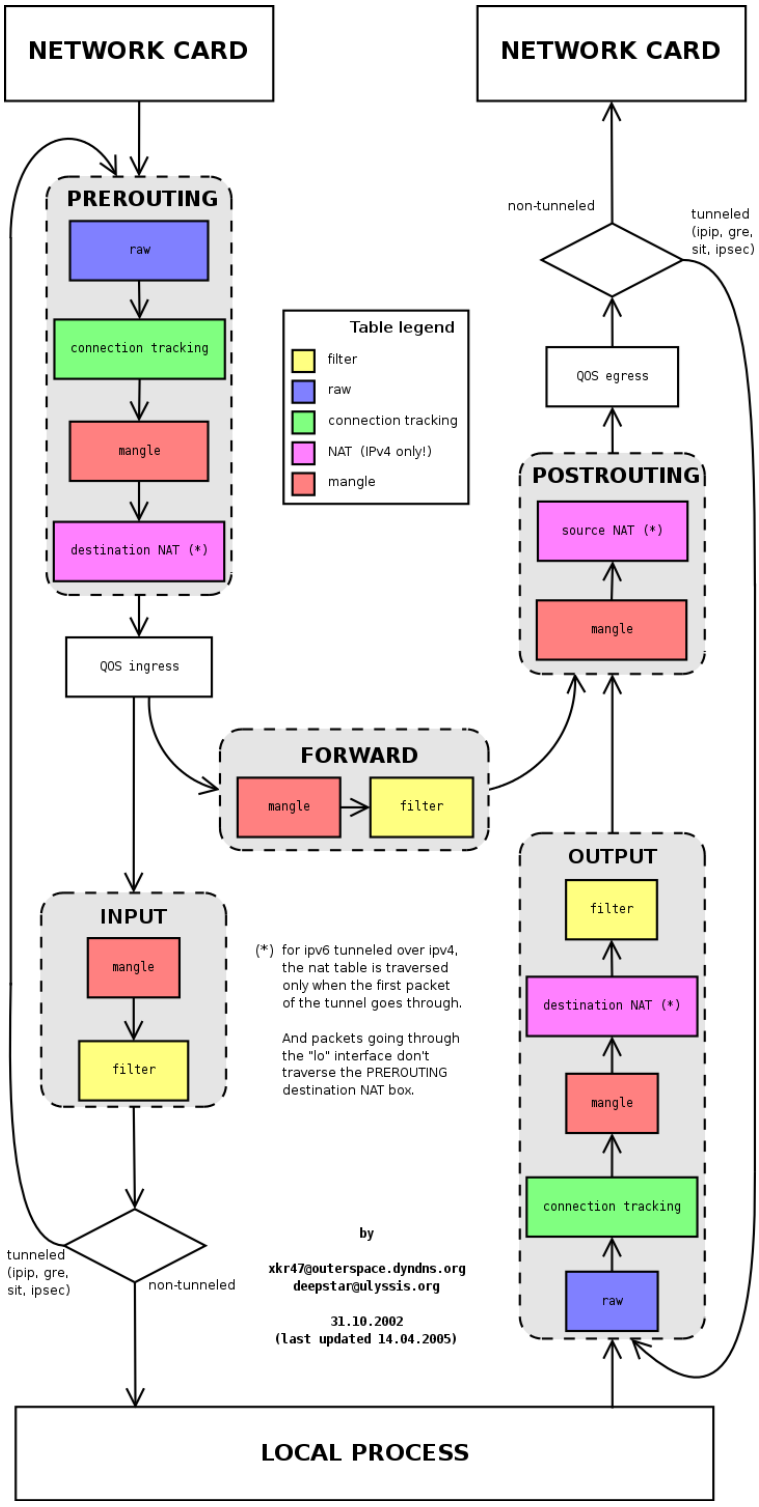

Abbildung 5.2: Packet Flow unter Linux durch `iptables`[1].

[1] `http://serverfault.com/a/523591`

`iptables` arbeitet dabei mit mehreren sogenannten Chains, die ein Paket im Rahmen der Verarbeitung durchläuft (Abbildung 5.2). Die `PREROUTING` Chain kann benutzt werden, um Pakete zu markieren und danach im Routingprozess basierend auf der Markierung eine Routingtabelle zu wählen. So können nicht-TCP Pakete markiert werden, um dann eine Routingtabelle mit dem VPN-Gateway als Next-Hop zu wählen (Abbildung 5.5).

Weiter bietet `iptables` die Möglichkeit, Pakete die für einen anderen Host bestimmt wären, an eine an einen lokalen Port gebundene Software weiterzuleiten, ohne das Paket zu modifizieren. Dies ermöglicht, alle TCP Pakete an einen lokalen SOCKS Redirector umzuleiten, der seinerseits die Pakete via SOCKS Proxy weiterleitet. Dieser Ablauf wird in Abbildung 5.3 grafisch dargestellt.

Da Pakete, die für private Subnetze bestimmt sind, im Normalfall nicht geroutet werden müssen, werden diese von den definierten Regeln ausgeschlossen ("Ignored Destination").

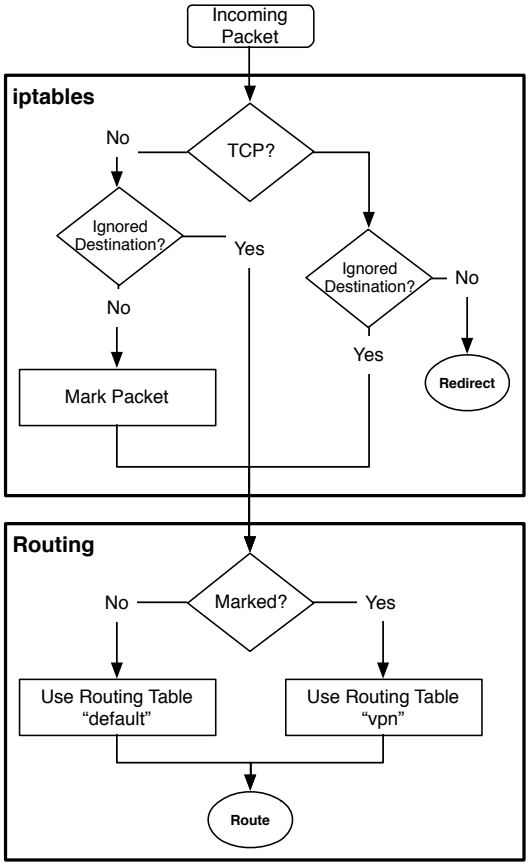

Abbildung 5.3: Protokollabhängige Routingtabellenwahl

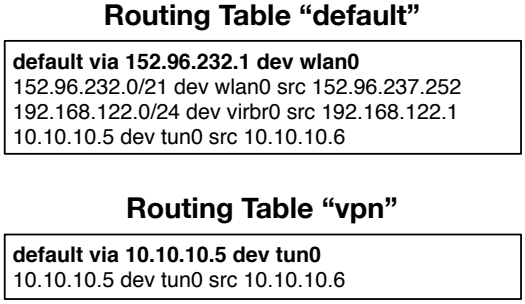

Abbildung 5.4: Beispiele von Routingtabellen

Ein Problem dabei ist die je nach Linux-Distribution unterschiedliche `rp_filter`-Einstellung: Je nach Konfiguration nimmt der Router keine Pakete auf einem Interface an, dass aus einem fremden Subnetz kommt.

5.2.2 Default-Gateway-Wahl

Das im Abschnitt 4.6.1 beschriebene Problem der Wahl des Default-Gateways wird mit einem Script gelöst. Mittels der im Abschnitt 4.3 dokumentierten Messverfahren werden die Pfade regelmässig geprüft und ein zuverlässiger Pfad als Default-Gateway gewählt. Da der Default-Gateway nur für den Handshake relevant ist (respektive 3 Pakete), wird auf eine aufwändige Geschwindigkeitsmessung verzichtet. Die Wahl hängt einzig von der Latenz und vom Packet-Loss ab.

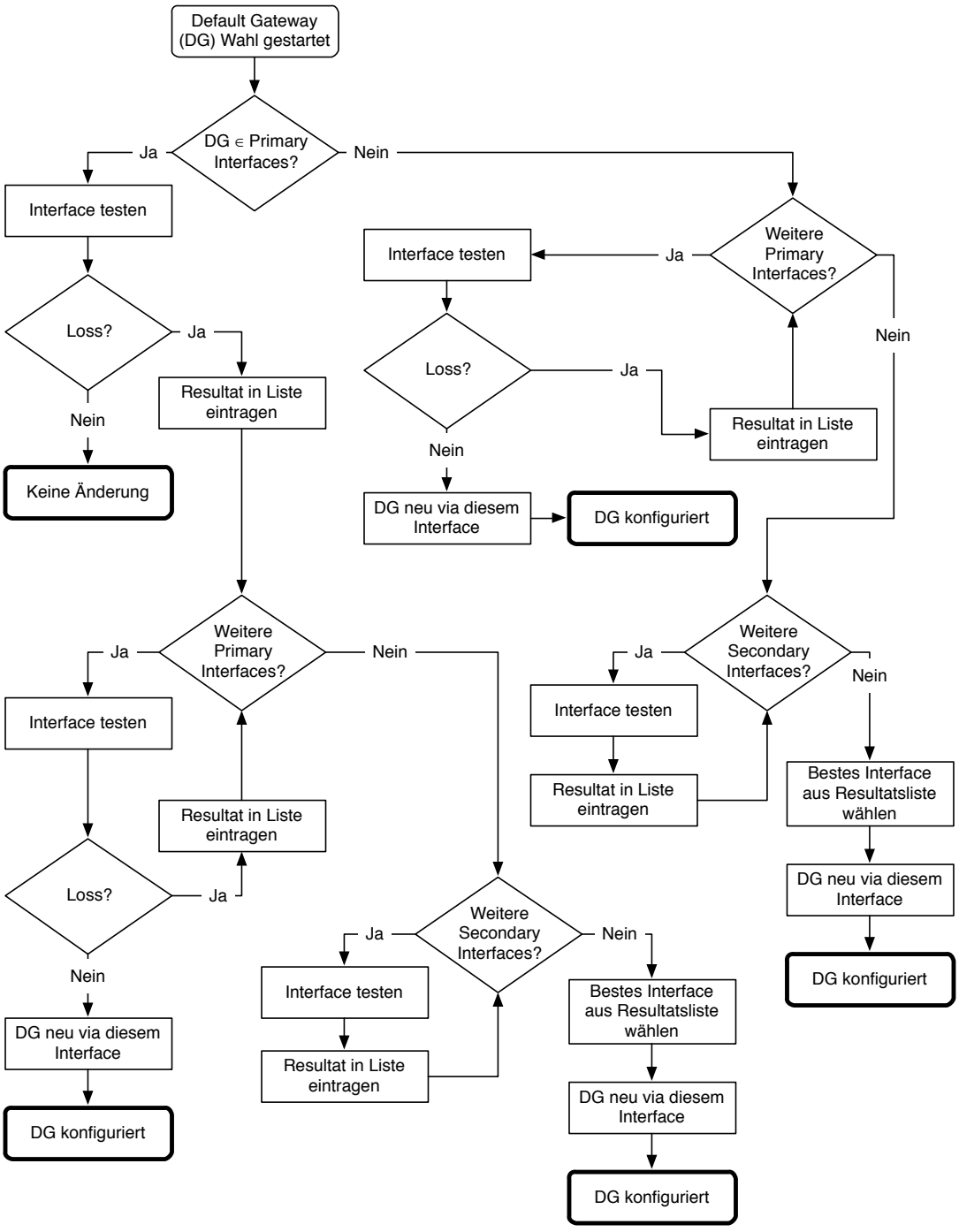

Abbildung 5.5: Ablauf Default-Gateway-Bestimmung

Grundsätzlich wird zwischen primären Interfaces (wie z.B. eine kabelgebundene Anbindung oder auch WLAN) und sekundären Interfaces (Verbindung über das Mobilfun-

knetz) unterschieden. Sofern ein primäres Interface ohne Packet-Loss verfügbar ist, soll der dazugehörige Router als Default-Gateway verwendet werden. Für den Fall, dass kein primäres Interface ohne Packet Loss verfübar ist, soll die zuverlässigste Route aus allen (primären und sekundären) Interfaces ausgewählt werden. Gibt es mehrere Interfaces mit gleichem Packet-Loss, wird das Interface mit der geringsten Latenz bevorzugt.

Um die Qualität eines Interfaces zu beurteilen, werden über das zu testende Interface 5 Ping-Pakete an den SOCKS-Server gesendet. Aus den Anforderungen erfolgt, dass möglichst wenig Overhead produziert werden soll (vgl. Abschnitt 3.1.1). Die folgende Rechnung soll verdeutlichen, dass der verursachte Traffic dieser Bedingung gerecht wird. Wird der Ping alle 10 Sekunden mit der minimalen Paketgrösse (64 Byte) durchgeführt, resultiert daraus der folgende Traffic:

$$\text{Traffic pro Link} = 5 * \frac{2 * 64\text{B}}{10\text{s}} = 64\frac{\text{B}}{\text{s}} = 230.4\frac{\text{kB}}{\text{h}}$$

Es darf vermutet werden, dass zwischen Latenz bzw. Packet Loss und Signalstärke ein Zusammenhang besteht. Deshalb werden in einer Datenbank jeweils die Ergebnisse der Messung sowie die Signalstärke (gemäss Abschnitt 4.9) der einzelnen Links dokumentiert. In einer Fortsetzung dieser Arbeit könnten diese Daten ausgewertet werden. Je nach Resultat könnte auf das Pingen verzichtet werden und stattdessen die Wahl auf Basis der Signalstärke getroffen werden.

Der Default-Gateway wird nach Auswahl des besten Interfaces mittels `ip`-Tool in der Routingtabelle "default" konfiguriert.

5.3 MPTCP

5.3.1 Design-Goals

MPTCP beruht auf drei wesentlichen Design-Zielen, welche in der Referenzimplementation beachtet wurden[8]:

Design Goal 1: MPTCP soll fair gegenüber normalem TCP sein

Das erste Designziel beschreibt, dass MPTCP in einem Engpass soviel Kapazität wie normales TCP brauchen soll, egal wie viele Sub-Flows bestehen. Normales TCP soll also nicht benachteiligt werden.

Design Goal 2: MPTCP soll effiziente Pfade nutzen

MPTCP soll die Pfade nutzen, die am wenigsten belastet sind (Least-Congested-Paths), was zu der effizientesten Verteilung und Auslastung führt.

Design Goal 3: MPTCP soll mindestens so gut wie TCP sein

(a) Ein MPTCP-Benutzer soll mindestens soviel Throughput bekommen, wie auf dem besten verfügbaren Pfad mit TCP möglich wäre.

(b) Ein MPTCP-Flow soll nicht mehr Kapazität auf einem Link brauchen, als ein einzelner TCP-Pfad nutzen würde.

5.3.2 Verbindungsaufbau

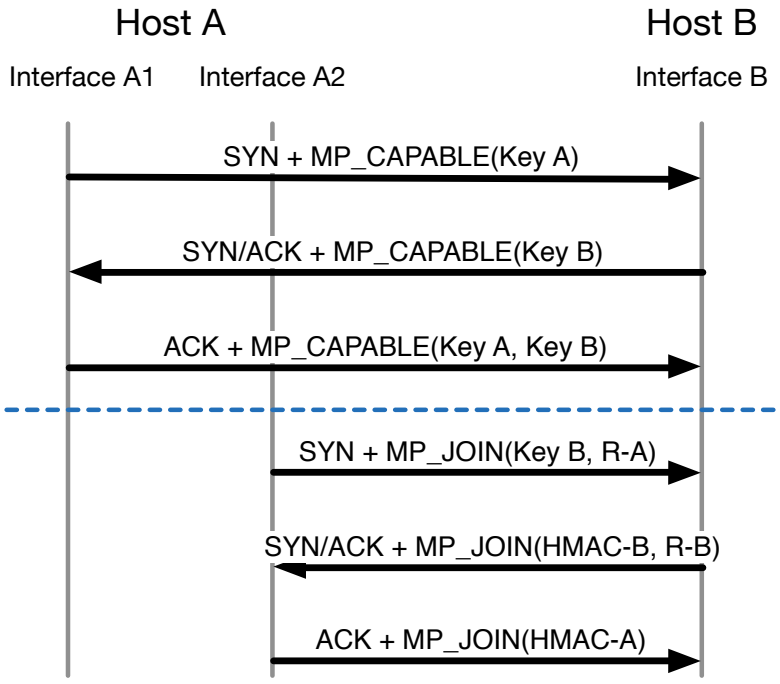

Abbildung 5.6: TCP Handshake mit der MPTCP Erweiterung

Wie in Abbildung 5.6 ersichtlich ist, teilt das verbindungsaufbauende System (Requester) beim TCP Verbindungsaufbau (`SYN`) mittels der `MP_CAPABLE` Option dem Zielsystem (Responder) mit, das MPTCP unterstützt wird. Zusätzlich wird ein Schlüssel (Key A) mitgeschickt, der für das spätere Zuschalten weiterer Verbindungen (Sub-Flows) benötigt wird. Unterstützt der Responder die Erweiterung, wird in der darauffolgenden Antwort (`SYN/ACK`) auch die `MP_CAPABLE` Option sowie ein eigener Schlüssel (Key B) integriert. Standardmässig werden die Schlüssel im Klartext ausgetauscht, was ein Sicherheitsrisiko darstellt. Es gibt allerdings Vorschläge, den Schlüsselaustausch durch den Einsatz von SSL/TLS abzusichern[10].

Dem Requester steht es nach dem Verbindungsbau offen, eine beliebige Anzahl an Sub-Flows zu öffnen. Dazu werden die `MP_CAPABLE`-Option sowie mittels der beiden Schlüssel (Key A und Key B) berechnete Identifier genutzt. Auf die genaue Berechnung letzterer wird hier auf Grund deren Komplexität nicht weiter eingegangen.

MPTCP verteilt den Verkehr mittels Congestion-Control wie TCP, mit dem Unterschied, dass für jeden Sub-Flow der Slow-Start-Algorithmus angewendet wird. MPTCP füllt

zuerst das Congestion-Window des Sub-Flows mit der niedrigsten RTT, danach wird auf den Sub-Flow mit der nächsthöheren RTT umgeschaltet. Diese Information kann dem TCP-Stack entnommen werden; die Timestamp-Extension[7] wird von MPTCP genutzt, um die vorhandenen Sub-Flows zu vergleichen und so die Interface-Präferenz zu bestimmen.

5.4 Redundanz / Skalierung

In der beschriebenen Ausführung (Abbildung 5.8) ist der Relay-Server ein Single Point of Failure (SPOF). Ausserdem skaliert die Lösung nicht, weil ein Relay-Server sämtliche Multifunktions-Wi-Fi-Router bedienen muss.

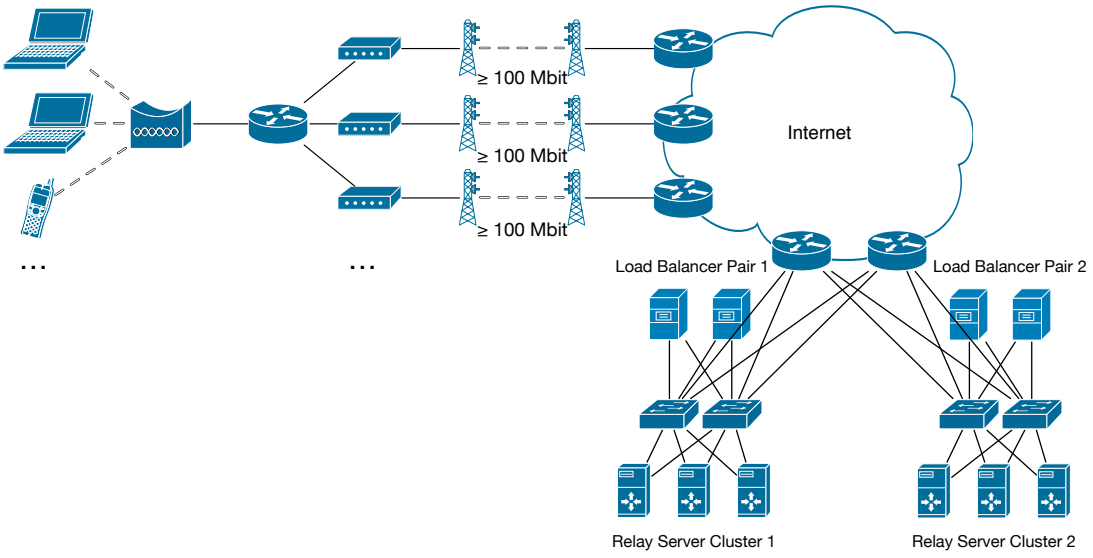

Abbildung 5.7: Erweiterte, skalierbare Netzwerktopologie mit erhöhter Ausfallsicherheit

Das Problem der Skalierung kann durch das Bilden von einem oder mehreren Relay-Server Clustern angegangen werden. Sind die vorgeschaltenen Loadbalancer dabei pärchenweise angeordnet und redundant an das Internet angebunden, werden in diesem Zuge sämtliche SPOFs eliminiert. Die Aufteilung der Multifunktions-Wi-Fi-Router auf die Cluster kann dabei beispielsweise per Round-Robin-Domain Name System (DNS) geregelt werden.

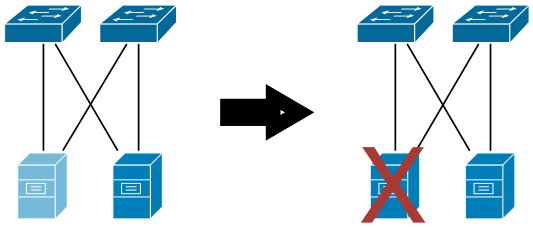

Abbildung 5.8: Failover auf Backup-Loadbalancer

Verliert der Backup-Loadbalancer die Verbindung zum aktiven Loadbalancer, muss von einem Ausfall ausgegangen werden. Der Backup-Loadbalancer übernimmt bei einem Ausfall automatisch die geteilte IP Adresse und wird zum aktiven Loadbalancer. Auch die aktiven TCP Sessions könnten mittels geeigneter Verfahren übernommen werden, damit kein Verbindungsneuaufbau erforderlich ist.

6 Testing

6.1 Test Cases

6.1.1 TC1: Link während Datei-Download hinzufügen

Beschreibung

Es soll getestet werden, ob ein neu aufgeschalteter Link vom System genutzt wird.

Es wird ein Mobilfunkantennen-Adapter physisch hinzugefügt/eingesteckt. Dies simuliert die Situation, dass ein Link während dem Betrieb hinzugefügt wird.

Preconditions

- es ist bereits ein Link vorhanden
- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).

Postconditions

- Datei wurde vollständig auf Zielserver hochgeladen

Testschritte

Step	Action	Expected System Response
1.	Datei-Download starten	-
2.	Aufschalten des Links X	Auf dem hinzugefügten Link X ist Traffic messbar

6.1.2 TC2: Link während Datei-Download entfernen

Beschreibung

Verteilt das System den Traffic bei einem Ausfall eines Links automatisch auf die verbleibenden Links? Im Unterschied zu *«6.1.1 TC1: Link während Datei-Download hinzufügen»* soll dieser Fall testen, ob bei einem Ausfall eines Links Daten verloren gehen.

Ein Link wird physisch entfernt/ausgesteckt

Preconditions

- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).

Postconditions

- Datei wurde vollständig auf Zielserver hochgeladen

Testschritte

Step	Action	Expected System Response
1.	Datei-Download starten	-
2.	Während dem Download wird Link X entfernt	Es gibt während der Übertragung keinen Unterbruch

6.1.3 TC3: Nutzung Links prüfen

Beschreibung

Dieser Testfall prüft, ob bei einem Dateidownload alle Links verwendet werden und somit, ob Multipathing stattfindet

Preconditions

- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).

Postconditions

- Datei wurde vollständig auf Zielsystem übertragen

Testschritte

Step	Action	Expected System Response
1.	Dateidownload starten	Alle Links haben Traffic

6.1.4 TC4: Abschirmung Link während Datei-Download

Beschreibung

Hat ein Link einen sehr schwachen oder gar keinen Throughput, soll er vom Multipathing ausgeschlossen werden. In *«6.1.2 TC2: Link während Datei-Download entfernen»* wurde der Link physisch entfernt. Hier ist der Link physisch noch vorhanden, kann keine Daten mehr senden. Dieser Fall testet, ob auch sehr schwache Links ausgeschlossen werden.

Preconditions

- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).

Postconditions

- Datei wurde vollständig auf Endgerät geladen hochgeladen

Testschritte

Step	Action	Expected System Response
1.	Dateidownload starten	-
2.	Während dem Download wird Link X abgeschirmt	auf Links X wird kein Traffic gemessen

6.1.5 TC5: Abschirmung eines Links aufheben

Beschreibung

Analog zu *«6.1.4 TC4: Abschirmung Link während Datei-Download»* soll hier getestet werden, ob ein Link wieder genutzt wird, wenn sich dessen Signalqualität nach einer gewissen Zeit wieder verbessert.

Im Testfall wird dies simuliert, indem die Abschirmung eines zuvor abgeschirmten Adapters aufgehoben wird.

Verbessert sich die Signalqualität eines Links, welcher zu Beginn einer Übertragung kein oder ein sehr schwaches Signal hat, soll dieser Link im Verlaufe der Übertragung dazugeschaltet werden.

Preconditions

- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).
- Es sind mehrere Links vorhanden
- Link X ist abgeschirmt

Postconditions

- Datei wurde vollständig auf Zielserver hochgeladen

Testschritte

Step	Action	Expected System Response
1.	Dateidownload starten; während dem Download wird bei Link X die Abschirmung aufgehoben	es gibt keinen Unterbruch
2.	Traffic pro Link messen	auf Link X ist Traffic messbar

6.1.6 TC6: Multipathing testen

Beschreibung

Dieser Whitebox-Test soll prüfen, ob auf den Modem-Interfaces nur MPTCP-Traffic gemessen wird.

Preconditions

- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).
- Es sind mehrere Links vorhanden

Testschritte

Step	Action	Expected System Response
1.	TCP- und UDP-Traffic generieren	Auf den Interfaces ist nur TCP-Traffic mit der Destination-IP der Middlebox sichtbar, es gibt keinen Paket-Loss

6.1.7 TC7: TCP-Verkehr testen

Beschreibung

Dieser Whitebox-Test soll sicherstellen, dass der TCP-Traffic nicht über den VPN-Tunnel geschickt wird. Ist dies sichergestellt, kann unter Voraussetzung von *«6.1.7 TC7: TCP-Verkehr testen»* davon ausgegangen werden, dass der SOCKS-Proxy für TCP-Traffic genutzt wird.

Preconditions

- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).
- Es sind mehrere Links vorhanden

Testschritte

Step	Action	Expected System Response
1.	Paketüberwachung auf Tunnel-Interface starten	-
2.	TCP-Traffic mittels Dateidownload generieren	Auf dem Tunnel-Interface ist kein TCP-Traffic sichtbar

6.1.8 TC8: Voice over IP (VoIP)

Beschreibung

Da VoIP anfällig auf eine auf hohe Latenz ist, wird getestet, ob bei einem Ausfall die Verteilung auf die verbleibenden Links genügend schnell ist, ohne dass dies im Gespräch bemerkt wird.

Preconditions

- Link muss bei Normalbedingungen gute Qualität (Access-Technology und Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).
- Es sind mehrere Links vorhanden
- Quality-of-Service bleibt mehr oder weniger konstant

Testschritte

Step	Action	Expected System Response
1.	VoIP-Gespräch wird gestartet	Verzögerung ist normal
2.	Link X wird mit `multipath backup` ausgeschlossen	es ist keine Verzögerung hörbar

6.1.9 TC9: Multipathing-Durchsatz testen

Beschreibung

Der effektive Throughput soll der Summe derThroughputs aller Links entsprechen.

Preconditions

- Throughput der einzelnen Links ist bekannt
- Es sind mehrere Links vorhanden

Testschritte

Step	Action	Expected System Response
1.	TCP-Traffic mittels Dateidownload generieren	Der erzielte Throughput sollte der Summe aller Throughputs der einzelnen Links entsprechen

6.1.10 TC10: Default-Gateway: Ausfall des stärkeren Links

Beschreibung

Dieser Test untersucht den Fall, dass vom System von zwei Links (Link X und Link Y) zuerst der stärkere Link X als Default-Gateway gesetzt wurde und später ausfällt. Damit soll getestet werden, ob die Gateway-Umschaltung auf Link Y aufgrund der Latenz funktioniert.

Standardmässig würde der zuerst vorhandene Link als Default-Gateway gesetzt. Mit der erarbeiteten Erweiterung werden die Links gemessen, worauf der Link mit der geringsten Latenz als Default-Gateway gesetzt wird.

Preconditions

- Link X ist als Default-Gateway gesetzt

Postconditions

- Link Y ist als Default-Gateway gesetzt

Testschritte

Step	Action	Expected System Response
1.	Link X abschirmen	Latenz von X höher als von Y. Default-Gateway wird auf Link Y umgeschaltet

6.1.11 TC11: Präferenz WLAN

Beschreibung

Es soll getestet werden, ob das System WLAN bevorzugt.

Preconditions

- Es sind mehrere Links verfügbar
- WLAN ist als Link vorhanden
- Maximaler Throughput des WLAN-Links bekannt
- WLAN-Link muss bei Normalbedingungen gute Qualität (Signalstärke) haben. Somit wird er vom System nicht frühzeitig ausgeschlossen (z.B. über `multipath backup`).

Testschritte

Step	Action	Expected System Response
1.	Traffic produzieren, die den max. Throughput des WLAN-Links nicht übersteigt	Traffic nur auf WLAN-Interface sichtbar

6.2 Testresultate

6.2.1 TC1: Link während Datei-Download hinzufügen

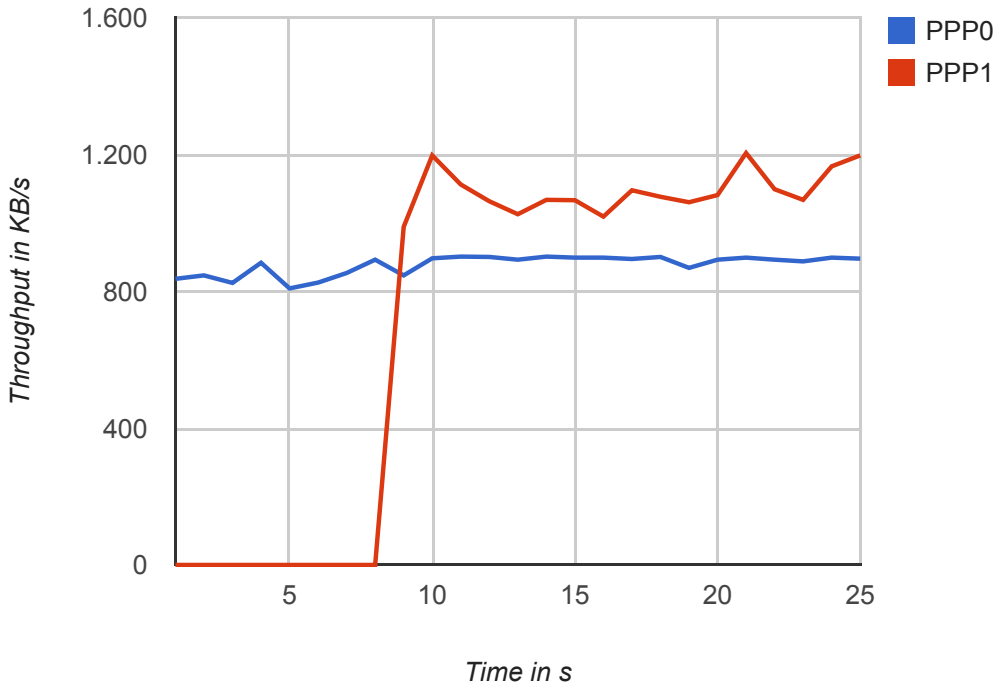

Abbildung 6.1: Throughput der einzelnen Links während Daten-Übertragung. Nach ca. 8 Sekunden wurde das Interface PPP1 dazugeschaltet.

Wie im Test-Case beschrieben, wurde während der Daten-Übertragung ein Link dazugeschaltet. Gut ersichtlich ist in der Abbildung 6.1, dass der Link sofort für die Übertragung genutzt wurde. Der Throughput hat sich dadurch mehr als verdoppelt.

6.2.2 TC2: Link während Datei-Download entfernen

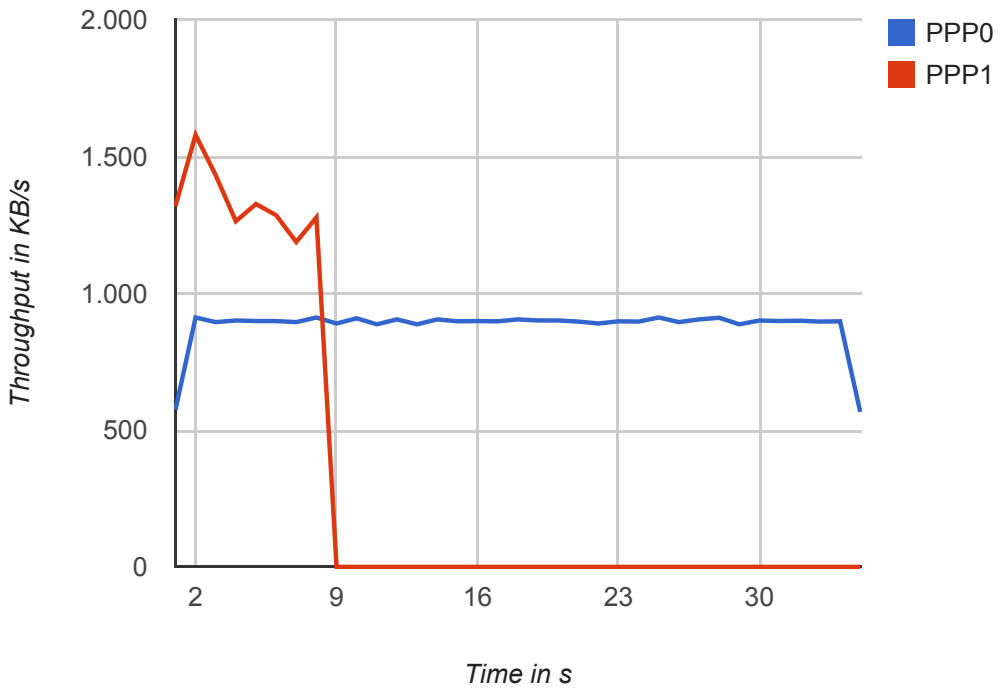

Abbildung 6.2: Throughput der einzelnen Links während Daten-Übertragung. Nach ca. 8 Sekunden wurde das Interface PPP1 entfernt.

Während der Daten-Übertragung wurde ein Link entfernt. In der Abbildung 6.2 ist ersichtlich, dass der Link naturgemäss keinen Throughput mehr hatte. Der Throughput des anderen Links wurde nicht beeinflusst und der Download wurde ohne Unterbrechung fortgesetzt.

6.2.3 TC3: Nutzung Links prüfen

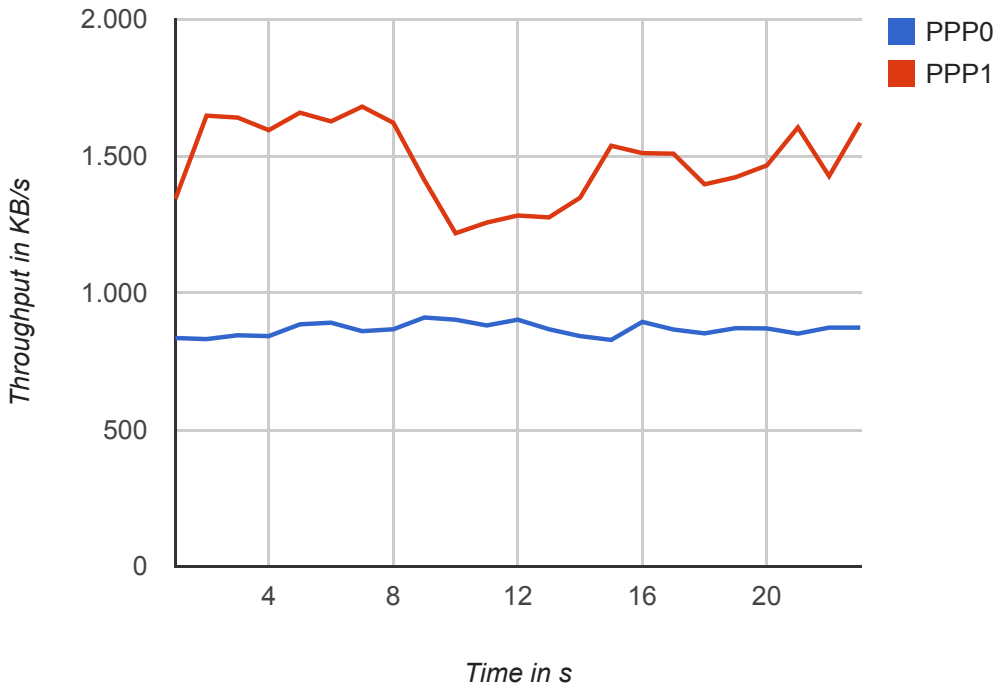

Abbildung 6.3: Throughput der einzelnen Links während Daten-Übertragung.

Bei diesem Test wurde der Throughput der einzelnen Links währen einer Daten-Übertragung gemessen. Zu bemerken ist, dass beide Links beansprucht wurden.

6.2.4 TC4: Abschirmung Link während Datei-Download

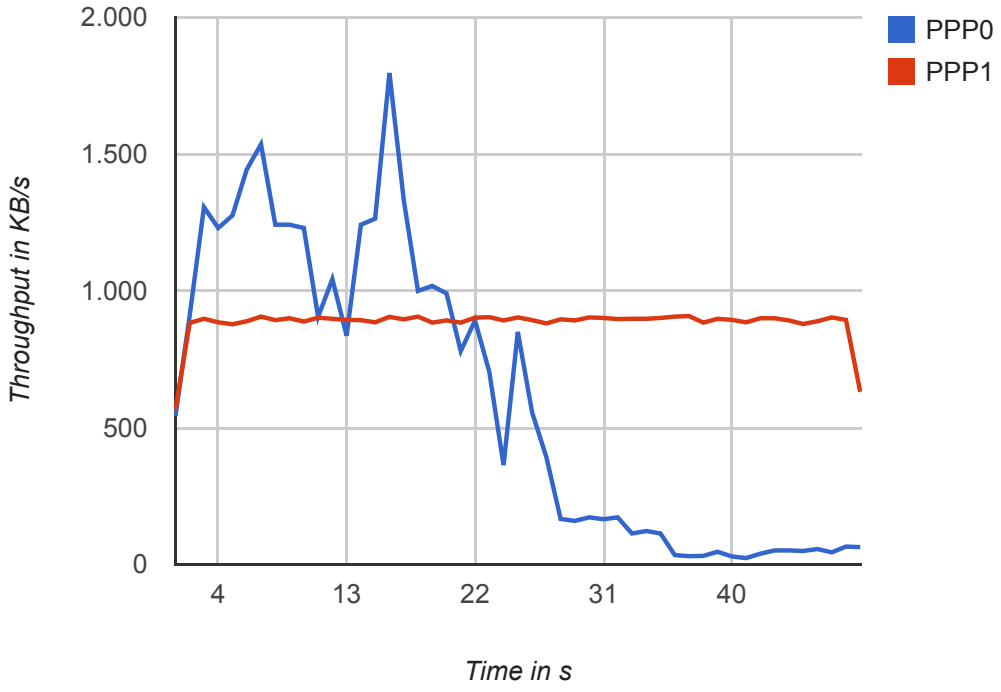

Abbildung 6.4: Throughput der einzelnen Links während Daten-Übertragung. Nach ca. 20 Sekunden wurde mit der Dämpfung des Interfaces PPP0 begonnen.

Gemäss Testbeschreibung aus Abschnitt 6.1.4 wurde einer der beiden Links (PPP0) bei laufender Übertragung gestört. Der Throughput der Links wurde während dem Versuch kontinuierlich aufgezeichnet.

Es fällt auf, dass der Throughput des gestörten Links zwar sinkt, allerdings wird er nie ganz ausgeschlossen. Dies hat jedoch keinen negativen Einfluss auf den gesamten Throughput.

6.2.5 TC5: Abschirmung eines Links aufheben

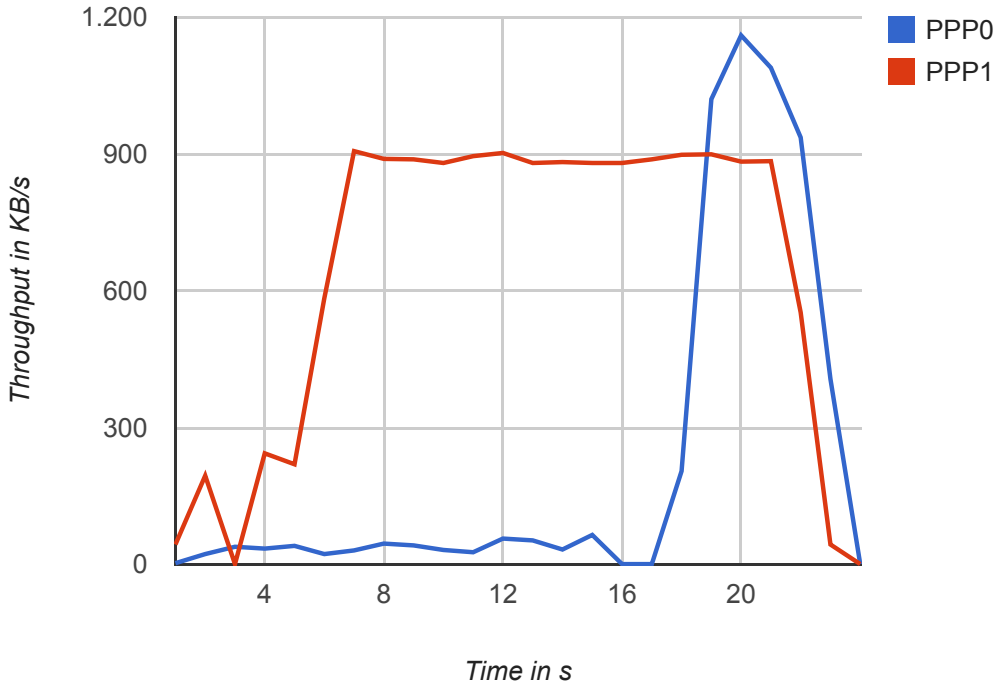

Abbildung 6.5: Throughput der einzelnen Links während Daten-Übertragung. Nach ca. 18 Sekunden wurde die Dämpfung des Interfaces PPP0 aufgehoben.

Der anfänglich gedämpfte Link PPP0 hatte einen sehr geringen Throughput. Nach dem Aufheben der Dämpfung wurde der Link sofort genutzt und der Gesamt-Throughput stieg dementsprechend.

6.2.6 TC6: Multipathing testen

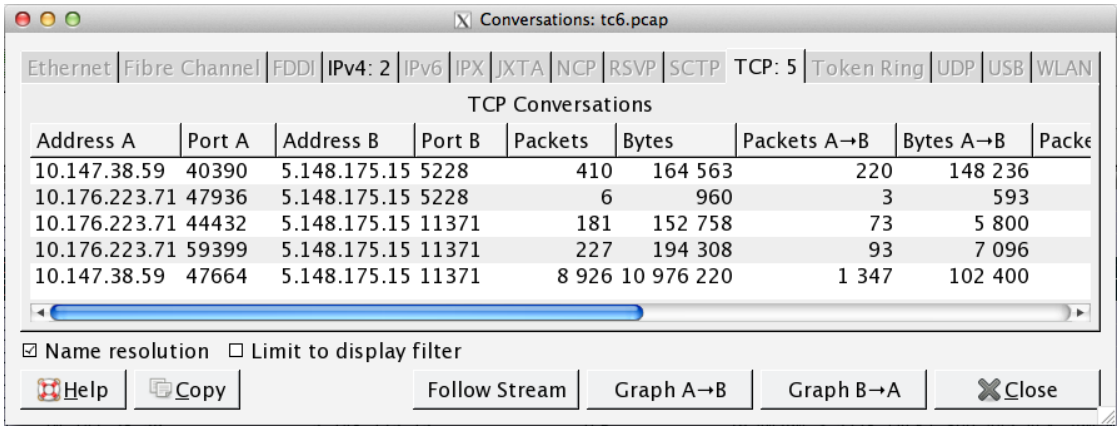

Abbildung 6.6: Während dem Test aufgezeichnete Konversationen

In der Test-VM wurde sowohl TCP- als auch UDP-Traffic generiert. Währenddessen wurde der Traffic auf den beiden Interfaces aufgezeichnet. Bei der Analyse der Konversations-Liste ist ersichtlich, dass das Gerät wie erwartet nur mit dem SOCKS-Proxy (`5.148.175.15:11371`) und dem VPN-Server (`5.148.175.15:5528`) Daten ausgetauscht hat. Weiter wurde mittels dem Query `!tcp.options.mptcp.subtype` nach Paketen ohne MPTCP-Optionen gesucht, was erwartungsgemäss zu keinen Treffern führte.

6.2.7 TC7: TCP-Verkehr testen

Protocol	% Packets	Packets	% Bytes
Frame	100.00 %	4	100.00 %
Raw packet data	100.00 %	4	100.00 %
Internet Protocol Version 4	100.00 %	4	100.00 %
User Datagram Protocol	100.00 %	4	100.00 %
Domain Name Service	100.00 %	4	100.00 %

Abbildung 6.7: Protokollhierarchie der Pakete, die während dem Test auf dem VPN-Tunnel-Interface aufgezeichneten wurden

Der TCP-Traffic wurde durch Downloaden einer Datei generiert. Auf dem VPN-Tunnel-Interface war dabei zwar Traffic sichtbar, es handelte sich dabei allerdings lediglich um die DNS-Anfrage, um den Hostnamen des Zielservers aufzulösen. Dies entspricht den Erwartungen.

6.2.8 TC9: Multipathing-Durchsatz testen

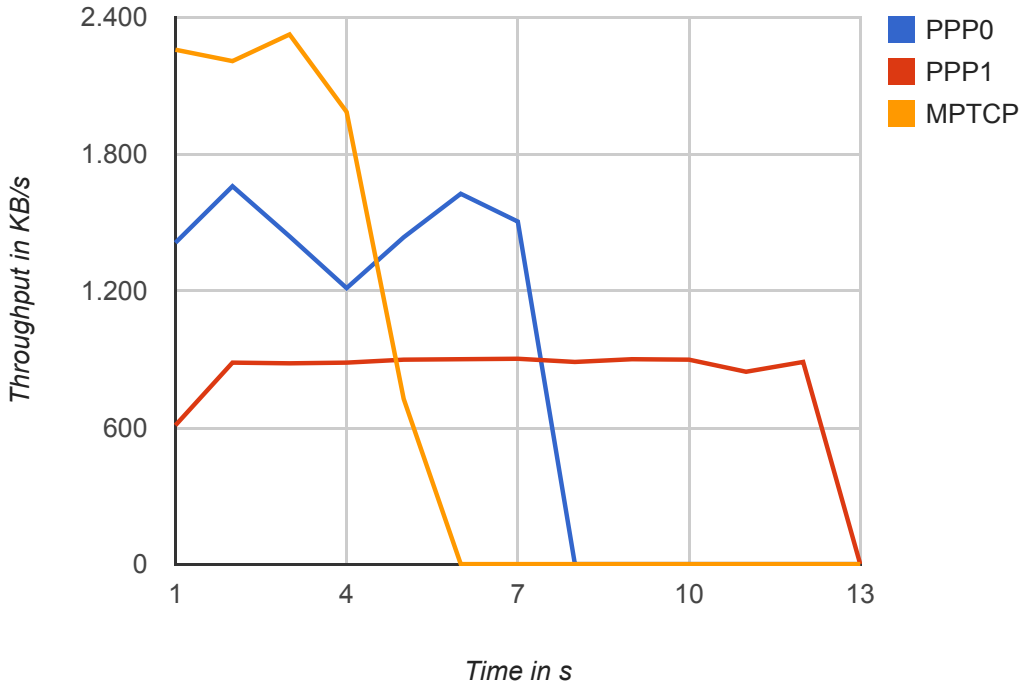

Abbildung 6.8: Vergleich Throughput MPTCP und TCP während Übertragung von 10 MB

Um den Throughput zu testen, wurden jeweils 10 MB mittels `iperf` übertragen. Dabei wurden zuerst die beiden Interfaces (PPP0, PPP1) separat direkt getestet.

Durch den ersten Link (PPP0) konnte die Datei mit einer durchschnittlichen Downloadgeschwindigkeit von 1428 KB/s übertragen. Beim zweiten Link (PPP1) wurde ein durchschnittlicher Throughput von 880 KB/s erzielt.

Mit der Nutzung der MPTCP-Implementation konnte ein Durchsatz von 2143 KB/s erzielt werden. Das entspricht über 90% des theoretischen Maximums.

6.2.9 TC10: Default-Gateway: Ausfall des stärkeren Links

```
1 default dev ppp0  scope link
2 10.10.10.1 via 10.10.10.5 dev tun0
3 10.10.10.5 dev tun0  proto kernel  scope link  src
    10.10.10.6
4 10.64.64.64 dev ppp0  proto kernel  scope link  src
    10.130.154.116
5 10.64.64.65 dev ppp1  proto kernel  scope link  src
    10.176.238.42
6 192.168.122.0/24 dev virbr0  proto kernel  scope link  src
    192.168.122.1
```

Listing 6.1: `ip route show` beim Start des Tests

```
1 default dev ppp1  scope link
2 10.10.10.1 via 10.10.10.5 dev tun0
3 10.10.10.5 dev tun0  proto kernel  scope link  src
    10.10.10.6
4 10.64.64.64 dev ppp0  proto kernel  scope link  src
    10.130.154.116
5 10.64.64.65 dev ppp1  proto kernel  scope link  src
    10.176.238.42
6 192.168.122.0/24 dev virbr0  proto kernel  scope link  src
    192.168.122.1
```

Listing 6.2: `ip route show` nach Abschirmen von PPP0

Zu Beginn des Tests wurde PPP0 als Default-Gateway genutzt. Nach dem Abschirmen dieses Interfaces, wurde der Default-Gateway des Systems automatisch auf das Interface PPP1 gelegt.

6.2.10 TC11: Präferenz WLAN

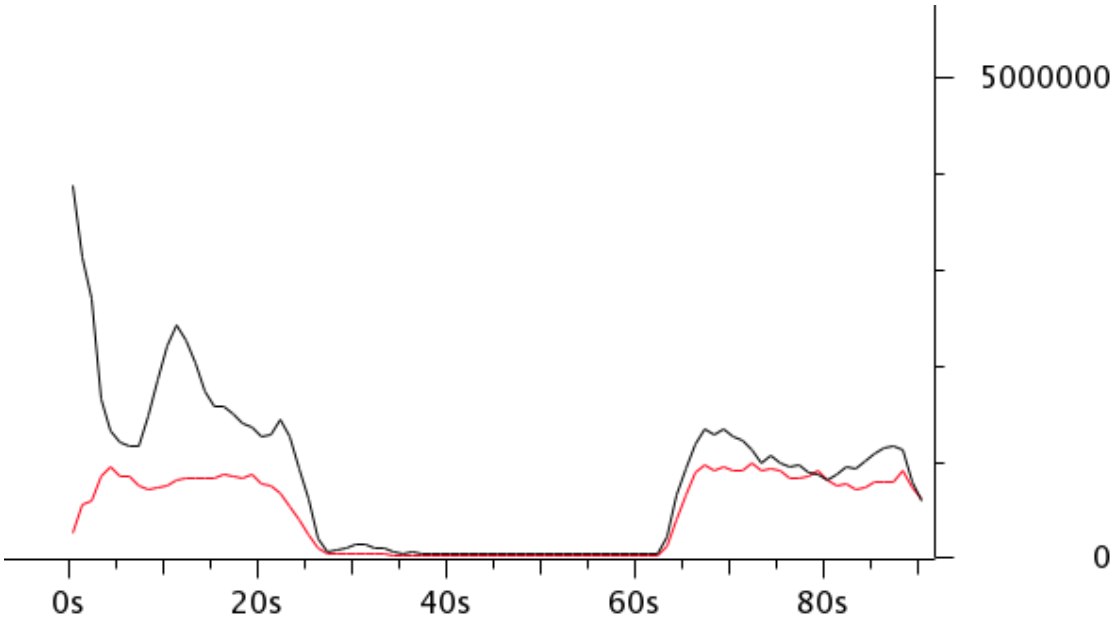

Abbildung 6.9: *IO Graph*

In einem ersten Test mit dem zugeschalteten WLAN-Interface, wurde etwas bemerkbares festgestellt: Der Throughput sank innerhalb weniger Sekunden auf ca. 100 KB/s. Zeitweise stagnierte der Download vollständig.

Analysen und Diskussionen auf der Mailingliste[1] bestätigten, dass es sich hierbei um einen Bug in der aktuellen MPTCP-Implementation handelt: MPTCP berechnet die Speichernutzung je nach WLAN-Treiber falsch.

Das Problem konnte durch Anpassen des `tcp_adv_win_scale`-Parameters entschärft werden. Der Test wurde mit verschiedenen Werten des Parameters wiederholt. Ein Wert von `-4` erzielte dabei die besten Resultate:

[1] N. Caspar, Dezember 2013: Low total throughput with 3G connection involved: `https://listes-2.sipr.ucl.ac.be/sympa/arc/mptcp-dev/2013-12/msg00036.html`

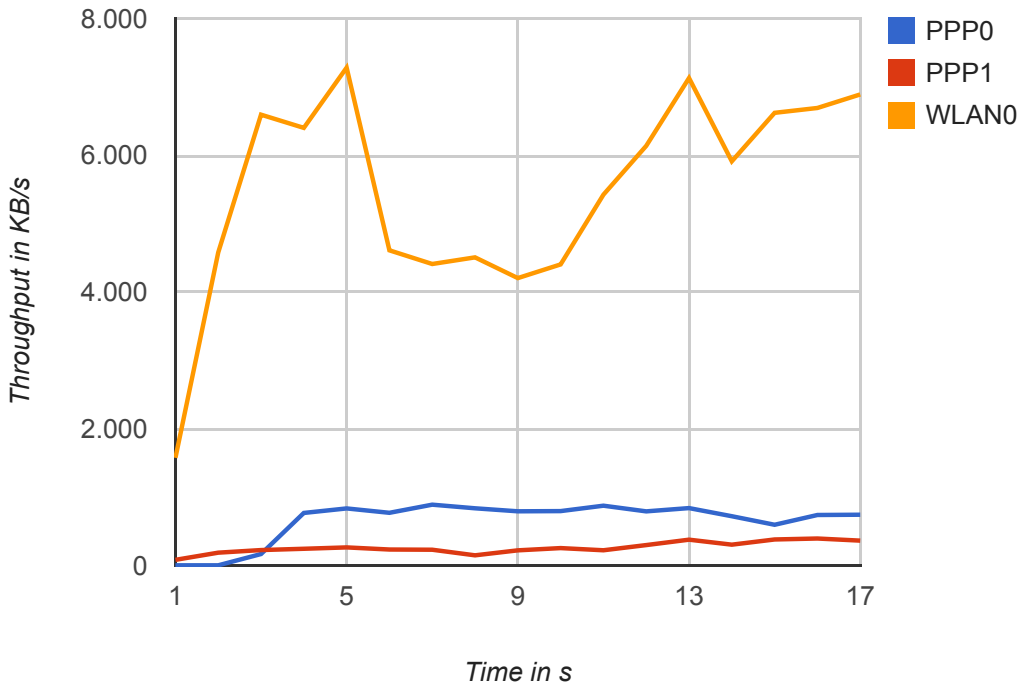

Abbildung 6.10: Throughput der einzelnen Links während Daten-Übertragung.

Falls eine WLAN-Verbindung vorhanden ist, wird diese nun stark beansprucht. Die restlichen Links werden allerdings nicht komplett ignoriert, weil kein entsprechender Mechanismus implementiert wurde.

7 Implementation

7.1 Server

Über LTE können in der Theorie beim Download bis zu 100 Mbit/s pro Pfad erreicht werden, weshalb SOCKS- und VPN-Server mit einem Gigabit-Uplink ausgestattet werden.

7.1.1 SOCKS-Server

Wie in der Analyse evaluiert, kommt *Dante* in der Version `v1.4.0-pre2` zum Einsatz.

Nur minimale Anpassungen müssen an der Standard-Konfiguration vorgenommen werden: Als Server-Port wurde `11371` gewählt, weil dieser Port im Netz der Hochschule für Technik Rapperswil (HSR) (im Gegensatz zum Standardport `1080`) nicht geblockt wird.

Die vollständige Konfiguration ist im Anhang unter *«E.1 Dante»* ersichtlich.

7.1.2 VPN-Server

Zum Einsatz kommt *OpenVPN* in der Version `2.2.1`.

Die Konfiguration wird wie in der Evaluation mittels Puppet bzw. dem Puppet-Modul *puppet-openvpn* generiert. Dazu kommt das unter *«E.2.1 Puppet»* beschriebene Puppet-Rezept zum Einsatz, was zu der unter *«E.2.2 Generierte Server-Konfiguration»* auffindbaren Konfiguration führt.

Wichtig ist, dass vom standardmässigen UDP auf TCP umgestellt wird. Als Port wurde `5228` gewählt, weil dieser im Gegensatz zum Standardport `1194` wiederum von der Firewall der HSR nicht geblockt wird.

Im zweiten Schritt muss ein NAT eingerichtet werden, damit der Zugang ins Internet via VPN-Tunnel korrekt funktioniert. Dies wird mittels dem Puppet-Modul *puppetlabs-*

firewall[1] und dem unter *«E.3.1 Puppet»* beschriebenen Puppet-Rezept erledigt, was zu der unter *«E.3.2 Generierte Konfiguration»* aufgeführten Konfiguration führt.

7.2 Multifunktions-Wi-Fi-Router

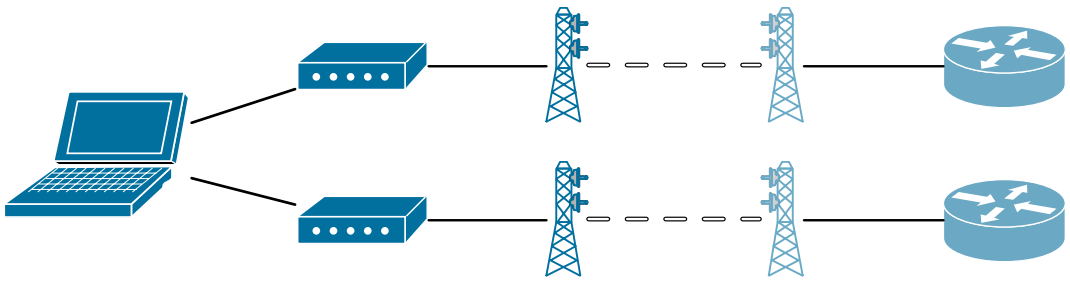

Abbildung 7.1: Physikalische Topologie des Multifunktions-Wi-Fi-Router-Prototyps

Im Versuchsaufbau wird der Multifunktions-Wi-Fi-Router durch einen Laptop mit zweier USB-LTE-Modems (*4G Systems XS-Stick W100 LTE Surfstick*) repräsentiert. Zum Einsatz kommen die SIM-Karten zweier unterschiedlicher Mobilfunkanbieter (*Swisscom* und *Orange*). Die orchestrierte Zusammenstellung ist folglich über zwei voneinander unabhängigen Pfaden mit dem Internet verbunden.

[1]Puppet Labs, September 2013: Puppet Firewall Module: `https://forge.puppetlabs.com/puppetlabs/firewall`

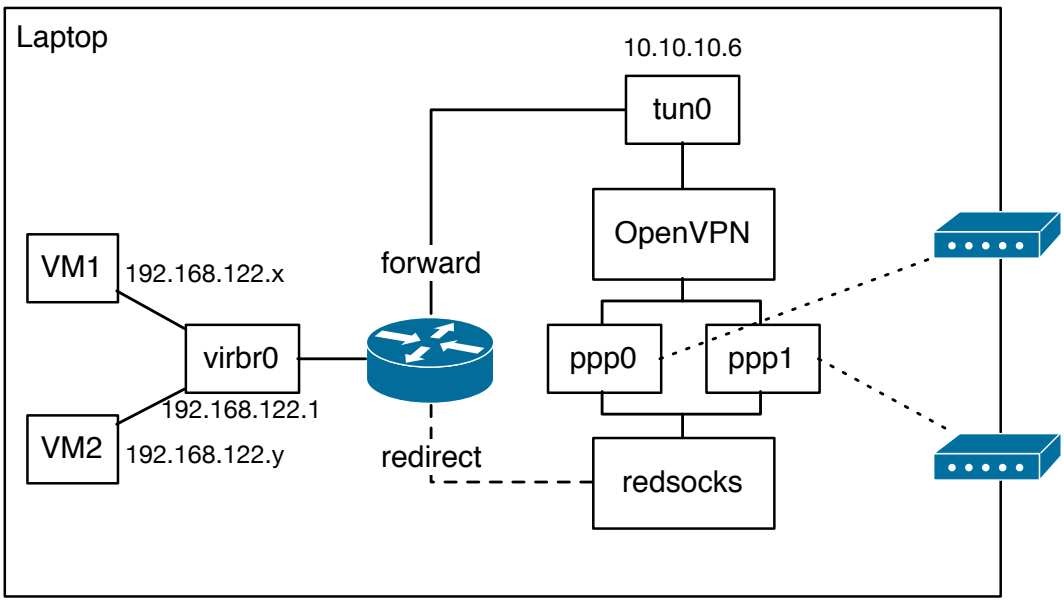

Abbildung 7.2: Logische Netzwerkstruktur des Multifunktions-Wi-Fi-Router-Prototyps

Die Clientgeräte werden durch virtuelle Maschinen, welche direkt auf besagtem Laptop betrieben werden, simuliert.

7.2.1 Modems

Für die Konfiguration der Modems wird der PPP-Daemon verwendet. Dazu muss eine sogenannte "Peer-Konfiguration" erstellt werden. Weiter muss ein entsprechendes "Chat-Script" erstellt werden, welches für den Verbindungsaufbau zuständig ist. Die entsprechenden Konfigurationen sind im Anhang unter *«E.5 Modems»* aufgeführt.

Die Verbindung kann mittels `pon swisscom` bzw. `pon orange` aufgebaut werden. Das Trennen der Verbindung geschieht durch das Beenden des entsprechenden Prozesses oder mittels `poff swisscom` bzw. `poff orange`.

Die Peer-Konfiguration enthält jeweils den Device-Pfad zum Modem. Weiter muss der Verbindungsaufbau manuell gestartet werden. Will man die Modems dynamisch konfigurieren und den Verbindungsaufbau automatisch starten, kann man das mittels eines entsprechenden `udev`-Triggers[2] erledigen:

[2]udev - dynamic device management: `http://manpages.ubuntu.com/manpages/karmic/en/man7/udev.7.html`

```
1 ACTION=="add", ATTRS{idVendor}=="1bbb",
    ATTRS{idProduct}=="011e",
    RUN+="/usr/local/sbin/autoconnect-usb-modem"
```

Listing 7.1: `/etc/udev/rules.d/90-usb-modems.rules`

Das Script `/usr/local/sbin/autoconnect-usb-modem` wäre dann dafür zuständig, die Konfiguration dynamisch zu erstellen und die Verbindung automatisiert aufzubauen.

7.2.2 VPN

Für das Generieren der *OpenVPN* Client-Konfiguration ist ebenfalls das erwähnte Puppet-Modul *puppet-openvpn* zuständig. Das Vorgehen wurde bereits im Rahmen der Evaluation ausfühlich erläutert (*«4.7 Client-Definition im Puppet-Rezept»*). Die generierte Konfiguration ist unter *«E.2.3 Generierte Client-Konfiguration»* aufgeführt.

7.2.3 Routing

Die Konfiguration der Paket-Firewall `iptables` gemäss *«5.2.1 Routing»* wird über ein entwickeltes *Ruby*-Script erledigt (`routing-setup.rb`). Mit folgendem Aufruf wird das Routing auf dem Testgerät eingerichtet:

```
sudo ./routing-setup.rb --vpn-gateway=10.10.10.5
   --socks-ip=<SOCKS_SERVER_IP> --socks-port=11371
   --nat-net=192.168.122.0/24 --nat-interface=virbr0
```

Durch das Hinzufügen des `--verbose` Flags, kann die Ausgabe um die ausgeführten Befehle ergänzt werden.

Mit dem `--log` Flag wird das Script angewiesen, sogenannte Log-Jumps via `iptables` einzufügen, was beim Debugging nützlich sein kann. `iptables` wird angewiesen, an strategisch wichtigen Punkten via `syslog` ein Paket zu loggen. In der Standardeinstellung landen diese Log-Meldungen in der Datei `/var/log/kern.log`.

Mittels der `--dry-run` Option können alle Befehle angezeigt werden (wie mit `--verbose`), jedoch ohne sie tatsächlich auszuführen. Dies ist besonders während der Entwicklung auf einem Gerät ohne entsprechende NAT-Interfaces etc. interessant.

Mit dem Parameter `--help` kann eine Übersicht über alle verfügbaren Parameter angezeigt werden.

Das Script setzt automatisch die `rp_filter`-Einstellung für das Tunnel-Interface. Je nach System kann es notwendig sein, zusätzlich die globale Konfiguration anzupassen. Die entsprechende Konfiguration findet sich im Anhang unter *«E.6 Routing-Flags»*. Dort ist auch dokumentiert, wie das Forwarding aktiviert werden kann.

7.2.4 Virtuelle Maschinen

Für die virtuellen Maschinen wird die im Linux-

```
virt-install --name guest1 --ram 512 --disk
   path=/home/<user>/disks/guest1.raw,size=20 --network
   network:default --vnc --os-variant ubunturaring --cdrom
   /home/<user>/images/ubuntu-13.04-amd64.iso
```

8 Schlussfolgerungen

8.1 Erreichte Ziele

Das primäres Ziel dieser Arbeit, einen funktionierenden Prototypen zu erstellen, konnte erreicht werden. Die anfänglich formulierten Use Cases (vgl. Abschnitt 3.5) wurden umgesetzt und damit konnten die Anforderungen (vgl. Abschnitt 3) vollständig erfüllt werden.

Der Prototyp bestand die ausformulierten Tests (vgl. Abschnitt 6.2). Im Bereich Performance konnte beispielsweise ein sehr gutes Resultat erreicht werden: Der Prototyp konnte über 90% der theoretisch zur Verfügung stehenden Bandbreite nutzen. Des Weiteren kann die Aufzeichnung der Signalqualität in gewissen Intervallen für Auswertungen und mögliche Optimierungen genutzt werden. In den Testresultaten zeigt sich, dass MPTCP den Verkehr sehr geschickt regelt und, wie in der Analyse angenommen, aufgrund der RTT die Wahl des genutzten Interfaces trifft.

Ein weiterer Erfolg ist, dass durch das Testen ein Fehler in der aktuellen MPTCP-Implementation gefunden werden konnte, der dem Entwicklungsteam mitgeteilt wurde und nun aktiv angegangen und behoben wird. Ebenfalls konnten auf diesem Weg einige Fragen über die Mailingliste von den MPTCP-Experten direkt beantwortet werden.

Das Ziel, ein funktionierendes Konzept auszuarbeiten und zu testen, wurde erreicht. Darüber hinaus wurden auch bereits Ideen für die Erweiterbarkeit dieses Produkts formuliert: Beispielsweise das dynamischen Aktivieren bzw. Deaktivieren von Links basierend auf deren Signalstärke oder das Erweitern der Server-Lösung zu einem Cluster. Auf Basis der Resultate dieser könnte eine funktionierende Produktiv-Installation implementiert werden.

Da die Analyse deutlich mehr Zeit in Anspruch nahm, als ursprünglich geplant, verzögerten sich Design und Implementation. Glücklicherweise konnten die dort aufgetretenen Probleme relativ problemlos und elegant umgangen werden. Daher war das Ziel dieser

Arbeit nie ernsthaft gefährdet. Hierbei zahlten sich die ausführliche Analyse und das Scrum-Vorgehen (beispielsweise durch die parallel stattfindende Evaluation) aus. Mögliche Problemstellen, wie beispielsweise der TCP-over-TCP-Meltdown-Effekt, wurden frühzeitig erkannt und konnten vermieden werden.

8.2 Offene Punkte

Für die Tests blieb am Ende nicht mehr genügend Zeit. Es konnten nur die kritischsten Szenarien getestet werden. Beispielsweise wurde nicht untersucht, wie das System bei einem Wechsel der Funkmasten oder dem vollständigen Verbindungsverlust reagiert.

Diese Arbeit zeigt, dass eine Produktiv-Realisation auf Basis von MPTCP durchaus möglich ist. Die aktuell verfügbare Experimental-Implementation ist allerdings für einen stabilen Betrieb noch zu wenig ausgereift. Die Entwickler von MPTCP bestätigten beispielsweise die Tatsache, dass der aktuelle Release ein Problem mit gewissen WLAN-Treibern hat.

In einem produktiven Betrieb würden die vorgeschlagenen Erweiterungen, wie beispielsweise die Berücksichtigung von Signalqualität oder vollständige Deaktivierung der Mobilfunkverbindungen bei Vorhandensein einer WLAN-Verbindung, aus Kostengründen Sinn machen.

Ein Belastungstest könnte Aufschluss geben, ob ein Loadbalancer vor dem Relay-Server sinnvoll wäre. Damit könnte überprüft werden, wie viele Multifunktions-Wi-Fi-Router bzw. damit verbundene Endgeräte pro Relay-Server bedient werden können.

Schlussendlich muss der Prototyp an die bestehende Architektur sowie die vorhandene Hard- und Software angepasst werden.

Teil II

Anhang

A Abkürzungsverzeichnis

ACK Acknowledgement. 83, 85

ACKs Acknowledgements. 20

AS Autonomous System. 84

CLI Command Line Interface. 35

D-Bus Desktop-Bus. 35

DHCP Dynamic Host Configuration Protocol. 12, 18, 84

DNS Domain Name System. 49, 67

EDGE Enhanced Data Rates for GSM Evolution. 15, 83

GPRS General Packet Radio Service. 18, 83

GSM Global System for Mobile Communications. 83

HSDPA High Speed Downlink Packet Access. 83

HSPA High Speed Packet Access. 83

HSR Hochschule für Technik Rapperswil. 72

HSRP Hot Standby Router Protocol. 25, 26, 109, 124

HSUPA High Speed Uplink Packet Access. 83

ICMP Internet Control Message Protocol. 6, 22, 33

IETF Internet Engineering Task Force. 6, 27

IOS Internetwork Operating System. 21

IP Internet Protocol. 10, 24–26, 30, 32, 50, 56, 117

IP SLA Internet Protocol Service Level Agreement. 21, 26

ISP Internet Service Provider. 85

LACP Link Aggregation Control Protocol. 23

LTE Long Term Evolution. 15, 18, 72, 73, 83

MLPPP Multilink PPP. 23

MPTCP MultiPath TCP. 4, 6–8, 27–34, 38, 47–49, 56, 67, 68, 70, 77, 78, 86–88, 109, 110, 112, 117, 118, 120, 121, 124, 125

MSS Maximal Segment Size. 85

NAPT Network Address and Port Translation. 11, 30, 115, 117

NAT Network Address Translation. 5, 25, 72, 75, 102, 115

OSI Open Systems Interconnection. 3, 22, 32, 128

PAgP Port Aggregation Protocol. 23

PPP Point-to-Point Protocol. 23, 74, 81

RTT Round-Trip Time. 19, 20, 22, 49, 77

RUP Rational Unified Process. 92, 122

RWin TCP Receive Window (Size). 20

SIM Subscriber Identity Module. 73

SOCKS Socket Secure. 6, 31–34, 38, 39, 41, 43, 46, 57, 67, 72, 109, 115, 117–122

SPOF Single Point of Failure. 49

TCP Transmission Control Protocol. 6, 19, 26–30, 33, 34, 38, 41, 43, 47–50, 56, 57, 59, 67, 68, 72, 78, 83, 109, 110, 114, 121

UDP User Datagram Protocol. 6, 29, 33, 56, 67, 72, 119, 121

UMTS Universal Mobile Telecommunications System. 7, 8, 15, 18, 83

USB Universal Serial Bus. 8, 73

VoIP Voice over IP. 58

VPN Virtual Private Network. 5, 6, 13, 18, 29–31, 33, 34, 37, 41, 43, 57, 67, 72, 102, 109, 110, 116–118

WLAN Wireless-LAN. 6, 7, 10, 11, 15, 40, 45, 61, 70, 71, 78, 118

B Glossar

Access-Technology Mobilfunknetz-Antennen unterstützen gewisse Access-Technologies (Mobilfunkstandards). Je nach vorhandenen Technologien wird eine andere verwendet. Bekannte Access-Technologies sind: Global System for Mobile Communications (GSM) (2G), GPRS (2.5G), EDGE (2.75G), UMTS(3G), High Speed Packet Access (HSPA) aufgeteilt in High Speed Downlink Packet Access (HSDPA) und High Speed Uplink Packet Access (HSUPA) (3.5G), LTE (3.9G), LTE Advanced (4G). 51–58, 83

Congestion-Avoidance Congestion-Avoidance ist ein Vorgang, um Staus im Netzwerk zu vermeiden. 19, 83

Congestion-Control Bestandteil von TCP um die Performance zu steigern und Staus zu vermeiden ist die Congestion-Control (durch Congestion-Avoidance). 48, 83, 85

Congestion-Window Das Congestion-Window verhindert, dass eine Middlebox mit Traffic überrannt wird. Das Window beschreibt als variable Grösse, wie viele Packete gesendet werden können, ohne ein Acknowledgement (ACK) zu erhalten. Bei jedem Empfang eines ACK wird das Congestion-Window verdoppelt, bei jedem Packet loss halbiert. Die maximale Grösse entspricht dem Receive-Window des Empfängers. 19, 49, 83, 85, 109

Default-Gateway Der *Default Gateway* oder auch Next-Hop ist derjenige Router, der von einem Gerät verwendet wird, um IPs aus einem anderen Netzwerk zu erreichen. 27, 31, 44–46, 60, 69, 109

Embedded-System Ein eingebettetes System welches in einen technischen Kontext eingebuden ist. Im Kontext dieser Arbeit entspricht es einem Router mit einem eigebetteten Linux. 12

Flow Ein Datenfluss über einen Pfad und bestimmten Link. Ein Flow kann aus mehreren Sub-Flows bestehen, die unterschiedliche Pfade nutzen. 17, 28, 32, 47–49, 84

Jitter engl. “zittern”, “flimmern”. Es beschreibt eine kleine Abweichung der Taktfrequenz bei Datenübertragungen. Netzwerktechnisch gesehen entspricht der Jitter der Varianz der Latenzzeit von Datenpaketen und ist unerwünscht, weil Pakete zu früh oder zu spät eintreffen. 21

Kernel-based Virtual-Machine *KVM* ist eine Vollvirtualisierungslösung für Linux. 5, 103

Latenz Verzögerung. 20, 26, 33, 46

Library Eine Systembibliothek die Funktionen zur Verfügung stellt. 35

Link Eine Verbindung auf Layer 1, die vom Provider zur Verfügung gestellt wird. 3, 12, 14–17, 19, 20, 25, 27, 28, 34, 35, 46, 47, 51–66, 68, 69, 71, 77, 84, 116, 117

Multifunktions-Wi-Fi-Router Ein Gerät, welches neben dem Routen auch noch die Funktion einer Wireless Bridge übernimmt und Server-Dienste (wie zum Beispiel DHCP) zur Verfügung stellt und über Mobilfunk-Modems ans Internet angebunden ist. 4, 11, 12, 17, 18, 40, 49, 73, 74, 78, 109, 110

Next-Hop Router, der das eigene Autonomous System (AS) mit dem nächsten AS auf dem AS-Pfad verbindet. 25, 27, 43, 83, *siehe* Default-Gateway

Packet-Loss Von Packet-Loss oder Paketverlust spricht man, wenn ein Netzwerkpaket nicht innerhalb eines bestimmten Zeitraums am Bestimmungsort eintrifft. 19–22, 44, 46

Peering-Agreement Vertraglich geregeltes Nutzungsverhältnis zwischen zwei Providern, um den Austausch von Datenpakete durchzuführen plural. 40

Pfad Eine Route zwischen zwei Geräten auf Layer 3. 84

Puppet Puppet ist ein Tool zum Konfigurationsmanagement von Servern. Es erlaubt die zentrale Verwaltung der Konfigurationen verschiedener Rechner. 38, 72, 73, 75

Quality-of-Service Beschreibt, ob die Qualität eines Services mit den Anforderungen aus Anwendersicht überein stimmen. 21, 58

Receive-Window Receive-Window (RWin) definiert die Anzahl Bytes, die der Sender schicken darf, ohne ein ACK erhalten zu haben. Die Grösse ist vom Betriebssystem vorgegeben (In Windows in der Registry, bei Linux in `/proc/sys/net/core/`). Der Empfänger teilt dem Sender die Anzahl Bytes mit, die im Receive-Buffer noch frei sind. 83, 85

Relay-Server Zwischenserver der eine Technologie anbietet, die nicht auf allen Servern verfügbar wäre. Die Kommunikation wird nach Empfang an den Zielserver weitergeleitet. 8, 11, 12, 17, 25, 26, 28, 29, 40, 78, 109

Scrum Eine agile Projektvorgehensweise. Der Begriff kommt aus dem Rugby-Sport. Gemeint ist wie beim Sport die Bewegung des gesamten Teams als Einheit. 78, 89, 92, 122

Slow-Start Der Slow-Start-Algogrithmus ist Teil der Congestion-Control. Um Staus zu vermeiden wird mit einer kleinen Sendemenge von einer Maximal Segment Size (MSS) gestartet, weil die Kapazität der involvierten Komponenten nicht bekannt ist. Bei jedem Empfang eines ACK wird das Congestion-Window verdoppelt. 48

SSL/TLS SSL (Secure Sockets Layer) ist die ursprünglichen Bezeichnung für Transport Layer Security (TLS), ein Verschlüsselungsprotokoll für die sichere Übertragung von Daten. 48

Throughput Datendurchsatz. 14–16, 20, 28, 47, 54, 59, 61–66, 68, 70, 71, 85, 110

Traffic Datenverkehr bei Computernetzwerken. 6, 11, 12, 14, 15, 17, 29–31, 33–35, 38, 41, 46, 51–57, 59, 61, 67, 83, 109

Traffic-Overhead Datenmenge, die wegen des verwendeten Protokolls zu den eigentlichen Nutzdaten zusätzlich übertragen werden müssen. 17, 26, 33, 39

Traffic-Shaping Methode um den Datendurchsatz beispielsweise durch Verzögerung in der Übermittlung zu limitieren. Internet Service Providers (ISPs) wenden Traffic-Shaping an, um Angebote mit unterschiedlichem Throughput anbieten zu können. 19, 85

C Persönliche Berichte

C.1 Simon Huber

> Wer immer tut, was er schon kann, bleibt immer das, was er schon ist.
>
> – Henry Ford

Der Beginn einer Arbeit in einem Gebiet, in dem man sich nicht gut auskennt, ist ein Risiko. Es war für mich am Anfang eine grosse Grundsatzfrage, ob das Vorwissen ausreicht. Ganz nach Henry Ford überwog die Neugier, sich einem spannenden Projekt auf neuem Terrain zu widmen. Im Unterschied zu den bisherigen beruflichen Erfahrungen, etwas Bestehendes als Ingenieur anzuwenden, lag der Fokus bei dieser Arbeit in der Wissenschaft und Forschung. MPTCP ist eine vielversprechende Technologie für die Zukunft und wird aus meiner Sicht das Surf-Erlebnis massgebend revolutionieren.

Es war eine neue und sehr eindrückliche Erfahrung, ein Forschungsprojekt so nahe zu verfolgen, mit den Entwicklern über Mailinglisten zu kommunizieren und sich je länger je mehr in ein Thema zu vertiefen. Die anfängliche Schwierigkeit lag vor allem darin, Theorielücken aufzufüllen und sich in die Materie einzuarbeiten. Das Verfassen der Texte erforderte viel Zeit, weil viel recherchiert werden musste.

Obwohl nicht alle Ziele restlos erreicht werden konnten, gelang es schliesslich, einen Prototypen mit einem eigenen Konzept auszuarbeiten, den es vermutlich so heute noch nicht gibt. Vor allem konnten wir durch unsere Tests etwas zum MPTCP-Projekt beitragen. Manche Probleme, wie beispielsweise das langwierige Problem beim Routing, lagen auch daran, dass die Arbeit sehr vertieftes Wissen erfordert hätte.

Um diese Fehler zu analysieren ging sehr viel Zeit verloren. Es war für mich ebenfalls oft eine Ungewissheit, ob es ein Problem vom (noch nicht so stabilen) MPTCP-Projekt verursacht wurde, oder ob die Konfiguration nicht korrekt war.

Am Anfang tat ich mich schwer, die Brücke zwischen Use-Cases, Test-Cases etc. (wie

ich sie aus Softwareprojekten kannte) auf ein Projekt in der Netzwerkinformatik zu schlagen. Nachdem ich die Abstraktion schaffen konnte, dass es im Grunde genau die gleichen Überlegungen wie bei Softwareprojekten sind, war die Verfassung wesentlich einfacher.

Die Arbeit in einem Zweierteam war sehr angenehm, da Entschlüsse schnell fassen konnten und auch die Organisation einfacher ablief. Nils konnte mit seinen technischen Vorkenntnissen und sehr schnellem Eindenken in neue Technologien bei diesem Projekt einige Male wichtige Beiträge leisten. Ich profitierte stark davon, weil wir so schneller zu Lösungen kamen, für die ich sonst Hilfe und Know-how von aussen hätten holen müssen. Herr Stettler gab uns jeweils konstruktive Rückmeldungen und unterstützte uns dadurch sehr.

Letzten Endes nehme ich deshalb sehr viel aus dieser Studienarbeit mit. Ich konnte einerseits viel über die Netzwerkinformatik lernen, kenne nun die Vorgehensweise in einem Netzwerkprojekt und hatte nicht zuletzt auch Spass, MPTCP als Projekt zu verfolgen. Nebenbei lernte ich die Arbeitsweise mit LaTeXund konnte auch viel über *Git* lernen. Viele Netzwerkgrundlagen die man im Studium behandelt hatte, wurden mir durch dieses Projekt handgreiflicher und verständlicher. Auf das Resultat der Arbeit bin ich auch besonders deshalb stolz, weil wir trotz anfänglichen Bedenken mit Fleiss und viel Effort zu einer guten Lösung gekommen sind.

C.2 Nils Caspar

Obwohl – oder gerade weil wir im Bereich "Netzwerk" nur wenig Vorwissen hatten, konnte ich mich schnell für diese Arbeit begeistern. Als die Zuteilung bekannt gegeben wurde, war die Freude gross; hatten wir dieser Arbeit doch die höchste Priorität zugeteilt.

Die Arbeit bot Gelegenheit, einen Einblick in ein äusserst aktives Projekt zu erhalten und auf Basis von experimentellen Entwicklungsergebnissen einen Prototyp zu bauen. Dass das Projekt und die Software noch immer in der Entwicklung ist, wurde uns wiederholt klar gemacht. Bei den letzten Tests begegneten wir so noch einem Problem, welches sich leider nicht mehr innert nützlicher Frist beheben liess: Die aktuelle MPTCP-Implementation hat Schwierigkeiten mit der von uns verwendeten Hardware-Konstellation. Dank dem ausführlichen Bug-Report via Mailingliste wird dieses Problem nun aktiv angegangen und in zukünftigen Versionen wohl behoben sein.

Die auftretenden Herausforderungen mussten jeweils akribisch untersucht und einer Komponente zugeteilt werden. Das erforderte einerseits zwar sehr viel Zeit, war aber

andererseits auch interessant und vermutlich eine der besten Methoden, die verschiedenen Bausteine genauer kennen zu lernen.

Es bereitete mir Freude, mich in dieses für mich neue Gebiet einzuarbeiten und entsprechend meinen Wissensfundus zu erweitern. Auch sehr spannend war für mich der aktive Austausch mit der Community hinter MPTCP. Insbesondere Christoph Paasch, einer der Hauptentwickler hinter dem MPTCP-Projekt, berät via Mailingliste jeweils geduldig und ausführlich. Dank seiner Hilfe gelang es uns, den erwähnten Bug genauer zu lokalisieren und das zugrunde liegende Problem besser zu verstehen.

Das Arbeiten im Team, wie auch mit dem Betreuer, war sehr angenehm, harmonisch und lösungsorientiert. Entsprechend fiel es mir auch leicht, auf Kosten von anderen Modulen und nicht zuletzt auch meiner Freizeit, an diesem Projekt zu arbeiten.

Das Ergebnis überzeugt und erfüllt mich mit Stolz. Ich konnte sehr viel Neues lernen und mein bestehendes Wissen vertiefen.

D Projektmanagement

D.1 Teamstruktur, Rollen und Verantwortlichkeiten

Das Team bestand aus Nils Caspar und Simon Huber. Die Rollenverteilung nach Scrum wurde nicht streng eingehalten, da das Team zu klein war um alle Rollen zu besetzen. Während Nils Caspar für die Implementation verantwortlich war, lag die Verantwortung von Simon Huber beim Projektmanagement und der Dokumentation. Die Analyse und das Design wurde aufgeteilt.

Die Vorgehensweise war, alle Texte jeweils vom Teampartner gegenlesen zu lassen, bevor sie in das Dokument bzw. den `master`-*Git*-Branch aufgenommen wurden. Dadurch konnten Fehler vermieden werden, die Projektmitarbeiter wurden über die Tätigkeiten und Fortschritte des jeweilig anderen unterrichtet und es gab einen Austausch des Erlernten.

D.2 Auswertung Zeitreporting

D.2.1 Zeitauswertung pro Person und Woche

Mitglied	2013-38	2013-39	2013-40	2013-41	2013-42
Nils Caspar	8.50	10.00	20.00	14.50	20.50
Simon Huber	10.25	9.00	19.00	14.00	20.75
Gesamtzahl	**18.75**	**19.00**	**39.00**	**28.50**	**41.25**

Tabelle D.1: Zeitauswertung - Arbeitswoche 1-5

Mitglied	2013-43	2013-44	2013-45	2013-46	2013-47
Nils Caspar	14.25	10.00	19.50	14.00	13.50
Simon Huber	12.00	8.75	20.75	14.00	14.25
Gesamtzahl	**26.25**	**18.75**	**40.25**	**28.00**	**27.75**

Tabelle D.2: Zeitauswertung - Arbeitswoche 6-10

Mitglied	2013-48	2013-49	2013-50	2013-51	**Total**
Nils Caspar	11.00	29.40	48.50	13.00	**246.65**
Simon Huber	11.50	26.75	51.75	11.50	**244.25**
Gesamtzahl	**22.50**	**56.15**	**100.25**	**24.50**	**490.90**

Tabelle D.3: Zeitauswertung - Arbeitswoche 11-14 und Summe

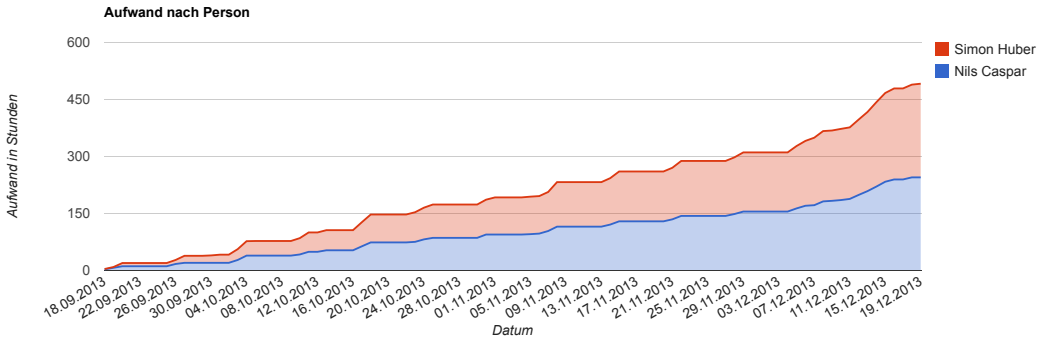

Abbildung D.1: Prozentualer Aufwand pro Aktivität

Gemäss Aufschlüsselung der Aufwände nach Personen (vgl. Abbildung D.1) ist ersichtlich, wie der Verlauf des Gesamtaufwands entstanden ist. Da beide Beteiligten jeweils am Wochenanfang einer beruflichen Beschäftigung nachgingen, ist ein stufenweiser Anstieg sichtbar.

D.2.2 Zeitauswertung pro Aktivität

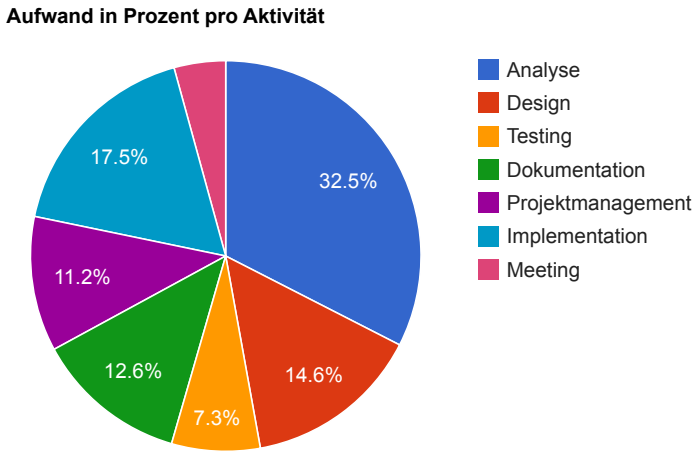

Abbildung D.2: Prozentualer Aufwand pro Aktivität

In Abbildung D.2 ist der Gesamt-Aufwand auf die verschiedenen Aktivitäten aufgeteilt. Die meiste Zeit floss in die Analyse. Für Design und Implementation wurde in etwa gleich viel Zeit aufgewendet.

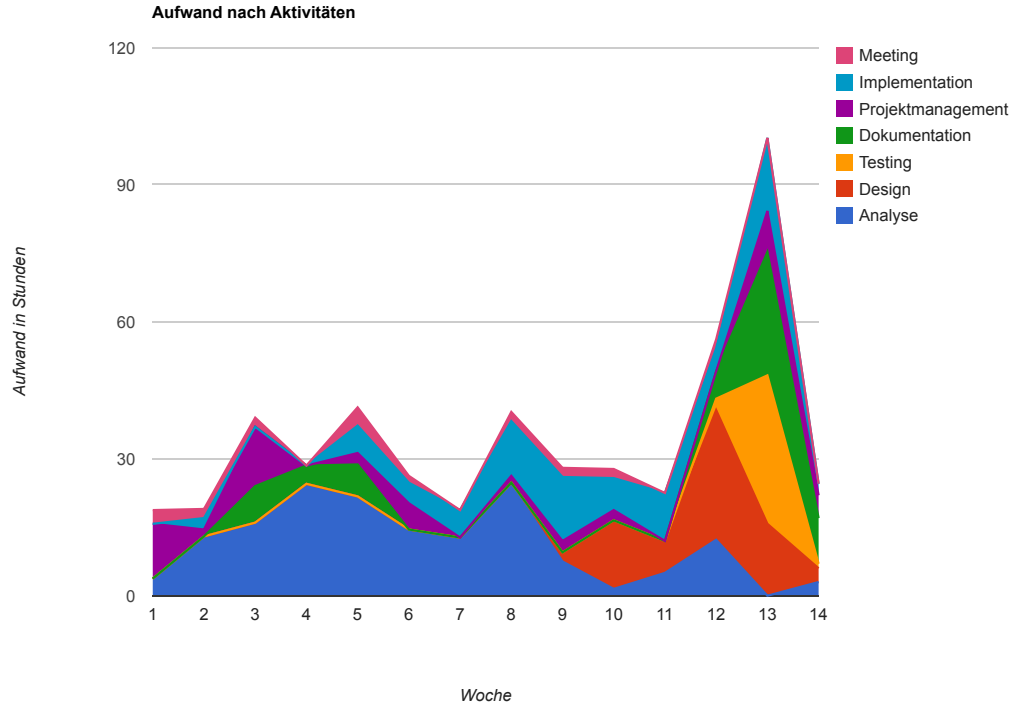

Abbildung D.3: Prozentualer Aufwand pro Aktivität

Die Abbildung D.3 zeigt, dass am Anfang die meiste Zeit für Analysen verwendet wurde. In den Wochen 8 bis 11 stand die Implementation im Vordergrund. Da in Woche 11 einige Verzögerungen zu einem Rückstand führten, wurden zwei zusätzliche Wochenenden investiert, was den Peak erklärt.

D.3 Planungsphasen

In der Planungsphase wurde entschieden nach Scrum vorzugehen. Es wurde trotzdem eine Inception- sowie eine Elaboration-Phase gemäss Rational Unified Process (RUP) vorgeschoben, um die Ziele und Grenzen der Arbeit zu definieren sowie Analysen zu betreiben.

Im Folgenden werden die Planungsphasen und die bearbeiteten Tickets aufgeführt.

D.3.1 Inception

Nr	Beschreibung	Aufw. (h)	Verantwortlich
6	Kickoff Meeting	2.00	Simon Huber
3	Grober Projektplan	7.00	Simon Huber
4	Problemstellung dokumentieren	5.00	Simon Huber
1	Redmine aufsetzen	5.00	Nils Caspar

D.3.2 Elaboration

Nr	Beschreibung	Aufw. (h)	Verantwortlich
2	Sitzung #3	2.00	Simon Huber
5	Inbetriebnahme USB Dongles	6.75	Nils Caspar
2	Einrichtung Ubuntu auf Test-Laptop	2.00	Nils Caspar
8	Sitzung #2	2.00	Simon Huber
9	Beschaffung Hardware	1.50	Simon Huber
15	Protokoll Sitzung #3	0.50	Simon Huber
7	Technologieanalyse	21.00	Nils Caspar
17	Protokoll Sitzung #2	2.00	Simon Huber

D.3.3 Sprint 1

Nr	Beschreibung	Aufw. (h)	Verantwortlich
18	Messmethoden beschreiben	15.50	Simon Huber
10	Studium Multipathing Strategien	4.00	Simon Huber
14	Protokollvorlage	2.00	Simon Huber
29	Erweiterung Nicht-Funktionale Anforderungen	2.50	Simon Huber
31	OpenVPN-Tunnel einrichten	4.50	Nils Caspar
11	Studium Scrum	1.50	Nils Caspar
23	MPTCP-Kernel auf Server+Client installieren und konfigurieren	3.00	Nils Caspar
21	Sitzung #5	2.50	Simon Huber
13	Projektanalyse	6.00	Simon Huber
25	Multipath Routing beschreiben	3.00	Nils Caspar
16	Sitzung #4	4.00	Simon Huber
34	Fazit für die unterschiedlichen Varianten	2.00	Nils Caspar
24	Analyse Korrekturen	5.00	Simon Huber
20	User Stories + Szenarien	7.50	Simon Huber
26	Anforderungs-Spezifikation gemäss Besprechung	7.25	Simon Huber
22	Restrukturierung gemäss Sitzung	7.00	Simon Huber

D.3.4 Sprint 2

Nr	Beschreibung	Aufw. (h)	Verantwortlich
40	Meeting vom 05.11.2013	3.50	Simon Huber
39	Use-Cases überarbeiten	8.75	Simon Huber
37	Systemgrenzen-Diagramm überarbeiten	1.00	Nils Caspar
48	Requirements: UCs gemäss Besprechung anpassen	4.25	Nils Caspar
45	Dokumentation SOCKS (Umgang mit UDP, ...)	4.00	Nils Caspar
41	OpenVPN-Stack grafisch darstellen	2.50	Nils Caspar
49	Anpassung Analyse VPN	4.00	Simon Huber
32	Alternative zu VPN evaluieren	24.50	Nils Caspar
44	SOCKS Stack darstellen	2.00	Simon Huber
46	Analyse Linux Verbinungs-Erkennung	6.00	Nils Caspar

D.3.5 Sprint 3

Nr	Beschreibung	Aufw. (h)	Verantwortlich
57	Sitzung vom 21.11.2013	2.75	Nils Caspar
19	MPTCP beschreiben	4.50	Nils Caspar
51	Testsetup mit VM und Router	20.00	Nils Caspar
56	Recherche USB-Adapter Driver	5.00	Simon Huber
30	Systemkomponenten beschreiben	3.50	Nils Caspar
33	Grafik mit Systemgrenze	1.50	Nils Caspar
36	Sitzung #5	4.75	
52	Sitzung vom 14.11.2013	4.50	
55	Dokumentation Layer 4 gemäss Meeting vom 14.11.2013 überarbeiten	0.50	Nils Caspar
50	VM aufsetzen	3.00	Nils Caspar
54	Testcases erfassen	2.00	Simon Huber
59	Grafik: iptables Flow	5.00	Nils Caspar
53	Anpassungen Use-Cases und Systemgrenzen	7.50	Nils Caspar

D.3.6 Sprint 4

Nr	Beschreibung	Aufw. (h)	Verantwortlich
80	Daten-Optionen für Mobilabos	1.00	Simon Huber
78	Beschreibung Netwerktopologie	6.40	Nils Caspar
69	MPTCP: strategy compared to tcp	10.75	Simon Huber
60	Dokumentation ModemManager	6.50	Simon Huber
64	MPTCP: Doku Entscheidung Datenverteilung	5.75	Simon Huber
75	Sitzung vom 06.12.2013	4.50	Simon Huber
70	Routing/iptables-Ablauf dokumentieren	4.00	Nils Caspar
63	MPTCP: Default Gateway Problem beschreiben	2.50	Nils Caspar
73	Umstrukturierung Kapitel	2.00	Nils Caspar
61	Architekturentscheide dokumentieren	7.00	Nils Caspar
76	Meeting vom 29.11.2013 (Technopark)	6.00	Simon Huber
77	Default Gateway Wahl-Algorithmus	10.25	Nils Caspar
58	Textanpassungen UCs und TCs gemäss Sitzung am 21.11.2013	8.00	Simon Huber

D.3.7 Sprint 5

Nr	Beschreibung	Aufw. (h)	Verantwortlich
83	Daeamon-Script: Auslesen Adapterdaten und backup on/off	0.00	Nils Caspar
74	Abschirmung Link beschreiben	1.00	Simon Huber
66	MASQUERADE durch SNAT ersetzen	0.00	Nils Caspar
91	Tests wiederholen	1.00	Nils Caspar
82	Testing vorbereiten	3.75	Simon Huber
65	MPTCP Tuning Dis-/Advantages	3.00	Simon Huber
86	Poster	13.00	Simon Huber
85	Kurzfassung	13.75	Nils Caspar
88	Erste Tests durchführen und auswerten	18.25	Nils Caspar
87	Link automatisch aktivieren	2.00	Nils Caspar
68	Middlebox	3.00	Simon Huber
62	Parameter mit libnm-glib auslesen	4.00	Nils Caspar
81	Implementation-Setup dokumentieren	12.00	Nils Caspar
67	RP-Filter dokumentieren	0.00	Simon Huber
93	MPTCP-Bug beschreiben	0.00	Nils Caspar
84	Testcases erweitern und Korrektur gemäss 6.12.13	7.00	Simon Huber
38	OpenVPN-Tunnel dokumentieren	3.00	Nils Caspar
89	Testcase: WLAN-Bevorzugung	7.00	Simon Huber
42	TCP-over-TCP-Meltdown beschreiben	0.00	Nils Caspar
71	Abbildungsverzeichnis/Literaturverzeichnis, Quellen, ...	10.75	Simon Huber
94	Product-Name statt emph	1.00	Nils Caspar
96	Protokolle extrahieren	0.00	Nils Caspar
79	Abstimmung Analyse/Design	16.75	Simon Huber

D.4 Meilensteine

D.4.1 Meilenstein #1: Problemdomäne definiert

Als Produkt der Elaboration soll die Problemdomäne definiert sein.

D.4.2 Milestone #2: Abschluss Analyse

Als Produkt des dritten Sprints soll die Analyse ganz abgeschlossen sein.

D.4.3 Milestone #3: Working Prototype

Als Produkt des vierten Sprints soll der Prototyp lauffähig sein.

D.4.4 Milestone #4: Arbeit + Prototyp abgabebereit

Als Produkt des letzten Sprints soll die Arbeit fertiggestellt werden, damit die Dokumentation sauber und der Prototyp abgabebereit ist.

D.5 Projektplan

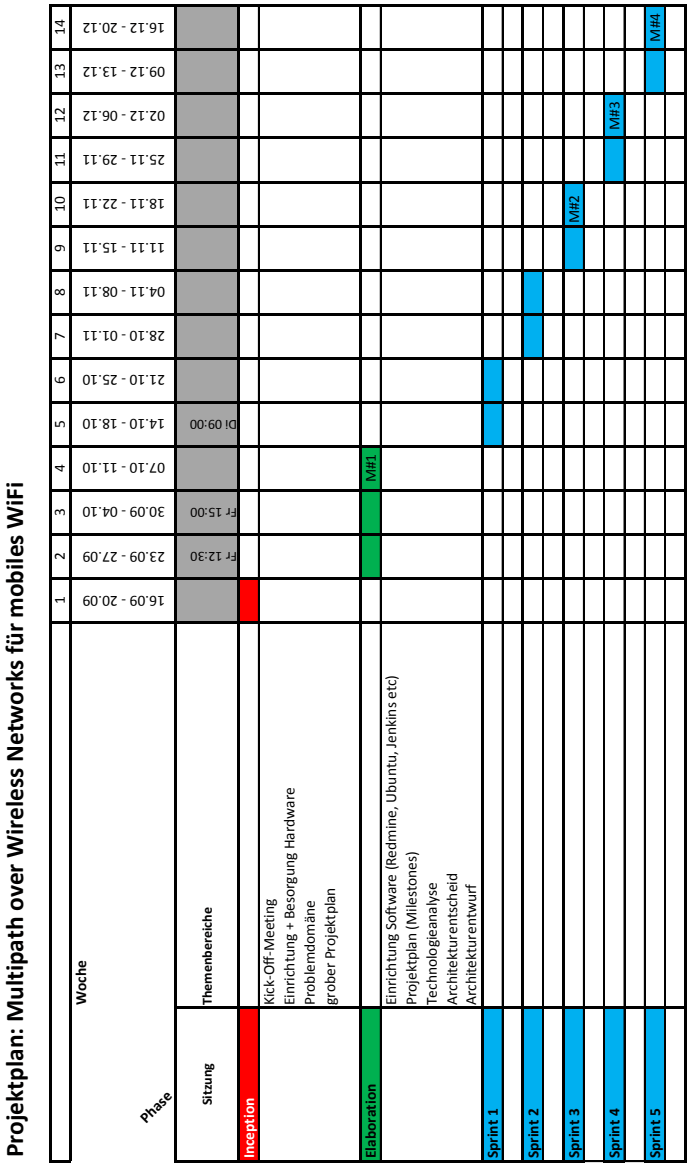

Abbildung D.4: Terminplan

E Konfigurationen

E.1 Dante

```
1 logoutput: syslog
2
3 internal: eth0 port = 11371
4 external: eth0
5
6 method: username none
7 user.notprivileged: nobody
8
9 client pass {
10   from: 0.0.0.0/0 to: 0.0.0.0/0
11   log: connect disconnect error
12 }
13 block {
14   from: 0.0.0.0/0 to: 127.0.0.0/8
15   log: connect disconnect error
16 }
17 pass {
18   from: 0.0.0.0/0 to: 0.0.0.0/0
19   protocol: tcp
20   command: bind connect
21   log: connect disconnect error
22 }
```

Listing E.1: `/etc/danted.conf`

E.2 OpenVPN

E.2.1 Puppet

```
openvpn::server { 'openvpn.example.com':
  country      => 'CH',
  province     => 'ZH',
  city         => 'Zurich',
  organization => 'example.com',
  email        => 'info@example.com',
  server       => '10.10.10.0 255.255.255.0',
  proto        => 'tcp',
  port         => 5228,
}

openvpn::client { 'testgeraet.openvpn.example.com':
  server => 'openvpn.example.com'
}
```

E.2.2 Generierte Server-Konfiguration

```
mode server
client-config-dir
    /etc/openvpn/openvpn.example.com/client-configs
ca /etc/openvpn/openvpn.example.com/keys/ca.crt
cert /etc/openvpn/openvpn.example.com/keys/server.crt
key /etc/openvpn/openvpn.example.com/keys/server.key
dh /etc/openvpn/openvpn.example.com/keys/dh1024.pem
proto tcp-server
port 5228
tls-server
comp-lzo
group nogroup
user nobody
status openvpn.example.com/openvpn-status.log
dev tun0
```

```
15 server 10.10.10.0 255.255.255.0
```

Listing E.2: /etc/openvpn/openvpn.example.com.conf

E.2.3 Generierte Client-Konfiguration

```
1 client
2 ca keys/ca.crt
3 cert keys/testgeraet.openvpn.example.com.crt
4 key keys/testgeraet.openvpn.example.com.key
5 dev tun
6 proto tcp
7 remote openvpn.example.com 5228
8 comp-lzo
9 resolv-retry infinite
10 nobind
11 persist-key
12 persist-tun
13 mute-replay-warnings
14 ns-cert-type server
15 verb 3
16 mute 20
```

Listing E.3: /etc/openvpn/testgeraet.openvpn.example.com.conf

E.3 VPN NAT

E.3.1 Puppet

```
1 firewall { '102 NAT for VPN':
2   chain    => 'POSTROUTING',
3   jump     => 'MASQUERADE',
4   proto    => 'all',
5   outiface => 'eth0',
6   source   => '10.10.10.0/24',
7   table    => 'nat',
```

```
8 }
```

E.3.2 Generierte Konfiguration

```
1 # Generated by iptables-save v1.4.12 on Sat Dec 14 11:25:44
    2013
2 *nat
3 :PREROUTING ACCEPT [0:0]
4 :INPUT ACCEPT [0:0]
5 :OUTPUT ACCEPT [0:0]
6 :POSTROUTING ACCEPT [0:0]
7 -A POSTROUTING -s 10.10.10.0/24 -o eth0 -m comment
    --comment "102 NAT for VPN" -j MASQUERADE
8 COMMIT
9 # Completed on Sat Dec 14 11:25:44 2013
```

E.4 Kernel-based Virtual-Machine

E.4.1 Netzwerkkonfiguration

```
1 <network>
2   <name>default</name>
3   <uuid>9b46b552-7c53-2404-d8c5-4fa69e3905d0</uuid>
4   <forward mode='nat'/>
5   <bridge name='virbr0' stp='on' delay='0' />
6   <mac address='52:54:00:07:A3:AD'/>
7   <ip address='192.168.122.1' netmask='255.255.255.0'>
8     <dhcp>
9       <range start='192.168.122.2' end='192.168.122.254' />
10    </dhcp>
11  </ip>
12 </network>
```

E.5 Modems

E.5.1 Swisscom

```
1 -detach
2 /dev/ttyUSB2
3 9600
4 defaultroute
5 usepeerdns
6 mtu 1492
7
8 noauth
9
10 crtscts
11 lock
12 # novj
13 # nobsdcomp
14 nodeflate
15 # nopcomp
16
17 connect '/usr/sbin/chat -v -t6 -f /etc/ppp/chats/swisscom'
```

Listing E.4: Peer-Konfiguration Swisscom (`/etc/ppp/peers/swisscom`)

```
1 ABORT BUSY
2 ABORT 'NO CARRIER'
3 ABORT ERROR
4 REPORT CONNECT
5 TIMEOUT 120
6 "" "AT&F"
7 OK "ATZ"
8 OK "ATQ0 V1 E1 S0=0 &C1 &D2"
9 # enter pin below if needed
10 #OK AT+CPIN? READY-AT+CPIN=1234-OK ""
11 OK 'AT+CGDCONT=1,"IP","gprs.swisscom.ch"'
12 SAY "Calling Swisscom"
13 OK "ATDT*99***1#"
```

```
14 TIMEOUT 120
15 CONNECT ''
```

Listing E.5: Chat-Script Swisscom (`/etc/ppp/chats/swisscom`)

E.5.2 Orange

```
1 -detach
2 /dev/ttyUSB5
3 9600
4 defaultroute
5 usepeerdns
6 mtu 1492
7
8 noauth
9
10 crtscts
11 lock
12 # novj
13 # nobsdcomp
14 nodeflate
15 # nopcomp
16
17 connect '/usr/sbin/chat -v -t6 -f /etc/ppp/chats/orange'
```

Listing E.6: Peer-Konfiguration Orange (`/etc/ppp/peers/orange`)

```
1 ABORT BUSY
2 ABORT 'NO CARRIER'
3 ABORT ERROR
4 REPORT CONNECT
5 TIMEOUT 120
6 "" "AT&F"
7 OK "ATZ"
8 OK "ATQ0 V1 E1 S0=0 &C1 &D2"
9 # enter pin below if needed
10 #OK AT+CPIN? READY-AT+CPIN=1234-OK ""
```

```
11 OK 'AT+CGDCONT=1,"IP","click"'
12 SAY "Calling Orange"
13 OK "ATDT*99***1#"
14 TIMEOUT 120
15 CONNECT ''
```

Listing E.7: Chat-Script Orange (`/etc/ppp/chats/orange`)

E.6 Routing-Flags

```
1 net.ipv4.conf.default.rp_filter=2
2 net.ipv4.conf.all.rp_filter=2
3
4 net.ipv4.ip_forward=1
```

Listing E.8: `/etc/sysctl.d/60-mptcp.conf`

Literaturverzeichnis

[1] Beijnum, I. van: *Multipath TCP lets Siri seamlessly switch between Wi-Fi and 3G/LTE.* Website, 2013. – Online verfügbar `http://arstechnica.com/apple/2013/09/multipath-tcp-lets-siri-seamlessly-switch-between-wi-fi-and-3glte/`; abgerufen am 25. September 2013

[2] Dawkins, S. : *End-to-end Performance Implications of Links with Errors.* Website, 2001. – Online verfügbar `http://tools.ietf.org/html/rfc3155`; abgerufen am 15. Dezember 2013

[3] Detal, G. : *Re: [mptcp-dev] Default Gateway selection.* Website, 2012. – Online verfügbar `https://listes-2.sipr.ucl.ac.be/sympa/arc/mptcp-dev/2013-12/msg00012.html`; abgerufen am 06. Dezember 2013

[4] Ford, A. : *TCP Extensions for Multipath Operation with Multiple Addresses: 2.5. Requesting a Change in a Path's Priority.* Website, 2012. – Online verfügbar `http://tools.ietf.org/html/rfc6824#section-2.5`; abgerufen am 18. Oktober 2013

[5] Ford, A. : *TCP Extensions for Multipath Operation with Multiple Addresses.* Website, 2013. – Online verfügbar `http://tools.ietf.org/html/rfc6824`; abgerufen am 18. Oktober 2013

[6] Honda, O. ; Ohsaki, H. ; Imase, M. ; Ishizuka, M. ; Murayama, J. : *Understanding TCP over TCP: Effects of TCP Tunneling on End-to-End Throughput and Latency.* Paper, 2007. – Online verfügbar `http://www.ispl.jp/~oosaki/papers/Honda05_ITCom.pdf`; abgerufen am 15. Dezember 2013

[7] Jacobson, V. ; Braden, R. ; Borman, D. : *TCP Extensions for High Performance.* Website, 1992. – Online verfügbar `http://tools.ietf.org/html/rfc1323#section-4l`; abgerufen am 15. Dezember 2013

[8] Khalili, R. : *Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP.* Website, 2013. – Online verfügbar `http://tools.ietf.org/html/draft-khalili-mptcp-congestion-control-02`; abgerufen am 18. Dezember 2013

[9] Maruseac, A. : *Multipath TCP Patch to packet-tcp.c.* Website, 2012. – Online verfügbar `https://bugs.wireshark.org/bugzilla/show_bug.cgi?id=6705`; abgerufen am 18. Oktober 2013

[10] Paasch, C. : *Securing the MultiPath TCP handshake with external keys.* Website, 2012. – Online verfügbar `http://tools.ietf.org/html/draft-paasch-mptcp-ssl-00`; abgerufen am 18. Oktober 2013

[11] *Kapitel* 14. In: Rubini, A. ; Jonathan, C. : *Linux Device Drivers, 2nd Edition.* O'Reilly Verlag GmbH, 2002. – Online verfügbar `http://www.oreilly.de/german/freebooks/linuxdrive2ger/book1.html`; abgerufen am 15. Dezember 2013

[12] samuli: *Improve TCP-over-TCP performance.* Website, 2010. – Online verfügbar `http://community.openvpn.net/openvpn/ticket/2`; abgerufen am 07. November 2013

[13] Young, J. ; Wing, D. : *MPTCP and Product Support Overview, Impact on Flow Inspection.* Website, 2013. – Online verfügbar `http://www.cisco.com/en/US/tech/tk364/technologies_tech_note09186a0080c18572.shtml`; abgerufen am 18. Oktober 2013

Abbildungsverzeichnis