HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ins institute for networked solutions

Multipath over Wireless Networks für mobiles WiFi

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2014

Autoren:	Jannis Grimm, Thomas Zeender
Betreuer:	Prof. Beat Stettler
Projektpartner:	CloudGuard Software AG, Pfäffikon SZ
Experte:	Michael Schneider
Gegenleser:	Prof. Dr. Andreas Rinkel

Abstract

Ausgangslage: Die CloudGuard Software AG bietet eine Mobile-Internet-Lösung für Betriebe des öffentlichen Verkehrs. Für dieses mittels Routing und VPN realisierte Produkt soll der Einsatz des Protokolls MPTCP (MultiPath TCP) einer näheren Betrachtung unterzogen werden. MPTCP erlaubt es, die Bandbreite mehrerer Interfaces für eine übergeordnete TCP-Verbindung zu kombinieren. Trotz schwankendem Empfang verschiedener Mobilfunkprovider könnte so stets von kombinierten Durchsatzgeschwindigkeiten profitiert werden. Da heutige Clients und Server meist noch nicht MPTCP-fähig sind, ist eine Lösung nötig, mit der die Pakete im Fahrzeug um MPTCP erweitert und ausserhalb des Fahrzeuges wieder zu normalen TCP-Verbindungen umgewandelt werden.

Vorgehen/Technologie: MPTCP wird im, momentan experimentellen, RFC 6842 beschrieben und eine Implementation von der Universität Université catholique de Louvain als Open Source veröffentlicht. Die Umgebung vom Fahrgast, die Fahrzeug-Hardware und eine Gegenstelle im CloudGuard-Rechenzentrum wurden virtuell nachgebaut und Fahrzeug-Hardware sowie Gegenstelle mit einer Installation von MPTCP versehen. Um den Datenverkehr zwischen Fahrzeug und Gegenstelle über MPTCP übertragen zu können, wird der Nicht-TCP-Verkehr über ein TCP-VPN geleitet. So kann sämtlicher Verkehr über TCP abgewickelt und um MPTCP erweitert werden.

Ergebnis: Durch MPTCP konnte in dieser Umgebung die Bandbreite je Fahrzeug erhöht werden und bleibt auch beim Wegschalten einzelner Interfaces stabil. Auf der Prototyp-Umgebung konnte die Funktionsweise erarbeitet und erfolgreich erprobt werden. Dokumentation und Installationsanleitung stellen sicher, dass auf den Ergebnissen der Arbeit aufgebaut werden kann.

Management Summary

Ausgangslage

Der Wunsch, unterwegs aufs Internet zugreifen zu können, ist allgegenwärtig. Um diesem Bedürfnis nachzukommen, bieten immer mehr Betreiber von Infrastrukturen Gratis-Internet über WiFi an. Die Firma CloudGuard hat eine Lösung für solche Betreiber entwickelt. Für die Redundanz wird der Datenfluss über zwei unabhängige Mobilanbieter verteilt.

Im Rahmen dieser Arbeit wurde dem Wunsch der CloudGuard nachgegangen, diese zwei Mobile-Anschlüsse mittels MPTCP zu kombinieren.

Vorgehen / Technologie

MPTCP erlaubt es, den Datenverkehr von verschiedenen Anschlüssen zum selben Ziel virtuell zu kombinieren. Fällt ein Anschluss weg, bleibt die übergeordnete Verbindung bestehen. Die Übertragungsrate für eine übergeordnete Verbindung wird erhöht und es wird gleichzeitig die Redundanz gesteigert. Um den Einsatz von MPTCP zu ermöglichen, wird eine Daten-Brücke zwischen dem MPTCP-fähigem Bus und einer MPTCP-fähiger Gegenstelle im Internet errichtet.

Ergebnisse

Das entwickelte System bildet die Fahrzeug-Umgebung virtuell nach. Es zeigt die Datenübertragung vom Fahrgast ins Internet auf und erlaubt das entwickeln der Software-Komponenten. Beim abschalten oder blockieren von Sub-Verbindungen bleibt die übergeordnete Verbindung bestehen. Dokumentation und Installationsanleitung stellen sicher, dass auf den Ergebnissen der Arbeit aufgebaut werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

ACK Acknowledgement-Paket in TCP[1]

CAN Controller Area Network

CARP Common Address Redundancy Protocol

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

ECTS European Credit Transfer and Accumulation System

GPRS General Packet Radio Service

HMAC Keyed-Hash Message Authentication Code

HSR Hochschule für Technik Rapperswil

HSRP Host Standby Router Protocol

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

INS Institute for Networked Solutions

[1] Transmission Control Protocol

LTE Long Term Evolution

MPTCP Multipath-TCP

NAT Network Address Translation

OSI Open Systems Interconnection

OSPF Open Shortest Path First

ÖV Öffentlicher Verkehr

RFC Request for Comments

RTT Round Trip Time

RJ45 Registered Jack 45 (Stecker für Ethernet)

SOCKS Sockets (SOCKetS)

SSH Secure Shell

SYN Synchronize-Paket in TCP

TCP Transmission Control Protocol

UCL Université catholique de Louvain

UDP User Datagram Protocol

VPN Virtual Private Network

VM Virtuelle Maschine

VRRP Virtual Router Redundancy Protocol

1. Einleitung und Übersicht

Im Rahmen der Bachelorarbeit „Multipath over Wireless Networks für mobiles WiFi“ wurde das Produkt *MPTCP-WiFi* entwickelt.

Die Arbeit stellt eine Fortsetzung der Studienarbeit mit selbem Titel von Nils Caspar und Simon Huber dar. Da der in der Studienarbeit erstellte Prototyp die Komponenten statisch statt automatisch konfiguriert hatte und daher nicht entsprechend der Aufgabenstellung funktionierte, mussten grosse Teile der Studienarbeit jedoch auch hinterfragt und neu erstellt werden.

Kunde und Auftraggeber der Arbeit ist das Unternehmen CloudGuard Software AG. Dieses Unternehmen erstellt unter anderem für Fahrzeugunternehmen Lösungen, die einen Internet-Zugriff über Mobilfunk innerhalb der Fahrzeuge bereitstellt. Dazu wird ein Multifunktions-Router im Fahrzeug installiert. Dieses Gerät erlaubt einen Internet-Zugriff über WiFi und leitet die Daten über mehrere Mobilfunk-Provider sowie, beispielsweise an grösseren Bahnhöfen, über WiFi an das Internet.

Heutiger Stand dabei ist es, dass zwischen mehreren Mobilfunkanbietern mittels Routing gewechselt wird. Es können nicht mehrere Leitungen gleichzeitig für eine Verbindung genutzt werden. Hat dieser Provider keinen Empfang mehr, wird sie über den anderen Provider umgeleitet. Zudem ist es nicht praktikabel messbar, welchen Durchsatz welche Leitung zu einem bestimmten Zeitpunkt erreichen kann, womit eine intelligente Aufteilung auf verschiedene Leitungen erschwert ist.

Im Rahmen dieser Bachelorarbeit und aufbauend auf die Vor-Studienarbeit soll eine neue Lösung basierend auf Multipath-TCP erstellt werden. Multipath-TCP ist ein Standard, der auf TCP aufbaut und dieses um die Fähigkeit erweitert, mehrere einzelne TCP-Leitungen zu einer logischen TCP-Verbindung zu bündeln. So können mehrere Mobilfunk-Provider gemeinsam für den gesamten Datenverkehr verwendet werden. Die Daten werden

auf alle verfügbaren Leitungen aufgeteilt. Fällt der Empfang zu einem Provider aus, kann so jede Verbindung über die anderen Provider fortgeführt werden. Zudem bietet Multipath-TCP auch für grössere Datenmengen Vorteile über Load Balancing.

Um Multipath-TCP einsetzen zu können, muss jeder Client und jeder Server diese TCP-Erweiterung implementiert haben. Nach aktuellem Stand gibt es jedoch kaum Server sowie keine verbreiteten Client-Systeme, die Multipath-TCP integriert haben. Zudem kann Multipath-TCP nicht für andere Protokolle als IP/TCP, beispielsweise UDP, eingesetzt werden. Für diese Probleme galt es in der Arbeit Lösungen zu finden. Dabei konnte grösstenteils auf Ideen aus der Vor-Studienarbeit aufgesetzt werden, jedoch mussten diese neu implementiert werden.

Auf Kapitel 1, diese Einleitung, folgt die Anforderungsspezifikation in Kapitel 2. Sie geht auf die Ziele ein, die diese Arbeit verfolgt. Der Methodik, mit welcher die Ziele erreicht wurden, widmet sich der Projektplan in Kapitel 3.

Es folgt in Kapitel 4 die anschliessend durchgeführte Analyse der Gegebenheiten. Die Vor-Studienarbeit sowie die wichtigsten Komponenten wurden untersucht, die hieraus gewonnenen Erkenntnisse dokumentiert und betrachtet, was dies für das in dieser Arbeit beschriebene Projekt bedeutet. Aus der Analyse der Gegebenheiten entstand die Domainanalyse, die in Kapitel 5 zu finden ist. Sie geht auf den geplanten Aufbau des in diesem Projekt entwickelten Produks *MPTCP-WiFi* ein und beschreibt dessen Umwelt und innere Funktionsweise.

Das anschliessend entwickelte Produkt *MPTCP-WiFi* ist in Kapitel 6 dokumentiert. Die Komponenten werden beschrieben und erklärt. Kapitel 7 beleuchtet die Installation der Komponenten. Die Anleitung, wie die Komponenten installiert und in Betrieb genommen werden, ist um Erklärungen ergänzt. Das Ziel dieser beiden Kapitel ist es, das Produkt *MPTCP-WiFi* nachvollziehbar und nachbaubar zu gestalten, damit auf dieser Arbeit aufgebaut und das Produkt weiterentwickelt werden kann. Damit wird der Wunsch verfolgt, das Produkt eines Tages produktiv einsetzen zu können.

Das Dokument schliesst mit den Schlussfolgerungen in Kapitel 8 mit einer kritischen Betrachtung der erreichten Ziele sowie einem Ausblick auf mögliche Erweiterungen.

2. Anforderungsspezifikation

2.1. Grossräumige Zielsetzung, Motivation

In modernen öffentlichen Verkehrsmitteln (z. B. Postautos) wird eine Internetverbindung via WiFi angeboten. Diese steht je nach Betreibergesellschafft nur den eigenen technischen Geräten oder auch den Kunden zur Verfügung.

Üblicherweise wird der Internetverkehr dabei über Aussenantennen an einen Mobilfunkbetreiber weitergeleitet. Die Nutzleistung wird jedoch durch Funklöcker und Kapazitätsengpässe eingeschränkt. Es wäre deshalb wünschenswert, die Netze verschiedener Mobilfunkbetreiber gleichzeitig nutzen zu können und den Datenverkehr dynamisch auf diese aufzuteilen.

2.2. Produktfunktion

Das Produkt *MPTCP-WiFi* soll ermöglichen, den Datenverkehr der Nutzer mithilfe von MPTCP[1] über mehrere Mobile-Provider gleichzeitig aufzuteilen. Der Benutzer soll durch allfällige Mobilfunknetz- und IP-Adress-Wechseln nicht beeinträchtigt werden, sondern von höherer Bandbreite profitieren können.

2.3. Benutzercharakteristik

Die Nutzer des Produkts *MPTCP-WiFi* bestehen aus zwei Gruppen:

[1] Multipath-TCP

Natürliche Nutzer: Die Zielgruppe, bestehend aus Menschen, lässt sich über ihren gemeinsamen Wunsch auf mobiles Internet definieren. Ihre jeweiligen Nutzerprofile unterscheiden sich durch die verwendeten Technologien. Durch diese haben sie unterschiedliche Ansprüche.

Technische Nutzer: Technische Geräte wie Kassensysteme greifen ebenfalls auf das Internet zu. Diese haben jedoch konkrete Anforderungen durch fest bestimmte Ressourcen, die sie im Internet verwenden.

2.4. Haupt-Use-Case

Aus der Sicht der Anwender lässt sich ein Use Case definieren.

2.4.1. Haupt-Use-Case-Diagramm

Das Use-Case-Diagramm in Abbildung 2.1 auf Seite 18 stellt den Haupt-Use-Case dar. Dieser beschreibt das oberste Ziel der Nutzer bei der Interaktion mit *MPTCP-WiFi*: die Verwendung des Internets.[2]

2.4.2. Aktoren und Stakeholder

Primäre Aktoren

Passagier: Der Passagier ist eine natürliche Person, der das Verkehrsmittel nutzt. Dieser Aktor existiert nicht bei allen Verkehrsmittelbetreibern.

ÖV³-IT-Gerät: Die IT-Geräte, welche die Betreiber des öffentlichen Verkehrs einsetzen, führen automatisierte Aufgaben durch. Dies können beispielsweise Kassensysteme, Videoüberwachungssysteme oder Kondukteur-Systeme sein.

[2] Auf welche Ereignisse *MPTCP-WiFi* zur Erreichung dieses Ziels reagiert, und damit auch in welche technischen Use Cases der Haupt-Use-Case unterteilt ist, wird in Kapitel 2.8 auf Seite 22 beschrieben.

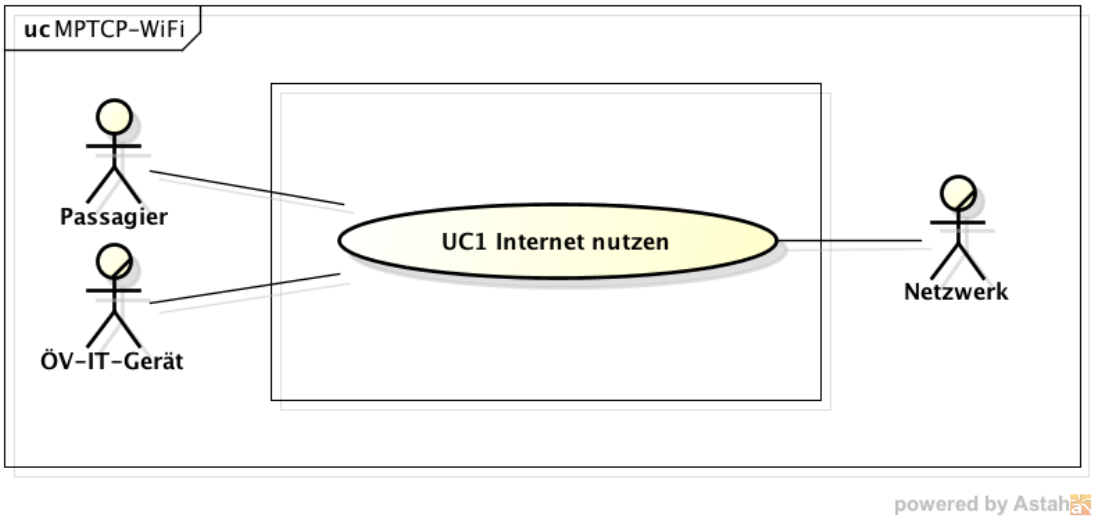

Abbildung 2.1.: Use-Case-Diagramm Haupt-Use-Case

Sekundäre Aktoren

Netzwerk: Das Netzwerk unterstützt die primären Aktoren beim Erreichen ihrer Ziele, indem es die Internetverbindung bereitstellt. Es sollen mehrere Netzwerke gleichzeitig genutzt werden, um die Nutzer (Passagier und ÖV-IT-Gerät) bestmöglich zu unterstützen. Es kann sich beim Netzwerk um ein Mobilfunknetzwerk (z. B. UMTS oder LTE[4]) oder ein lokales Funknetwerk (WiFi) handeln.

Stakeholder

Passagier: Der Anwender möchte sich auf seiner Fahrt mit dem Verkehrsmittel seine Zeit vertreiben oder produktiv arbeiten. Er möchte möglichst wenige (am liebsten keine), und wenn, dann möglichst kurze Verbindungsunterbrüche erleben.

IT-Gerät: Das IT-Gerät möchte seine eigens definierten Aufgaben erfüllen, die Zugriff auf bestimmte Server im Intranet des ÖV-Anbieters benötigen. Dies kann beispielsweise ein Abgleich mit Dateien auf einem Ticketserver sein oder das Heraufladen von

[4]Long Term Evolution

Kamera-Aufnahmen. Dazu verbindet sich das Gerät über das Internet mit den nötigen Intranet-Servern, beispielsweise über einen VPN[5]-Tunnel.

Betreiber des Internetzugangs: Er ist daran interessiert, dass der Zugang möglichst schnell und stabil für seine Kunden (die Betreiber des Verkehrsmittels) abläuft.

Betreiber des Verkehrsmittels: Er möchte durch das Anbieten von mobilem Internet seine Dienstleistungen aufwerten und entweder eine tiefere Kundenbindung schaffen oder seine eigenen Geräte mit dem Internet verbinden.

2.4.3. Use Case 1: Internet nutzen

Die primären Akteure des öffentlichen Verkehrs möchten von einem Verkehrsmittel aus auf Information im Internet zugreifen, Informationen versenden und dies mit stabilem und idealerweise möglichst schnellem Zugang.

Situationsbeschreibung

- Der Passagier liest anhand eines Infoblatts im Verkehrsmittel, dass das Transportunternehmen einen Internetzugang anbietet. Er nimmt sein Mobilgerät hervor und verbindet sich über WiFi mit dem Netz. Er empfindet dies als angenehmen Service des Transportunternehmens.

- Das IT-Gerät führt eine Aufgabe durch, für die es geschaffen wurde. Dazu muss es Informationen aus dem Internet abrufen oder Informationen über das Internet versenden: Das Überwachungssystem sendet die Überwachungsdaten der letzten Stunden auf einen Server des Transportunternehmens; das Kondukteursystem prüft, ob ein Fahrgast, der sein Generalabonnement vergessen hat, wirklich eines besitzt.

[5]Virtual Private Network

Voraussetzung

Der Passagier besitzt ein WiFi-fähiges Mobilgerät, das IT-Gerät eine WiFi-Schnittstelle. Der MPTCP-Router ist in Betrieb und über die Mobil-Interfaces mit dem Internet verbunden.

2.5. User Stories

Da der Nutzer die Vorteile von MPTCP nutzen möchte, sollen dazu folgende beide User Stories unterstüzt werden:

- Im Normalbetrieb soll der Verkehr der Nutzer über alle verfügbaren Interfaces mit maximal möglichem Durchsatz ablaufen.
- Der Nutzer kann TCP- und UDP[6]-Pakete verwenden.

2.6. Weitere Anforderungen

Die weiteren Anforderungen wurden im Meeting mit CloudGuard Software AG [10] besprochen.

Das oberste Ziel ist es, ein funktionsfähiges Produkt abzuliefern, welches nachbaubar und später praktisch nutzbar ist. Das funktionsfähige Produkt ist einer umfangreichen Vergleichsanalyse vorzuziehen. Das Produkt soll anschliessend von CloudGuard und seinen Partnern erweitert werden können, um es in der Zukunft im Produktiveinsatz nutzen zu können.

Es wurden genauere Anforderungen an die Funktionsweise des Produkts definiert:

1. Sämtliche IP-Pakete müssen über MPTCP versendet werden.

[6] User Datagram Protocol

2. TCP-Pakete müssen versendet werden, ohne TCP-over-TCP-Verschachtelungen zu verwenden.

3. Wenn ein Interface „down“ war und anschliessend wieder „up“ kommt, muss es wieder verwendet werden.

4. Wenn ein Interface „up“ ist, aber einen Durchsatz von 0 erreicht hatte und nun wieder einen höheren Durchsatz erreicht, muss es wieder verwendet werden.

5. Wenn über sämtliche Interfaces keine Daten gesendet werden können, müssen die Verbindungen, sobald ein Interface wieder nutzbar ist, neu aufgebaut werden.

6. Wenn das Interface nicht mehr verfügbar ist, das bisher zum Verbindungsaufbau genutzt wird, muss ein neues, zum Datacenter nutzbares Interface diese Funktion übernehmen.

7. Das Produkt muss auf ein Linux-System aufbauen.

Es ist gewünscht, dass das Produkt auf der Fahrzeughardware (*IPmotion CAR-A-WAN.coach Plus Vorserie*) bereits läuft, dies ist jedoch kein Muss-Kriterium. Ebenfalls als Kann-Anforderung liesse sich eine Interface-Logik implementieren, beispielsweise dass an Bahnhöfen nur (kostenloses) WiFi und kein (kostenpflichtiger) Mobilfunk verwendet wird. Nice to have wäre zudem ein Umgang mit Quality of Service, dass beispielsweise nach Paketart (Daten, Audio, Video) unterschieden wird und eine unterschiedliche Behandlung stattfindet.

2.7. Abgrenzung

Welche Uplinks an welchem geografischen Ort zur Verfügung stehen oder genutzt werden sollen, ist nur die Aufgabe vom Modem und nicht vom Projekt, muss daher nicht als Teil dieser BA betrachtet werden.[7]

[7] Besprochen am Meeting mit CloudGuard Software AG [10].

2.8. Technische Use Cases

Ergänzend zum Haupt-Use-Case gibt es verschiedene Use Cases aus technischer Sicht.

2.8.1. Technisches Use-Case-Diagramm

Das Use-Case-Diagramm in Abbildung 2.2 auf Seite 23 stellt die technischen Use Cases dar.

2.8.2. Aktoren

Primäre Aktoren

Nutzer: Der Nutzer ist ein Passagier oder ein ÖV-IT-Gerät. Er sendet Daten über das System an das Internet.

Internet-Teilnehmer: Der Internet-Teilnehmer ist ein Server oder ein anderes Gerät im Internet. Er empfängt Daten des Nutzers und sendet ihm Antworten zurück.

Mobilfunk-Interface des Access Nodes: Das Mobilfunk-Interface sendet und empfängt Daten über ein bestimmtes Mobilfunknetz. Es beeinflusst das System, indem es in ein Funkloch gerät oder wieder Empfang bekommt.

WiFi-Interface des Access Nodes: Das WiFi-Interface sendet und empfängt Daten über ein WiFi-Netz. Es beeinflusst das System, indem es Emfang zu einem WiFi-Netz bekommt oder den Empfang verliert.

Sekundäre Aktoren

Access Node: Der Access Node empfängt die Daten des Nutzers und sendet sie über mehrere Mobilfunk- und WiFi-Interfaces unter Verwendung von MPTCP an den Exit Node. Die Antworten des Internet-Teilnehmers werden auf dem umgekehrten Weg übertragen.

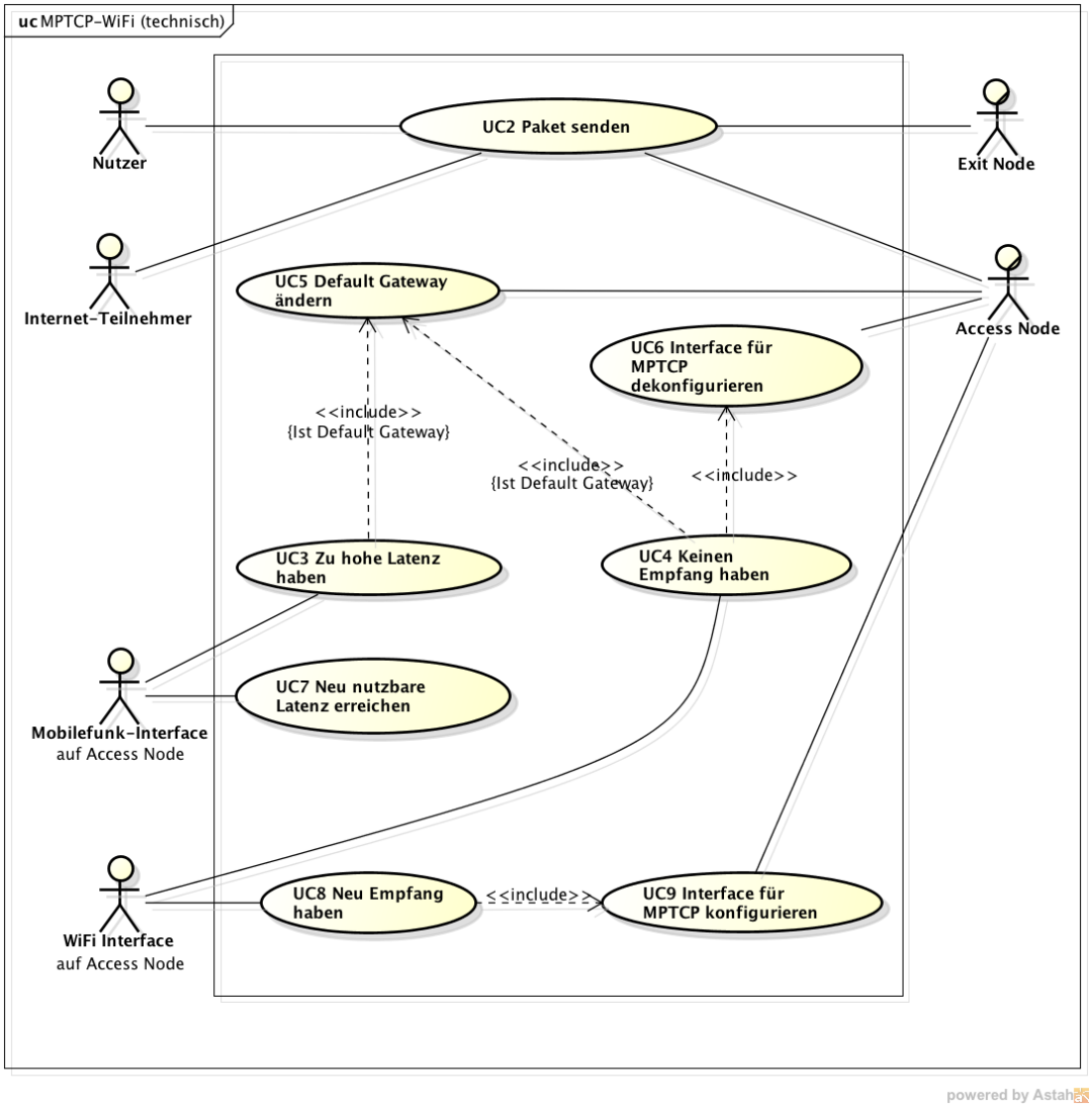

Abbildung 2.2.: Technisches Use-Case-Diagramm

Exit Node: Der Exit Node empfängt die Daten des Access Nodes unter Verwendung von MPTCP und sendet sie an den gewünschten Internet-Teilnehmer. Die Antworten des Internet-Teilnehmers werden auf dem umgekehrten Weg übertragen.

2.8.3. Use Case 2: Paket senden

Beschreibung: Der Nutzer oder der Internet-Teilnehmer sendet ein Paket über das System. Dieses Paket traversiert den Access Node sowie den Exit Node.

Beteiligte Akteure: Primär: Nutzer, Internet-Teilnehmer; Sekundär: Access Node, Exit Node

Auslöser: Ein Paket des Nutzers trifft auf dem Access Node ein.

Vorbedingungen: Der Access Node ist bereit, Pakete des Nutzers anzunehmen bzw. der Exit Node ist bereit, Pakete des Internet-Teilnehmers anzunehmen. Damit der Use Case zielführend ist, muss zudem eine Verbindung zwischen Access Node und Exit Node über ein oder mehrere Mobilfunk- und WiFi-Interfaces des Access Nodes bestehen.

Invarianten: Die Pakete anderer Nutzer und Internet-Teilnehmer dürfen nur geschwindigkeitsmässig beeinflusst werden. Die Verbindung zwischen Access Node und Exit Node darf durch die Ausführung des Use Cases nicht unterbrochen werden.

Nachbedingungen: Falls eine Verbindung zwischen Access Node und Exit Node besteht, muss mindestens ein Zustellungsversuch des Pakets unternommen worden sein.

Standardablauf: Der Nutzer sendet ein Paket an den Access Node. Das Paket wird an den Exit Node weitergeleitet. Der Exit Node leitet das Paket an den Internet-Teilnehmer. Alternativ: Der Internet-Teilnehmer sendet ein Paket an den Exit Node. Das Paket wird an den Access Node weitergeleitet. Der Access Node leitet das Paket an den Nutzer.

Alternative Abläufe: Wenn beim Access Node sämtliche Mobilfunk- und WiFi-Interfaces keine Verbindung haben: Der Nutzer sendet ein Paket an den Access Node. Der Use Case wird abgebrochen. Alternativ: Der Internet-Teilnehmer sendet ein Paket an

den Exit Node. Der Use Case wird abgebrochen. In diesem Fall sind die Modems noch aktiv, haben aber keinen Durchsatz. Pakete werden gedroppt, es gibt kein ICMP[8] unreachable.

2.8.4. Use Case 3: Zu hohe Latenz haben

Beschreibung: Das Mobilfunk-Interface hat eine Latenz, die höher als ein festgelegter Schwellwert ist, und ist damit nicht mehr praktisch nutzbar.

Beteiligte Akteure: Mobilfunk-Interface des Access Nodes

Verwendete Anwendungsfälle: Enthält Use Case 6: Interface für MPTCP dekonfigurieren.

Auslöser: Der Empfang zu einem Mobilfunk-Netzwerk wird schwächer oder bricht ab. Dadurch entsteht eine Latenz, die höher als ein definierter Schwellenwert liegt.

Vorbedingungen: Das Mobilfunk-Interface war mit einem Mobilfunk-Netzwerk verbunden.

Invarianten: Die Funktionalität anderer Interfaces darf nicht beeinträchtigt werden.

Nachbedingungen: Keine.

Standardablauf: Das Mobilfunk-Interface hat keinen Empfang mehr oder eine zu schwache Verbindung. Es geht nicht „down“, ist jedoch nicht mehr nutzbar. Das System achtet darauf, dass künftige Pakete über andere Interfaces gesendet werden, falls verfügbar.

2.8.5. Use Case 4: Keinen Empfang haben

Beschreibung: Das WiFi-Interface hat keinen Empfang mehr.

Beteiligte Akteure: WiFi-Interface des Access Nodes

[8] Internet Control Message Protocol

Verwendete Anwendungsfälle: Enthält Use Case 6: Interface für MPTCP dekonfigurieren. Enthält Use Case 5: Default Gateway ändern, falls das Interface der Default Gateway für den Access Node war.

Auslöser: Der Empfang zu einem WiFi-Netzwerk bricht ab. Das WiFi-Interface geht „down“.

Vorbedingungen: Das WiFi-Interface war mit einem WiFi-Netzwerk verbunden.

Invarianten: Die Funktionalität anderer Interfaces darf nicht beeinträchtigt werden.

Nachbedingungen: Keine.

Standardablauf: Das WiFi-Interface hat keinen Empfang mehr. Es geht „down“. Damit ist es nicht mehr nutzbar. Das System achtet darauf, dass künftige Pakete über andere Interfaces gesendet werden, falls verfügbar.

2.8.6. Use Case 5: Default Gateway ändern

Beschreibung: Der Default Gateway des Access Nodes wird auf das nächste noch nutzbare Interface geändert.

Beteiligte Akteure: Primär: Mobilfunk-Interface des Access Nodes, WiFi-Interface des Access Nodes; Sekundär: Access Node

Auslöser: Das Interface kann neu nicht mehr verwendet werden.

Vorbedingungen: Das Interface war der Default Gateway auf dem Access Node.

Invarianten: Es muss weiterhin ein Interface als Default Gateway eingestellt sein.

Nachbedingungen: Es ist ein Default Gateway auf dem Access Node eingestellt. Falls mindestens ein anderes Interface verwenden werden kann, ist der neue Default Gateway ein anderes Interface.

Standardablauf: Das Interface ist nicht mehr verwendbar. Auf dem Access Node wird ein anderes Interface als Default Gateway eingestellt.

Alternative Abläufe: Das Interface ist nicht mehr verwendbar. Es ist auch kein anderes Interface verwendbar. In diesem Falle ist es irrelevant, welches Interface zum Default Gateway wird – der Default Gateway kann auf ein anderes Interface gewechselt werden. Das bestehende Interface kann Default Gateway bleiben oder es kann zyklisch zwischen den Default Gateways gewechselt werden.

2.8.7. Use Case 6: Interface für MPTCP dekonfigurieren

Beschreibung: Die MPTCP-Konfiguration für das WiFi-Interface auf dem Access Node wird auf dem Access Node entfernt.

Beteiligte Akteure: Primär: WiFi-Interface des Access Nodes; Sekundär: Access Node

Auslöser: Das Interface ist „down“ gegangen.

Vorbedingungen: Das Interface war vorher auf dem Access Node für MPTCP konfiguriert.

Invarianten: Die anderen Interfaces müssen konfiguriert bleiben.

Nachbedingungen: Das Interface ist nicht mehr für MPTCP konfiguriert.

Standardablauf: Das Interface ist „down“ gegangen. Es wird für MPTCP dekonfiguriert.

2.8.8. Use Case 7: Neu nutzbare Latenz erreichen

Beschreibung: Das Mobilfunk-Interface hat neu eine Latenz, die unter einem festgelegten Schwellwert liegt, und ist damit wieder für MPTCP nutzbar.

Beteiligte Akteure: Mobilfunk-Interface des Access Nodes

Auslöser: Das Mobilfunk-Interface hat neu nutzbaren Empfang zum Mobilfunk-Provider.

Vorbedingungen: Das Mobilfunk-Netzwerk hatte zu schwachen Empfang.

Invarianten: Die Funktionalität anderer Interfaces darf nicht beeinträchtigt werden.

Nachbedingungen: Keine.

Standardablauf: Das Mobilfunk-Interface erreicht genügenden Empfang mit dem Mobilfunk-Provider. Es wird nun wieder vom System verwendet.

2.8.9. Use Case 8: Neu Empfang haben

Beschreibung: Das WiFi-Interface hat nun Empfang und ist damit für MPTCP nutzbar.

Beteiligte Akteure: WiFi-Interface

Verwendete Anwendungsfälle: Enthält Use Case 11: Interface für MPTCP konfigurieren.

Auslöser: Das WiFi-Interface hat neu Empfang zu einem WiFi-Netzwerk. Es ist nun „up“.

Vorbedingungen: Das WiFi-Interface war „down“.

Invarianten: Die Funktionalität anderer Interfaces darf nicht beeinträchtigt werden.

Nachbedingungen: Keine.

Standardablauf: Das WiFi-Interface ist neu in Reichweite eines WiFi-Netzwerks. Es geht „up“ und verbindet sich mit diesem Netzwerk. Es kann nun verwendet werden.

2.8.10. Use Case 9: Interface für MPTCP konfigurieren

Beschreibung: Das WiFi-Interface wird konfiguriert, damit es für MPTCP genutzt werden kann.

Beteiligte Akteure: WiFi-Interface

Auslöser: Das WiFi-Interface ist nun „up“.

Vorbedingungen: Das WiFi-Interface war nicht für MPTCP konfiguriert. Es ist nun verfügbar und hat eine zugewiesene IP-Adresse und eine zugewiesene Subnetzmaske.

Invarianten: Die Funktionalität anderer Interfaces darf nicht beeinträchtigt werden.

Nachbedingungen: Das WiFi-Interface ist nun für MPTCP konfiguriert.

Standardablauf: Die nötigen Daten werden ausgelesen und berechnet sowie die nötigen Änderungen am System ausgeführt.

Alternative Abläufe: Keine.

Anmerkung: Mobilfunk-Interfaces werden bei Systemstart bereits für MPTCP konfiguriert und behalten ihre Konfiguration während der Systemlaufzeit. Sie lösen daher Use Case 9: Interface für MPTCP konfigurieren während der Laufzeit nie aus.

3. Projektplan

3.1. Einführung

3.1.1. Zweck

In diesem Dokument werden planungsspezifische Punkte zum Projekt *Multipath over Wireless Networks für mobiles WiFi* beschrieben. Dazu gehören unter anderem die Produktbeschreibung, das Risikomanagement, die Projektorganisation und auch die Qualitätsmassnahmen.

3.1.2. Gültigkeitsbereich

Dieses Dokument dient als zeitlicher Leitfaden für das Projekt. Es ist gültig für die Laufzeit der Bachelorarbeit, Frühjahrssemester 2014. Falls die Arbeit nach diesem Zeitpunkt fortgeführt oder erweitert wird, würde dies in einem neuen Projekt behandelt werden und ist daher nicht Teil dieses Projektplans.

3.2. Projektübersicht

Das Projekt hat zum Ziel, einen praxisgerechten Prototyp für einen MPTCP-Einsatz für mobiles WiFi über mehrere Mobilfunkanbieter umzusetzen sowie zu testen und zu dokumentieren. Die genaue Aufgabenstellung findet sich im Kapitel A auf Seite I.

3.2.1. Lieferumfang

Das Endprodukt besteht aus mehreren Komponenten:

- Prototyp gemäss Aufgabenstellung (künftig genannt: *MPTCP-WiFi*)
- Dokumentation des Prototyps inkl. Installationsanleitung
- Testdokumentation mit Testergebnissen
- Bachelorarbeitsdokument gemäss Richtlinien der HSR[1]

3.2.2. Annahme und Einschränkungen

Da der Prototyp auf der bestehenden Mobile-WiFi-Plattform von CloudGuard Software AG (*IPmotion CAR-A-WAN.coach Plus Vorserie*) bestehen soll, wird davon ausgegangen, dass diese dem Projektteam zu Beginn des Projekts bereit gestellt wird. Um die geforderten Tests in Jungfraubahnen und Postbussen umsetzen zu können, müssen des Weiteren Testfahrten durch diese Unternehmen ermöglicht werden. Da die Mobilfunkschnittstellen Teil des Prototyps sind, müssen SIM-Karten mit nötigem Datenvolumen bereitgestellt werden. Die Standards von MPTCP beim IETF[2] [2] sind momentan, zum Zeitpunkt der Aufgabenstellung, noch als experimentell gekennzeichnet und können sich daher zu jedem Zeitpunkt ändern. Die Lieferobjekte können daher möglicherweise nicht den aktuellen Stand des Standards unterstützen.

3.3. Projektorganisation

Das Projektteam besteht aus Jannis Grimm und Thomas Zeender, Studenten an der HSR. Die Funktion des Betreuers wird von Prof. Beat Stettler, Dozent an der HSR, INS Institute for Networked Solutions, übernommen. Kunde und Projektpartner ist CloudGuard Software AG, Pfäffikon SZ.

[1] Hochschule für Technik Rapperswil
[2] Internet Engineering Task Force

3.4. Managementabläufe

3.4.1. Zeitliche Planung

Die Zeiterfassung und Planung wird in der Projektmanagementsoftware *Redmine* vorgenommen. Dort können alle relevanten Information zu Meilensteinen, Zeitaufwand und Zeitplanung festgehalten und bearbeitet werden. Für die Verarbeitung und Auswertung der Daten kann ein Report erstellt und anschliessend exportiert werden.

Der Zeitrahmen des Projekts ist das Frühjahrssemester 2014, bestehend aus 16 Wochen. In den ersten 14 Wochen („Vorlesungszeitraum“) werden jeweils pro Teammitglied 20 Stunden Arbeit budgetiert, in den Wochen 15 und 16 („Vorlesungsfreier Zeitraum“) werden je 30 Stunden pro Teammitglied eingeplant. Nach Abgabe der der Arbeit dienen pro Teammitglied 20 Stunden zur Vorbereitung zur mündlichen Bachelorarbeitsprüfung. Damit wird die Vorgabe von 12 ECTS[3,4] pro Teammitglied erfüllt.

In Tabelle 3.1 auf Seite 33 sind die Meilensteine des Projekts beschrieben.

Arbeitspakete

Die Arbeitspakete werden in Redmine festgehalten. Zu Beginn des Projekts wird eine voraussichtliche Planung inklusive Meilensteine und Arbeitspakete erstellt. Jede Woche sowie beim Erreichen von Meilensteinen werden iterativ die folgenden Arbeitspakete der kommenden Woche bzw. des kommenden Meilensteins geprüft und nötigenfalls in Absprache mit dem Projektbetreuer angepasst.

3.5. Risikomanagement

Die Risiken wurden zu Beginn des Projektes abgeschätzt. In regelmässigen Abständen werden die Risiken neu beurteilt, spätestens bei Abschluss eines Meilensteins, um auf Änderungen zeitnah reagieren sowie konstant die Risiken minimieren zu können.

[3] European Credit Transfer and Accumulation System
[4] 1 ECTS entspricht 30 Stunden Aufwand

Nr.	Nach Phase	Beschreibung	Woche
MS1[a]	Inception	Projektplan fertig	SW 03[b]
MS2	Elaboration	Analyse, Requirementspezifikation und Domainanalyse fertig	SW 08[c]
MS3	Construction 1	1. Prototyp (Einzelkomponenten funktionieren: 1. Modem–Internet, 2. MPTCP in Testumgebung, 3. Exit Node–Internet, 4. Verbindung über *IPmotion CAR-A-WAN.coach Plus Vorserie*, 5. Umgangskenntnis *IPmotion CAR-A-WAN.coach Plus Vorserie*	SW 13
MS4	Construction 2	Integration der Einzelkomponenten (Verbindung Client–Access Node–MPTCP–Exit Node–Internet) abgeschlossen	SW 16
MS5	Transition	Bereit zur Abgabe	SW 17

[a] MS: Meilenstein
[b] SW: Semesterwoche
[c] Dieser Meilenstein wurde nicht erreicht und daher während des Projekts auf SW 09 verschoben.

Tabelle 3.1.: Meilensteine

3.6. Infrastruktur

3.6.1. Software

Folgende Software wird im Verlauf der Bachelorarbeit verwendet:

- Git: Versionierung von Programmcode und Dokumentation
- gcc: Compiler
- LaTeX: Dokumentation
- Redmine: Projektverwaltung
- Dropbox: Austausch von Dokumenten
- astah*: Diagramme

- Time Machine: Backups
- Linux Kernel MultiPath TCP: MPTCP-Referenzimplementation der Université catholique de Louvain
- ESXi: Virtualisierungssystem für virtuelle Maschinen der Prototyp- und Testumgebung

Die Software, die in *IPmotion CAR-A-WAN.coach Plus Vorserie* zum Einsatz kommt, ist zu Beginn des Projekts noch unbekannt, die Analyse ist Teil des Projektes.

3.6.2. Hardware

Folgende Hardware kommt im Verlauf der Bachelorarbeit zum Einsatz:

- 2× privates MacBook Pro
- 2× HSR-Computer
- 2× USB-LTE-Stick (*4G Systems XS-Stick W100*)
- Privater USB WiFi Adapter
- SIM-Karte Swisscom
- SIM-Karte Orange
- Apple Mac Mini Server (ESXi)
- DD-wrt Router
- Fahrzeughardware: Multifunktions-WiFi-Router (*IPmotion CAR-A-WAN.coach Plus Vorserie*), Vorabversion des LTE-Modells, dazu 3× externe Antenne

3.7. Qualitätsmassnahmen

3.7.1. Dokumentation

Gegenlesen

Die Dokumentation wird vom Teampartner sowie von externen Personen gegengelesen, um Fehler und unverständliche Formulierungen zu finden und zu minimieren.

Git

Zur Versionierung der Dokumentation wird Git eingesetzt. Somit ist die Betrachtung von früheren Versionen möglich und die Projektpartner können sämtliche Änderungen des Teams nachvollziehen und prüfen.

3.7.2. Projektmanagement

Projektfortschrittsdiagramm

Zum Feststellen des Projektfortschrittes wird Redmine genutzt. Dort kann im Gantt-Diagramm übersichtlich der Projektfortschritt festgestellt werden. Im Projektteam wird der Projektfortschritt wöchentlich betrachtet und Abweichungen besprochen.

Sitzungen mit Betreuer

Es findet mindestens wöchentlich eine Sitzung mit dem Betreuer statt. In dieser Sitzung werden Änderungen und Fragen besprochen sowie Entwürfe von Dokumenten betrachtet.

Kommunikation

Es findet, soweit möglich, direkte Kommunikation mit den Autoren der Vorarbeit, den Autoren der Software Linux Kernel MultiPath TCP sowie CloudGuard Software AG statt. So können gegenseitige Interessen, Fragen, Issues und Ähnliches ausgetauscht und bearbeitet werden.

3.7.3. Entwicklung

Dokumentation

Die Entwicklung der Umgebung, insbesondere deren Installation und Konfiguration, wird in der Dokumentation festgehalten. So sind sämtliche genutzten Komponenten stets nachvollziehbar, nachbaubar und für den Notfall austauschbar.

Git

Programmcode, Installations- und Konfigurationsdokumente werden mittels Git verwaltet. Somit ist das Nachvollziehen von Änderungen möglich und die Projektpartner können sämtliche Aktionen des Teams nachvollziehen und prüfen.

Code Reviews

Wo Programmcode erstellt wird, findet mindestens vor Ende des Meilensteins ein gegenseitiger Code Review statt. Fehler und Verbesserungsvorschläge werden dem Code-Autor innerhalb des Projektteams weitergeleitet, damit dieser sensibilisiert wird und ein Refactoring vornehmen kann.

3.7.4. Testing

Integrationstests

Integrationstests sind Teil der Aufgabenstellung und werden daher an dieser Stelle nicht genauer betrachtet. Sie sind im Kapitel 6.5 auf Seite 88 dokumentiert.

Testdefinition

Die Testabläufe werden jeweils im Voraus definiert und dokumentiert. So können diese auf Sinnhaftigkeit überprüft und die Tests nachvollzogen sowie nachgestellt werden.

4. Analyse der Gegebenheiten

4.1. Vorarbeit von anderem Projektteam

Der Fokus der Vorarbeit [1], die als Studienarbeit von Nils Caspar und Simon Huber durchgeführt wurde, lag auf der Untersuchung und Erarbeitung des grundsätzlichen Konzepts des gesamten Systems sowie auf Referenzimplementation mit eigener Hardware.

4.1.1. Analyse der Vor-Studienarbeit

Hardware-Analyse der Vor-Studienarbeit

Bezüglich Hardware wurde in der Vor-Studienarbeit analysiert, dass die von CloudGuard gegebene Hardware aus einem Multifunktions-WiFi-Router plus externen Antennen besteht. Der Multifunktions-WiFi-Router enthält folgende Komponenten:

- Wireless-Bridge
- Switch
- Router
- Server mit Diensten: DHCP[1], möglicherweise[2] VPN, Weitere[3]
- Modems: LTE, UMTS, GPRS[4]

[1] Dynamic Host Configuration Protocol
[2] sic, was mit „möglicherweise“ gemeint ist, ist nicht dokumentiert
[3] nicht dokumentiert
[4] General Packet Radio Service

Diese Analyse kann übernommen werden, es gilt jedoch zu überprüfen, ob das Gerät VPN nun integriert hat oder nicht – und falls ja, ob als Client oder als Server. Da das Gerät in der Vor-Studienarbeit lediglich bis zu dieser Ebene analysiert und anschliessend mit eigener Prototyp-Hardware gearbeitet wurde, ist in der vorliegenden Arbeit eine weitergehende Analyse des Gerätes nötig.

Link-Qualitätsanalyse der Vor-Studienarbeit

Bezüglich Link-Qualität wurde in der Vor-Studienarbeit analysiert, wie die „Qualität“ eines Links gemessen werden kann. Qualität wird hierbei definiert als ein Verhältnis von vier Faktoren:

- Packet Loss
- Round Trip Time
- Bandbreitenlimitierung durch Traffic Shaping
- Bandbreitenlimitierung durch physikalische Einschränkungen

Einem zu hohen Packet Loss wird dabei durch den Congestion-Avoidance-Algorithmus von TCP entgegengewirkt. Dort wird nach jedem Packet Loss die Sendegeschwindigkeit durch Senken des Congestion Windows vermindert, was jedoch auch für schlechteren Durchsatz sorgt. Es wird eine Formel zur Berechnung des Durchsatzes unter Berücksichtigung von Paketgrösse, Round Trip Time, Retransmission Timeout und Wahrscheinlichkeit eines Paket Loss angegeben.

Je höher die Round Trip Time, desto geringer ist der Durchsatz. Die ACK[5]s werden später gesendet, wodurch die Latenz ansteigt.

Es wird vorgeschlagen, regelmässige Pings über alle verfügbaren Links durchzuführen. Damit könnte die Round Trip Time gemessen sowie der Packet Loss festgestellt werden. Dadurch entsteht bei ICMP-Paketen von 60 Bytes ein Datenverkehr von etwa 10 Megabytes. Dies wäre ein vertretbarer Overhead.

[5] Acknowledgement-Paket in TCP

Es besteht noch keine Klarheit darüber, in welchem Rahmen die angegebenen Analysen für die Arbeit benötigt werden, da die Aufgabenstellung nicht von einer Messung der Link-Qualität spricht. Stattdessen ist detaillierter angegeben, welche Parameter gemessen werden sollen. Gewisse Tätigkeiten, wie die Kapazitätsfeststellung, werden zudem von MPTCP selbst übernommen. Die Idee, regelmässige Pings zu versenden, kann jedoch zur Feststellung der momentan verfügbaren und damit für MPTCP bereitstehenden Links verwendet werden.

Zudem wurde analysiert, welche Daten zur Verbindungsqualität bei Cisco-Routern erhoben werden. Da in den Fahrzeugen jedoch keine Cisco-Geräte eingesetzt werden, ist dies für dieses Projekt unerheblich.

Realisierungsanalyse der Vor-Studienarbeit

Bezüglich Realisierung wurde in der Vor-Studienarbeit analysiert, dass nur eine Lösung auf OSI[6]-Modell[7] Layer 3[8] und 4[9] infrage kommen, um die gegebenen Voraussetzungen erfüllen zu können.

Folgende Lösungen wurden in Betracht gezogen:

- Multipath Routing (OSI Layer 3)
- Host Standby Router Protocol (OSI Layer 3)
- MPTCP (OSI Layer 4)

Als Ergebnis der Analyse zeigte sich, dass MPTCP zur Erfüllung der Vorgaben am besten geeignet ist.

Da MPTCP in der Aufgabenstellung dieser Bachelorarbeit bereits gegeben ist und das Ziel dieses Projektes ist, eine Lösung über Routing zu ersetzen, ist ein Prüfen und Hinterfragen dieser Analyse kein Teil der vorliegenden Arbeit.

[6] Open Systems Interconnection
[7] ISO OSI 7-Schichtenmodell für Netzwerkprotokolle
[8] Network Layer
[9] Transport Layer

MPTCP-Analyse der Vor-Studienarbeit

MPTCP ist eine Erweiterung von TCP und wird von einer IETF-Arbeitsgruppe im experimentellen Standard RFC[10] 6824 spezifiziert. Die parallele Nutzung mehrerer Wireless-Technologien ist ein vorgesehener Anwendungsfall von MPTCP. Neben dem Standard existiert eine Referenzimplementation der Université catholique de Louvain in Form eines angepassten Linux-Kernels.

MPTCP ändert lediglich die Header, die mit den TCP-Paketen mitgeschickt werden, und ist daher gegenüber der anderen Schichten transparent. Nach dem TCP-Handshake kann MPTCP automatisch eingesetzt werden, wenn beide Kommunikationspartner den Standard unterstützen. Der Verbindungsaufbau findet über den konfigurierten Default Gateway statt.

Die Netzwerkpakete-Analysesoftware *Wireshark* unterstützt MPTCP bereits. Für *netstat* und *tcpdump* gibt es angepasste Versionen.

MPTCP bietet Konfigurationsmöglichkeiten in geringem Rahmen an: Mittels `multipath off` wird ein bestimmter Link nicht mehr verwendet, mittels `multipath backup` wird er nur genutzt, wenn alle anderen Links gerade nicht verwendet werden können. Während der Verbindung kann eine Prioritätsänderung über das Signal `MP_PRIO` angefordert werden, womit kostenpflichtige oder langsame Links benachteiligt werden können.

Es kommt vor, dass Sicherheitssoftware ihr unbekannte TCP-Header, darunter oft MPTCP-Header, entfernt. Dies verunmöglicht eine Nutzung von MPTCP in diesen Umgebungen.

Diese Analyse deckt sich auch mit den Erkenntnissen des Projektteams der vorliegenden Arbeit, nachzulesen im Kapitel 4.2 auf Seite 44. Es sei an dieser Stelle jedoch darauf hingewiesen, dass das Gerät, da es ein Router und kein Endgerät ist, abweichend zur Analyse in der Vor-Studienarbeit nicht den Default Gateway für jegliche neuen Verbindungen nutzt. Es kann über die Routing-Tabelle einen beliebigen Link wählen.

[10] Request for Comments

Implementationsanalyse der Vor-Studienarbeit

Da bisher nur eine sehr geringe Anzahl von Servern und Clientgeräten MPTCP nutzt, müssen zwei Proxy-Geräte eingesetzt werden, eines davon im Fahrzeug und das andere in einem Rechenzentrum. Über MPTCP wird zwischen ihnen eine Abdeckung über mehrere Mobilfunk-Provider erreicht.

Dies liesse sich beispielsweise über VPN realisieren, mit welchem ein Paketweiterleitungs-Tunnel über TCP eingerichtet werden kann. Von TCP über TCP wird jedoch in der Vor-Studienarbeit abgeraten, sodass diese Lösung nur für Nicht-TCP-Pakete genutzt werden sollte.

Für TCP-Pakete ist eine Weiterleitung über einen SOCKS[11]-Proxy machbar. Dieses tauscht die unteren Layer aus, statt die Pakete zu verpacken, womit Traffic-Overhead eingespart und die TCP-über-TCP-Problematik vermieden wird. Die Lösung über SOCKS-Proxy funktioniert ausschliesslich für TCP-Pakete.

Aufgrund dieser Erkenntnisse sollte eine Kombination aus beiden Techniken genutzt werden, je nach Protokoll auf Layer 4: TCP-Pakete werden über einen SOCKS-Proxy geleitet, während Nicht-TCP-Pakete (beispielsweise Pakete über UDP) über die VPN-Verbindung gleitet werden.

Diese Lösung deckt sich auch mit den Implementationsvorschlägen des Projektteams der vorliegenden Arbeit. Sie kann damit für dieses Projekt übernommen werden und wird in den folgenden Abschnitten der Analyse weiter untersucht. Da sie in der Vorarbeit bereits als bevorzugte Variante evaluiert wurde, ist eine Neuevaluation kein Bestandteil der vorliegenden Arbeit.

Ausfallsicherheits- und Skalierungsanalyse der Vor-Studienarbeit

Ebenfalls wurde in der Vor-Studienarbeit betrachtet, wie man durch Redundanz dem Risiko eines Ausfalls entgehen könnte und durch welche Mittel das Gesamtsystem skalierbar gestaltet werden könnte. Dafür wird ein Cluster aus Relay-Servern benötigt, die redundant angebunden werden. Es sollen Load Balancer eingesetzt werden. Die

[11] Sockets (SOCKetS)

Aufteilung der Multifunktions-WiFi-Router auf die Relay-Server-Cluster könnte per Round-Robin-DNS[12] erfolgen.

Die Aspekte Ausfallsicherheit und Skalierung sind jedoch kein Bestandteil der Aufgabenstellung dieser Arbeit und wurden daher im Rahmen dieses Projektes nicht weiter analysiert. Dieser Ansatz könnte in einer Folge-Arbeit genauer untersucht werden.

4.1.2. Prototyp der Vor-Studienarbeit

Software

Der Prototyp nutzt *OpenVPN*[13] für VPN. Für SOCKS wird auf dem Fahrzeuggerät *redsocks*[14], auf dem Relay-Server *Dante*[15] eingesetzt. Um Nicht-TCP-Pakete über ein anderes Gateway zu senden, wird *iptables* genutzt.

SOCKS

Die SOCKS-Server-Software *Dante* wurde mit Standardkonfiguration übernommen, der Port wurde jedoch von 1080 auf 11371 geändert, da dieser an der HSR Hochschule für Technik Rapperswil nicht geblockt ist.

VPN

Die VPN-Software *OpenVPN* wurde von UDP auf TCP umgestellt, der Port aus obigen Gründen von 1194 auf 5228 geändert und NAT[16] eingerichtet.

[12] Domain Name System
[13] Versionsnummer unbekannt
[14] Versionsnummer unbekannt
[15] Versionsnummer unbekannt
[16] Network Address Translation

Multifunktions-WiFi-Gerät

Es wurde nicht die finale Hardware von CloudGuard genutzt, sondern durch einen Laptop mit zwei USB-LTE-Modems ersetzt.

Funktionsfähigkeit

Gespräche mit dem Betreuer der Vorarbeit haben ergeben, dass der Prototyp der Vorarbeit zwar mit TCP gut funktioniert hat, ein Versenden von UDP-Paketen jedoch nicht möglich war. Dies ist jedoch nicht dokumentiert.

4.1.3. Fazit bezüglich der Vor-Studienarbeit

Gemäss dem Bericht der Verfasser der Vor-Studienarbeit konnten die Anforderungen zwar komplett erfüllt werden, jedoch nicht alle selbstgesetzten Ziele. Zudem konnten aus Zeitgründen nur Teile der Tests durchgeführt werden.

Auf den dokumentierten Erkenntnissen der Analyse der Vorarbeit kann in dieser Arbeit aufgebaut werden. Der Prototyp ist jedoch nur spärlich dokumentiert, die Konfigurationen enthalten nur teilweise Kommentare, Versionsnummern der eingesetzen Software fehlen. Der Prototyp ist dem Projektteam dieser Arbeit auch nicht zugänglich. Daher besteht keine Möglichkeit, den Prototyp zu übernehmen, es muss im Rahmen dieser Arbeit ein neuer Prototyp erstellt werden.

4.2. MPTCP

4.2.1. Einführung in MPTCP

MPTCP ist ein experimenteller TCP-Standard. Er erweitert TCP um die Fähigkeit, mehrere Interfaces gleichzeitig für eine TCP-Verbindung benutzen zu können. Dabei muss nicht zwingend mehr als ein Interface verwendet werden, ein transparentes Umschalten ist ebenfalls möglich.

4.2.2. Vorteile

Durch die Verteilung der Daten bei MPTCP können sowohl die Performanz als auch die Redundanz erhöht werden. Von diesen Vorteilen kann in mehreren Szenarien profitiert werden:

Mobile: Im Mobile-Bereich wird insbesondere vom transparenten Umschalten profitiert. Wenn man Zuhause sitzt, möchte man ausschliesslich die WiFi-Verbindung nutzen. Beim Verlassen des Hauses kann, dank dem transparenten Handover, die aktive Verbindung automatisch auf das Mobilfunk-Netz umgeschaltet werden.

Server: Es ist unwahrscheinlich, dass der Server häufig sein eigenes Netz verlässt. Er kann jedoch durch den Performanz- und Redundanz-Zuschub profitieren. Selbst wenn es vom Betreiber unerwünscht ist, selbst MPTCP zu verwenden, kann er diesen Dienst dennoch seinen Clients anbieten.

Hybrid Postauto: Beim Postauto-Szenario werden die Vorteile der Performanz und der Redundanz kombiniert. Beim Wegfallen eines Interfaces (Funkloch Betreiber A, aber nicht Betreiber B) können die Verbindungen von A auf B weitergeführt werden. Sind beide Betreiber erreichbar, kann ihre Performanz kombiniert werden.

4.2.3. Nachteile

MPTCP bringt durch seine Neuheit auch Nachteile mit sich. So ist es noch nicht im Grosseinsatz erprobt und auch noch nicht weit verbreitet. Auch müssen zum Einsatz die Gegenparteien über MPTCP verfügen, was heute auch noch nicht in grösserem Umfang der Fall ist. Die wohl verbreiteste Implementation ist die des Smartphone-Betriebssystems *iOS 7* von Apple, welche im folgenden Abschnitt 4.2.4 genauer bertachtet wird.

4.2.4. Betrachtung der iOS-Implementation

iOS 7 hat nach Apples Supportseite [3] MPTCP implementiert. In einem Test, bei dem sich ein iPhone 5s auf einen anderen Host verbindet, hat es bei den TCP-Optionen kein

MP_CAPABLE[17] angehängt, wie in Abbildung 4.1 auf Seite 46 mit einem Capture zu sehen ist.

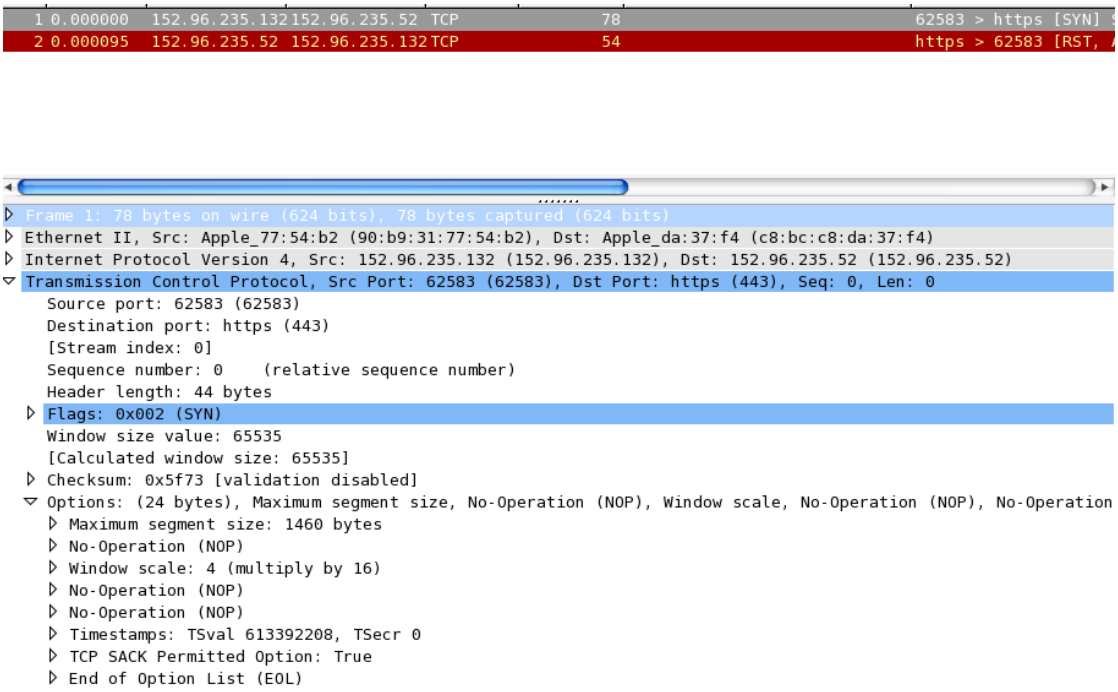

Abbildung 4.1.: iOS MPTCP

Dies steht im Kontrast zu einem Verbindungsaufbau, bei welchem der mögliche Einsatz von MPTCP gekennzeichnet wird. Siehe in Abbildung 4.2 auf Seite 47, wo im SYN-Paket die Option MP_CAPABLE besteht.

Ein Artikel [4] der Université catholique de Louvain beschreibt, wie MPTCP in bestimmten Paketen von iOS 7 gefunden wurde, dies jedoch nur beim Verbindungsaufbau zu den Servern von Siri geschieht. Daher ist iOS 7 zwar MPTCP-fähig, MPTCP jedoch nicht für die Zwecke der vorliegenden Arbeit nutzbar.

[17] Kennzeichnet, dass das Gerät MPTCP-fähig ist

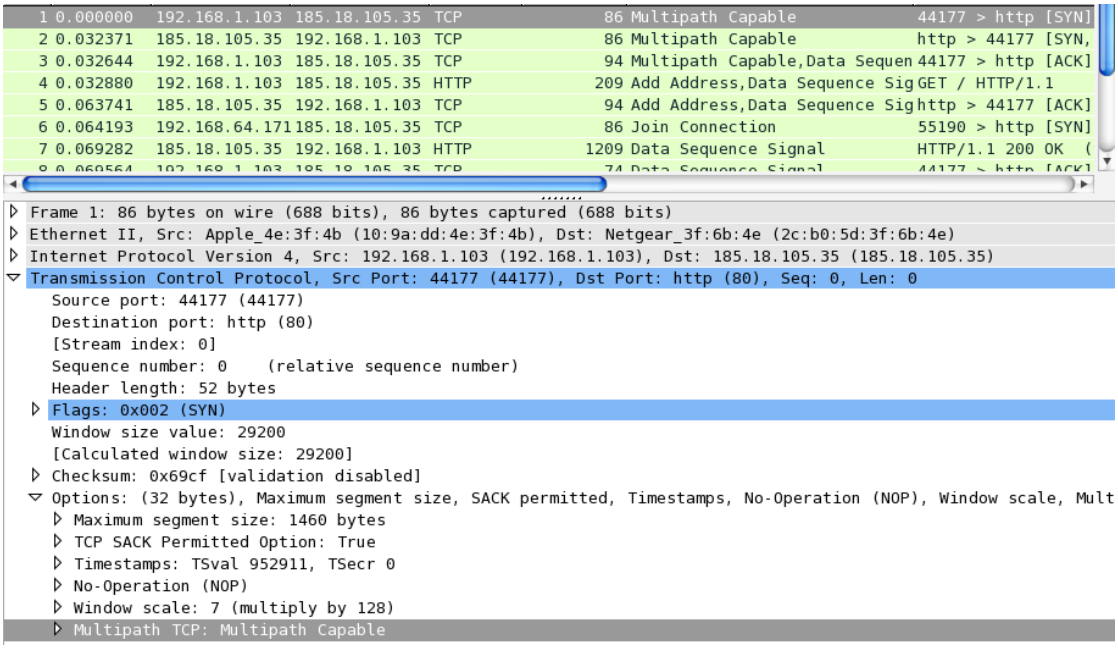

Abbildung 4.2.: Verbindung mit MPTCP

4.2.5. Active Internet-Draft

Der experimentale RFC 6824 [5] beschreibt, wie MPTCP umgesetzt werden soll. TCP ist momentan limitiert auf jeweils einzelne Verbindungen von Host zu Target. Es kann jedoch vorkommen, dass ein Gerät mehrere Pfade zu einem Ziel hat. So kann beispielsweise ein Server über mehrere Interfaces verfügen oder ein Mobilgerät kann sowohl über Mobilfunk als auch über WiFi auf das Internet zugreifen. MPTCP soll dies berichtigen.

In den folgenden Abschnitten soll kurz beschrieben werden, auf welche Weise MPTCP verschiedene Aktionen, beispielsweise den Verbindungsaufbau, regelt.

Terminologie

Path: Eine Sequenz von Links zwischen Sender und Empfänger, welcher durch vier Tupel beschrieben wird: Quellen- und Zieladresse, Quellen- und Zielport.

Subflow: Ein Subflow ist eine Teilverbindung der gesamten MPTCP-Verbindung. Ein

Subflow wird ähnlich wie eine normale TCP-Verbindung aufgebaut und terminiert.

(MPTCP)-Verbindung: Dies ist ein „Set“ von einem oder mehreren Subflows, die zusammen eine Verbindung ergeben, über welche eine Applikation kommunizieren kann. Es besteht eine Eins-zu-eins-Zuteilung zwischen Verbindung und einem Applikations-Socket.

Token: Ein Token ist eine lokal eindeutige Identifikationsnummer für eine Multipath-Verbindung. Er wird auch „Connection ID“ genannt.

Host A: Host A ist ein MPTCP-fähiger Teilnehmer und initialisiert die Verbindung.

Host B: Host B ist ein MPTCP-fähiger Teilnehmer und ist Ziel der von Host A initialisierten Verbindung.

Initialisierung einer MPTCP-Verbindung

Der Verbindungsaufbau bei MPTCP läuft gleich ab wie bei TCP. Dem üblichen SYN[18]-, SYN/ACK- und ACK-Handshake wird noch eine MP_CAPABLE-TCP-Option angehängt, welche die Kompatibilät der Teilnehmer anzeigt.

Zuerst wird sichergestellt, dass beide Teilnehmer MPTCP unterstützen. Danach werden MPTCP-relevante Informationen in der MP_CAPABLE-Option ausgetauscht, wie beispielsweise eine kryptografische Verifizierung der Gegenstelle.

Weitere Informationen befinden sich in [5, Abs. 3.1].

Hinzufügen neuer Subflows

Besteht eine Verbindung, können mit MPTCP neue Subflows hinzugefügt werden. Diese werden anhand der Option MP_JOIN angegeben.

Wenn Host A eine neue Verbindung von einer seiner Adressen zu einer von Host B aufbauen möchte, gibt er dies mithilfe der Option an. Die Angaben der neuen Verbindungen

[18] Synchronize-Paket in TCP

werden anhand eines HMAC[19] gesichert. Der Seed für den HMAC sind die Keys, die bei der Initialisierung ausgetauscht wurden, sowie zufällige Zahlen (nonces) in der Join-Nachricht.

Weitere Informationen befinden sich in [5, Abs. 3.2].

Über neue Adressen informieren

Erhält Host A oder Host B ein neue Adresse, möchte er unter Umständen nicht gleich eine Verbindung initialisieren, oder es kann nicht, da zwischen beiden ein NAT liegt.[20] Mit der Option ADD_ADDR kann Host A oder B den jeweiligen anderen Host über zusätzliche Adressen informieren. Darüber kann auch festgestellt werden, ob ein NAT die Adressen ändert, da in der Anfrage die Adresse des NAT steht und in der Option die des NAT-Clients.

Weitere Informationen befinden sich in [5, Abs. 3.4.1].

Verlust Default Gateway

Ein Gerät ohne MPTCP verwendet den Default Gateway für den Zugang ins Internet. Bei MPTCP-Systemen werden mehrere Default Gateways eingetragen. Dies ist notwendig, damit die einzelnen Interfaces unabhängig voneinander eine Verbindung ins Internet aufbauen können. Wird eine MPTCP-Verbindung initiiert, werden zusätzliche Subflows zugeschaltet.

Offen ist die Frage, was geschieht, wenn das Interface mit dem Haupt-Default-Gateway ausfällt. In Tests, die vom Projektteam als Teil der Analyse mit der Implementation von Université catholique de Louvain durchgeführt wurden, zeigte sich, dass bei Wegfallen des Default Gateway keine neuen Verbindungen über den zweiten Gateway initiiert werden. In diesem Fall muss gewartet werden bis das erste Interface wieder aktiv wird oder der Default-Pfad auf das zweite Interface geschaltet wird.

Dieses Umschalten könnte mit einem dynamischen Routingprotokoll umgesetzt werden. In

[19] Keyed-Hash Message Authentication Code
[20] Nur der Teilnehmer, der „hinter“ dem NAT steckt, kann eine Verbindung initialisieren.

einer Evaluation hat sich gezeigt, dass Routingprotokolle häufig in den Mobilfunknetzen nicht zugelassen werden und nur über VPN-Tunnel durchgeführt werden könnten. Da dies die Umgebung jedoch unnötig verkompliziert und zusätzliche Faktoren mit einbezieht, die das Ausfallrisiko erhöhen und damit das Ergebnis verfälschen könnten, hat sich das Projektteam für ICMP-Pings zur Feststellung der Verbindungsfunktionalität entschieden. Ein möglicher Ansatz für das Umschalten des Default Gateway ist in Abbildung 4.3 auf Seite 50 gezeigt.

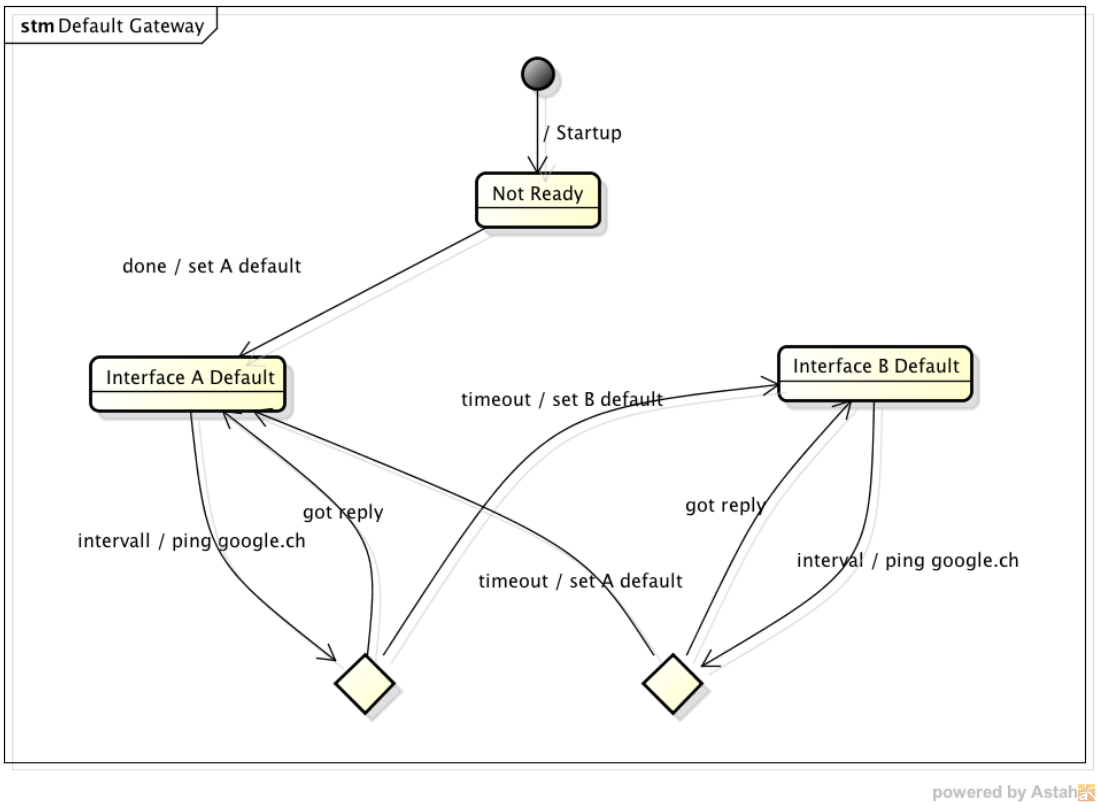

Abbildung 4.3.: Default Gateway

Coupled Congestion Controll für MPTCP

Der RFC zu MPTCP greift in [5, Abs. 3.3.7] das Problem von geteilten Engpässen auf. Verschiedene MPTCP-Verbindung können geteilte Ressourcen nutzen, wobei zwei Zugriffe sich gegenseitig ausbremsen können. Als Lösung für dieses Problem wird der

Gebrauch eines neuen Algorithmus vorgeschlagen. Dieser ist in einem weiteren RFC [6] beschrieben.

MPTCP erkennt über diesen Algorithmus, wie ausgelastet ein Link ist, und verschiebt die Last entsprechend zwischen verfügbaren Links, wobei es einfach versucht, so viel wie möglich zu versenden. Versendet ein Link keine Daten mehr, wodurch keine ACKs mehr ankommen, versendet MPTCP die Daten erneut über einen anderen Link. In einem Test des Projektteams in der Analysephase wurde während eines Transfers der Datenverkehr auf einem der Interfaces blockiert. MPTCP leitete die Daten über das andere Interface um und versuchte weiterhin, Daten über das blockierte Interface zu versenden. Sobald das Interface deblockiert wurde, transferierte MPTCP wieder Daten darüber.

4.2.6. Konfiguration von MPTCP

Für ein funktionierendes MPTCP nach Université catholique de Louvain werden drei Elemente benötigt:

MPTCP-Kernel: Der Kernel erlaubt es, dass Verbindungen mit MPTCP gestartet werden können.

Path Manager: Es können verschiedene Path Manager verwendet werden, für *MPTCP-WiFi* relevant ist der Full-Mesh Path Manager. Dieser verteilt alle seine Daten über alle verfügbaren Links.

Konfiguration: Zuletzt ist noch die Konfiguration des IP-Systems für MPTCP anzupassen. Diese Konfiguration ist abhängig von der Anzahl an Interfaces und deren IP-Einstellungen. Verändern sich diese Einstellungen, muss die MPTCP-Konfiguration angepasst werden.

4.2.7. MPTCP Redundanz und Stabilität

In einem Test des Projektteams in der Analysephase wurden zwei Server miteinander über zwei Switches parallel verbunden. Der Aufbau ist in Abbildung 4.4 dargestellt. Mithilfe des Performanz-Messprogramms *Iperf* wurde eine MPTCP-Verbindung initiiert.

MPTCP erweiterte den Datentransfer selbstständig um die zweite Verbindung.

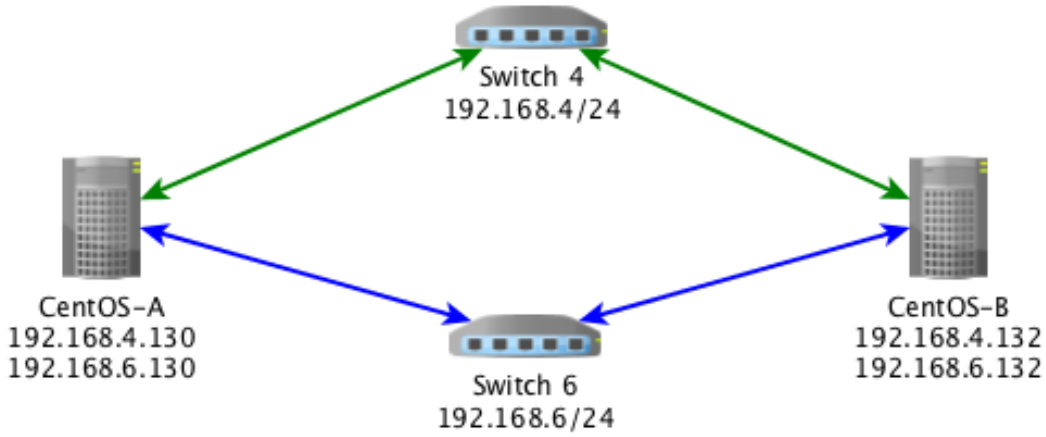

Abbildung 4.4.: MPTCP Stability Test

Wie in Abbildung 4.5 zu sehen ist, wurde die zur Verfügung stehende Bandbreite der Gigabit-Verbindung verdoppelt, womit sich die Performanz-Ziele von MPTCP verifizieren lassen.

```
[root@MPTCP-132 sai]# iperf -t 600 -c 192.168.4.130
------------------------------------------------------------
Client connecting to 192.168.4.130, TCP port 5001
TCP window size: 45.1 KByte (default)
------------------------------------------------------------
[  3] local 192.168.4.132 port 42865 connected with 192.168.4.130 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-600.0 sec    125 GBytes  1.79 Gbits/sec
```

Abbildung 4.5.: MPTCP Speed Test

In diesem Test wurden über eine Zeitspanne von 10 Minuten 125 GB übertragen. Wärend eines separaten Transfers wurde eines der beiden Kabel an CentOS-A aus- und wieder eingesteckt. Der Transfer-Speed reduzierte sich kurzeitig und erholte sich wieder nach dem Einstecken. Dies wurde mit allen Kabelverbindungen nacheinander wiederholt. Die Verbindung blieb bestehen. Erst als beide Kabel an einem Server ausgesteckt wurden, fiel die Verbindung aus. Die durch diesen Test gezeigte Redundanz lässt MPTCP für das Produkt *MPTCP-WiFi* besonders geeignet erscheinen.

4.2.8. MPTCP für Produkt *MPTCP-WiFi*

MPTCP in seiner jetzigen Form ist geeignet für technisch versierte Menschen, die diese Einstellungen selbst vornehmen können. Daher ist es in der aktuellen Gestaltung nicht für *MPTCP-WiFi* geeignet. Es werden zusätzliche Softwarekomponenten benötigt, welche die Interfaces überwachen und bei Veränderungen die Einstellungen anpassen, da MPTCP nicht von wechselnden Interface-Zuständen ausgeht.

MPTCP bietet das dynamische Umleiten von Daten und dadurch eine robustere Verbindung als bei konventionellem TCP. MPTCP kann sich von „unverlässlichen“[21] Interfaces erholen, sobald das Interface wieder verlässlicher ist.

4.3. GSM Modems

GSM-Modems erlauben anhand von Mobilfunkantennen den Zugriff aufs Internet. Heute weitverbreitet ist der Anschluss über eine USB-Schnittstelle. Bei einigen Business-Laptops ist das Modem bereits eingebaut.

4.3.1. *4G Systems XS-Stick W100*

Für die in der vorliegenden Arbeit beschriebene Durchführung hat das Projektteam Zugriff auf zwei *4G Systems XS-Stick W100* der Firma 4G Systems. Diese 4G-USB-Sticks können auf LTE-Netzwerke, beispielsweise die von Swisscom und Orange, zugreifen. Hierdurch können die Datenverbindungen auf eigener Hardware simuliert werden. Ein Beispielbild ist in Abbildung 4.6 auf Seite 54 zu sehen.

4.3.2. Storage/Modem-Modus Problematik

Beim Einstecken der Modems in einen Laptop verfügen sie über zwei verschiedene Modi. Im ersten Modus wird ein Datenträger mit den jeweiligen Treibern und ihrem eigenen Modem Manager *XSManager* exponiert, siehe Abbildung 4.7. Im zweiten Modus kann

[21] „unverlässlich“ wird hier definiert als unregelmässiges Interface bei Durchsatz, Jitter, Laufzeit usw.

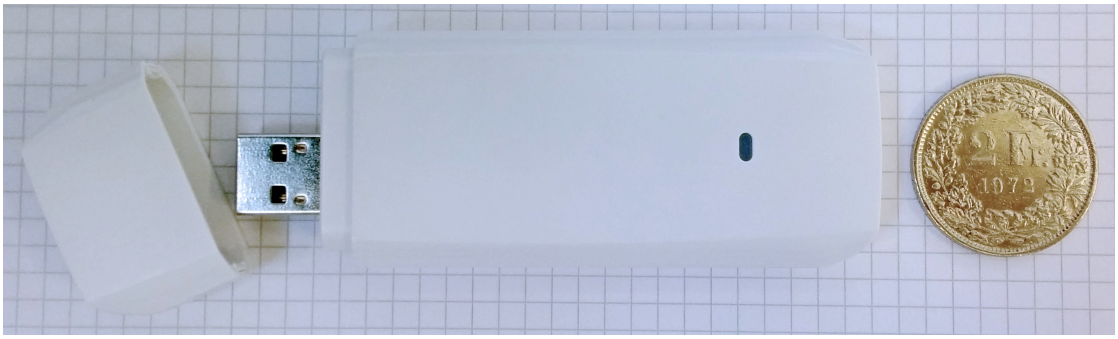

Abbildung 4.6.: GSM-Modem

auf das eigentliche Modem zugegriffen werden. Wie dieses Umschalten geschieht, ist weder auf dem Datenträger noch auf der Webseite `http://4g-systems.com/produkte/xsstick/xsstickrw100.html` des Herstellers dokumentiert. Der Modem-Modus ist jedoch notwendig für die Datenübertragung.

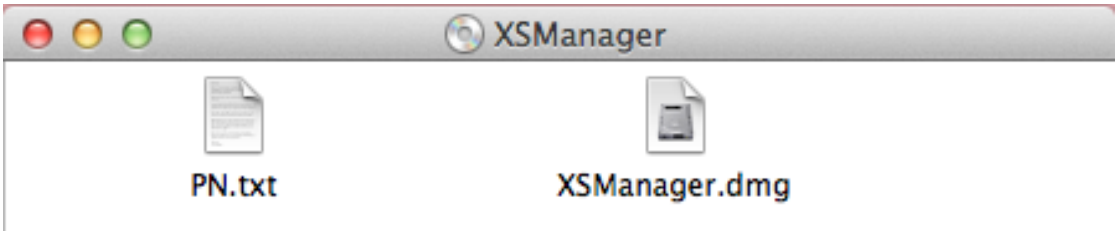

Abbildung 4.7.: XS-Manager

Wird der *4G Systems XS-Stick W100* direkt an einer virtuellen Maschine angeschlossen, kann nicht auf das Modem zugegriffen werden. Mit dem Befehl „lsusb“ können die angeschlossenen USB-Geräte angezeigt werden. Abhängig vom Modus, in dem sich der *4G Systems XS-Stick W100* befindet, verändert sich auch die USB-ID. Abbildung 4.8 zeigt die Ausgabe von „lsusb“, wenn sich *4G Systems XS-Stick W100* (T & A Mobile Phones) im Datenträger-Modus befindet. Die ID im Datenträger-Modus ist „1bbb:f000“.

Abbildung 4.9 zeigt die Ausgabe von „lsusb“, wenn sich *4G Systems XS-Stick W100* (T & A Mobile Phones) im Modem-Modus befindet. Die ID im Modem-Modus ist „1bbb:011e“.

```
cruiser@mptcp:~$ lsusb
Bus 001 Device 007: ID 1bbb:f000 T & A Mobile Phones
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 007: ID 0e0f:0008 VMware, Inc.
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Abbildung 4.8.: Storage Modus

```
cruiser@mptcp:~$ lsusb
Bus 001 Device 006: ID 1bbb:011e T & A Mobile Phones
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 007: ID 0e0f:0008 VMware, Inc.
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Abbildung 4.9.: Modem Modus

4.3.3. Modem Modus Workaround – VMware Fusion

Im Regelbetrieb würde man den XSManager vom Storage Modus installieren und der XSManager stellt den *4G Systems XS-Stick W100* in den Modem Modus. Für Linux bietet der Hersteller jedoch keine Version des XSManagers an.

Ein Workaround, um den *4G Systems XS-Stick W100* in den Modem-Modus innerhalb einer virtuellen Maschine in VMware Fusion zu wechseln, besteht darin, den Stick erst dem Host (OS X) zuzuweisen und anschliessend der VM[22] zu übertragen. Für eine Schritt-für-Schritt-Beschreibung des Workaround siehe Abschnitt 7.3.7 auf Seite 113. Das Modem bleibt im Modem-Modus, wenn die VM neugestartet wird. Es wechselt jedoch in den Datenträger-Modus, wenn es physisch vom Host (OS X) getrennt wird oder der Host neustartet.

[22] Virtuelle Maschine

4.3.4. GSM-Abonnemente

Von der HSR wurden für das Projektteam zwei SIM-Karten gestellt. Mit diesen werden die zwei Modems betrieben.

Swisscom: Swisscom Data Business Start - 500 MB/Monat

Orange: Orange Prepaid - Gratis 20 MB/Tag

Für die Construction-Phase wurde zugesagt, dass das Datenvolumen der Orange-SIM-Karte bei Bedarf auf 1 GB/Monat erhöht wird.

Besonders bei diesen limitierten Abonnementen ist klar ersichtlich, weshalb ein Bandbreiten-Detektierverfahren funktionieren muss, ohne das Datenvolumen beträchtlich zu beanspruchen, damit genug Datenvolumen für Nutzdaten verfügbar bleibt.

4.3.5. Modem Manager

Nach der Vorarbeit [1] kann für zusätzliches Auslesen von Informationen das Programm *Modem Manager* verwendet werden. So können beispielsweise Werte wie Empfangsstärke ausgelesen werden. Da für die vorliegende Arbeit statt der absolute Empfangsstärke der daraus folgende Durchsatz relevant ist, wurde dies nicht weiter untersucht.

4.4. *IPmotion CAR-A-WAN.coach Plus Vorserie*

Dem Projektteam wurde von der CloudGuard Software AG die Hardware *IPmotion CAR-A-WAN.coach Plus Vorserie* übergeben. Dieses Gerät ist eine Vorabversion der Weiterentwicklung derjenigen Geräte, die heute in den Fahrzeugen in Betrieb sind. Ziel der Hardware ist es, ein lokales Netzwerk im Fahrzeug bereitzustellen, das über Mobilfunk mit dem Internet verbunden ist.

Die Dokumentation des Geräts kann unter [9] gefunden werden.

4.4.1. Interfaces

IPmotion CAR-A-WAN.coach Plus Vorserie bietet folgende Schnittstellen, wie in Abbildung 4.10 und Abbildung 4.11 auf Seite 57 gezeigt:

- LAN (RJ45[23], 100 MBit/s)
- 2× Anschluss für Mobilfunkantenne (UMTS und LTE)
- 2× SIM-Karten-Slot
- 2× USB 2.0
- 12-V-/24-V-Stromanschluss
- CAN[24]-Anschluss
- Anschluss für GPS-Antenne
- 2× Anschluss für WiFi-Antenne

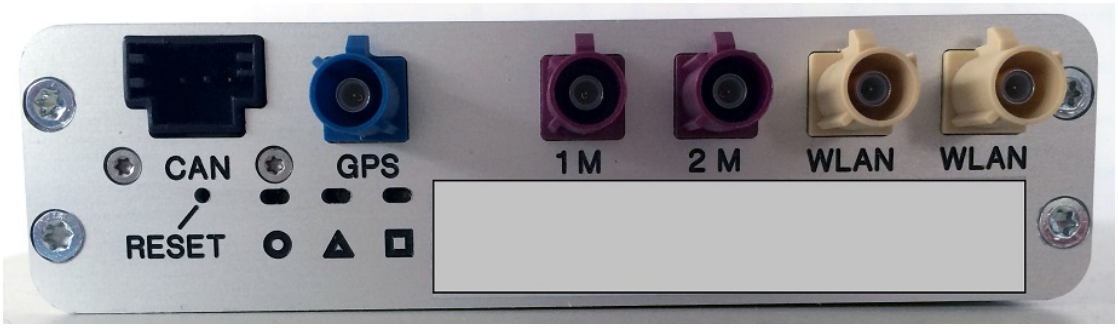

Abbildung 4.10.: IPmotion hinten

Zudem zeigen fünf LEDs verschiedene Zustände an. Drei LEDs – beschriftet mit Kreis, Dreieck und Quadrat – zeigen den Zustand der Mobilfunkverbindung und den Gerätezustand. Neben dem RJ45-Port finden sich zwei weitere LEDs, welche den Zustand der Ethernet-Verbindung und laufende Übertragungen anzeigen.

[23] Registered Jack 45 (Stecker für Ethernet)
[24] Controller Area Network

Abbildung 4.11.: IPmotion vorne

4.4.2. Antennen

Die Antennen-Stecker sind farblich passend zu den jeweiligen Anschlüssen am Gerät gekennzeichnet. Die Kabel verfügen über eine Länge von jeweils 3 Metern. Um zu vermeiden, dass sich die Antennen gegenseitig stören, sollten sie ca. 2 Meter voneinander entfernt aufgestellt werden. Die Mobilfunk-Antenne ist in Abbildung 4.12 auf Seite 59 und die WiFi-Antenne in Abbildung 4.13 auf Seite 60 agebildet.

4.4.3. Konfiguration und Software

Der Router hat standardmässig die IP-Adresse 10.10.10.1. SSH[25] ist standardmässig deaktiviert, kann jedoch temporär über die mitgelieferte Software aktiviert werden.

Über diese IP-Adresse kann über HTTP[26] Port 8888 eine Web-Administrationsplattform erreicht werden, welche grundlegende Konfigurationsmöglichkeiten erlaubt.

Es existiert eine Monitoring-Software *cawmonitor.jar*, welche vom Gerät über HTTP Port 8888 heruntergeladen werden kann. Mit diesem Programm können verschiedene Informationen, beispielsweise IP-Adressen und Durchsatz, abgerufen werden. Zudem können Konfigurationen, wie Aktivierungen von Verbindungen, Neustart des Geräts oder temporäre Aktivierung von SSH, vorgenommen werden.

[25] Secure Shell
[26] Hypertext Transfer Protocol

Abbildung 4.12.: GSM-Antenne

Da Benutzername und Passwort des Gerätes dem Projektteam während der Analyse-Phase nicht zur Verfügung standen, kann eine genauere Analyse über Konfiguration und Software erst in der Construction-Phase stattfinden.

4.4.4. Hardwareanalyse

Dem Projektteam waren bis zum Abschluss der Analyse die Passwörter des bereitgestellten *IPmotion CAR-A-WAN.coach Plus Vorserie* nicht bekannt. Daher muss eine Hardwareanalyse in der Construction-Phase stattfinden.

Abbildung 4.13.: WiFi-Antenne

4.5. Anpassbarkeit der MPTCP-Implementation

In diesem Kapitel wird die Anpassbarkeit der MPTCP-Implementation der Université catholique de Louvain behandelt. Diese Open-Source-Implementation[27] stellt einen erweiterten Linux-Kernel bereit. Direkt im Kernel sind die entsprechenden Erweiterungen umgesetzt, welche für MPTCP nötig sind.

4.5.1. Kernel-Modul

Tiefgründige Anpassungen lassen sich im Kernel selbst vornehmen. Da ganz MPTCP in diesem Bereich implementiert ist, lassen sich so beliebige Änderungen durchführen.

Dadurch müssen jedoch Nachteile in Kauf genommen werden:

- Sehr hohe Einarbeitungszeit in den Code (mehrere 10 000 Zeilen nahezu undokumentierter C-Code)

[27] Git-Repository: `https://github.com/multipath-tcp/mptcp`, Abgerufen am 21. März 2014

- Sehr hohe Testdurchführungszeit (Kompilierungsvorgang Kernel: mind. 1.5 Stunden)
- Kompatibilität zur ursprünglichen Implementation wird gebrochen (Aktualisierungen, beispielsweise Sicherheits-Fixes, werden sehr aufwendig)
- Besonderes Augenmerk muss darauf gelegt werden, die MPTCP-Spezifikation trotz Änderungen weiterhin genau einzuhalten

4.5.2. User-Space-Programm

Eingreifen in kleinem Rahmen ist in der Multipath-Implementation der Université catholique de Louvain auch von ausserhalb des Kernels möglich. Dazu werden verschiedene Schnittstellen bereitgestellt.

Da Kernelanpassungen in der zur Verfügung stehenden Zeit nicht realistisch sind, werden die abweichenden Anforderungen des Projekts gegenüber der MPTCP-Standardimplementation über ein oder mehrere User-Space-Programme gelöst. Die Anforderungen an dieses Programm[28] sind im Kapitel 2 auf Seite 16 festgehalten.

Die detaillierte Umsetzung des oder der User-Space-Programme wird in der Construction-Phase als Teil der Programmdokumentation festgehalten und besprochen.

4.5.3. Routing

Routing definiert auf OSI-Layer 3, auf welche Weise Daten von einem Netzwerk in ein anderes geleitet werden können. Auf einem lokalen Router mit statischem Routing werden die direkt zur Verfügung stehenden Pfade in einer Routing-Tabelle definiert. An den Default-Pfad werden alle Netzwerkpakete weitergegeben, deren genauer Pfad unbekannt ist. Höher gelegene Router in der Hierarchie werden meist über dynamische Routingprotokolle gesteuert. Diese dynamischen Protokolle sind im Gegensatz zu statischem Routing auf den Ausfall von Pfaden ausgelegt. Client-Systeme, welche über ihren Default Gateway auf einen statischen, lokalen Router zugreifen, haben keine Funktionalität, um bei Ausfall ein anderen Router zu verwenden. Um diesem Umstand Abhilfe zu schaffen, können

[28] oder diese Sammlung von Programmen

Protokolle wie VRRP[29], CARP[30] oder HSRP[31] verwendet werden. Dabei können zwei Master-/Slave-Router auf die gleiche virtuelle IP-Adresse antworten. Sollte der Master ausfallen, kann der Slave antworten. Der Client bemerkt dabei keinen Unterschied.

Vor- und Nachteile

Es existieren verschiedene Ansätze, um mehrere Pfade per Routing zu nutzen. Ein Ansatz ist es, per Lastenverteilung ausgehende Verbindungen auf die zur Verfügung stehenden mit dem Internet verbundenen Interfaces zu verteilen. Der Linux Kernel[32] bietet schon solch eine Funktionalität mittels *IPtables*. Es muss darauf geachtet werden, dass die Antwort auf eine Anfrage über das gleiche Interface wieder nach aussen geht, auf welchem die zugehörige Anfrage eingegangen ist.

Rein routing-basierte Lösungen ohne MPTCP haben den Nachteil, dass bei Verlust eines Pfades die zugewiesenen TCP-Verbindungen nicht auf einem anderen Interface ohne Unterbruch aufrecht erhalten werden können, da die IP-Adresse eine andere ist. Sie müssen neu aufgebaut werden. Beispiel: eine Datenübertragung über zwei Interfaces A und B die über zwei unterschiedliche Internet-Provider angeschlossen sind. Wenn Interface B ausfällt und Interface A Pakete im Namen von B versendet, würden die Pakete beim Ziel ankommen. Die Antwort würde jedoch dorthin zurück gesendet werden, wo B „gemeldet“ ist, was nicht bei A ist. Somit würde die Antwort nicht ankommen. Im Gegensatz dazu meldet MPTCP seine zur Verfügung stehenden Interfaces seinem Partner und kann somit bei einem Ausfall von B die Übertragung mit A fortführen.

Durch zwei VPN-Verbindungen könnte ein dynamisches Routingprotokoll wie OSPF[33] übermittelt werden und Routingpfade zwischen Seite A und B aushandeln. Beide sind über Tunnel A und Tunnel B verbunden. Würde wie im vorgegangenem Beispiel Interface B ausfallen, dann würden somit auch über Tunnel B keine Pakete ankommen. Die Hello-Pakete kämen nicht mehr an und der Traffic würde über die zweite VPN-Verbindung geleitet werden. Die TCP-Verbindung würde nicht abbrechen.

[29] Virtual Router Redundancy Protocol
[30] Common Address Redundancy Protocol
[31] Host Standby Router Protocol
[32] `http://lartc.org/howto/lartc.rpdb.multiple-links.html`, Abgerufen am 05. April 2014
[33] Open Shortest Path First

Routing vs. MPTCP Fazit

Für sich bewegende Fahrzeugen können sich die verfügbaren Verbindungen schnell ändern und MPTCP würde sehr gut funktionieren, da es besonders im Hinblick auf Mobilität entwickelt wurde [8]. In einem Vortrag der Université catholique de Louvain [7] ist die Umschaltgeschwindigkeit von MPTCP demonstriert.

Inwiefern dynamisches Routing besser geeignet ist, könnte in einer Folgearbeit untersucht werden.

4.5.4. VPN

VPN erlaubt es, Netzwerke miteinander zu verbinden. Diese Verbindung kann auf verschiedenen OSI-Layern geschehen. Bei dedizierten VPN-Geräten, wie z. B. einer VPN-Firewall, werden häufig Netzwerke aus Routing-Sicht miteinander verbunden. Ein Beispiel für solch ein Szenario ist ein Firmennetzwerk. Der Hauptsitz nutzt den Range 10.0.0.0/16 und vergibt 10.1.x.0/24 Netzwerke an die verteilten Niederlassungen. Obwohl diese Netzwerke nicht direkt miteinander verbunden sind, kann über das Internet ein VPN aufgebaut werden. Das VPN stellt hier eine virtuelle Verbindung her und erlaubt das Routen von Paketen von der Niederlassung zum Hauptsitz und umgekehrt. Die VPN-Nutzung kann zur Lösung unterschiedlicher Probleme dienen:

Szenario: Road Warrior Mit „Road Warrior“ wird ein Mitarbeiter einer Organisation bezeichnet, der sich von aussen Zugriff auf die Organisation verschaffen möchte, um auf Ressourcen der Organisation zuzugreifen, beispielsweise das Intranet, E-Mails oder eine Datenablage. Dabei kann auch entschieden werden, ob jeglicher Verkehr über das VPN geleitet werden soll.

Szenario: Office Branch Der Datenverkehr einer Geschäftsaussenstelle soll am Hauptstandort konzentriert werden. Mit einem VPN kann jeglicher Datenverkehr zum Hauptstandort geleitet werden.

Szenario: Mesh Es können auch Zweigstellen untereinander verbunden werden. Falls es Verkehr unter den Zweigstellen gibt, muss dieser nicht über einen zentralen Ort geleitet werden.

VPN ist vielseitig, viele seiner Anwendungsgebiete können jedoch auch über Routing realisiert werden. Ein wichtiger Grund für den Einsatz von VPN ist die Authentifizierung und Autorisierung von Zugängen in ein internes Netz einer Organisation. So kann der Zugang von aussen selektiv nach innen erlaubt werden. Ein Bonus ist die Möglichkeit zur Verschlüsselung des Datenverkehrs.

OSI-Layer des VPN

VPN kann auf verschiedenen Ebenen des OSI-Modells umgesetzt werden. Im Rahmen des Produkts *MPTCP-WiFi*, dessen Umsetzung in dieser Bachelorarbeit beschrieben wird, sind folgende Varianten relevant:

Layer 3: Auf Layer 3[34] arbeitet z. B. *IPsec*. IPsec ist ein performantes und flexibles Protokoll, das sich für grosse Datenmengen empfehlen lässt. Es ist jedoch zu beachten, dass durch die IP-Adress-Knappheit bei IPv4 sich NAT bei vielen Internetanschlüssen durchgesetzt hat. Durch NAT besteht kein direkter Zugriff auf die IP-Adressen eines anderen Netzwerks. Daher muss für das NAT-Gerät unbedingt VPN-Passthrough auf Client-Seite für das jeweilige Layer-3-VPN-Protokoll aktiviert werden. Da diese Option über das Mobilfunk- und WiFi-Netz nicht zuverlässig umgesetzt werden kann, stellt ein OSI-Layer-3-VPN keine Option dar.

Layer 4: Auf Layer 4, dem Transportlayer, arbeitet z. B. *OpenVPN*. OpenVPN verbindet sich entweder über UDP oder über TCP. Auf Layer 4 müssen im Zusammenhang mit NAT keine weiteren Einstellungen getroffen werden, wodurch es auch im Mobilfunk-Umfeld keine weiteren Schwierigkeiten bereitet. Daher ist OSI-Layer-4-VPN für *MPTCP-WiFi* geeignet.

VPN Keepalive OpenVPN

Für *MPTCP-WiFi* müssen die genutzten Dienste und Interfaces überwacht werden. OpenVPN bietet eine Einstellung („Keep Alive“)[35], über welches es selbst kontrolliert,

[34] dem Netzwerklayer

[35] `https://openvpn.net/index.php/open-source/documentation/howto.html`, Abgerufen am 05. April 2014

ob eine Verbindung noch besteht. In ping-ähnlichen Paketen wird überprüft, ob das Gegenüber noch erreicht werden kann. Ein zweiter Parameter („Retry“) gibt an, wie oft dies bei Misslingen wiederholt werden soll. Als Nebeneffekt werden Verbindungen aufrecht gehalten, was NAT-Timeouts für die Portzuweisungen verhindert.

Für *MPTCP-WiFi* soll ausgetestet werden, welche Kombination an Werten das beste Ergebnis erzielt. Hierbei muss auf eine gute Balance von Konvergenz und Stabilität geachtet werden.

Nutzen für *MPTCP-WiFi*

Für *MPTCP-WiFi* muss TCP als Transportprotokoll verwendet werden, um die Funktionalität von MPTCP nutzen zu können. OpenVPN kann TCP als Transport-Protokoll verwenden und ist aus diesem Grund für *MPTCP-WiFi* geeignet. Die Verwendung von TCP muss in der Konfigurationsdatei angegeben werden, da OpenVPN per Default UDP verwendet.

Die VPN-Verbindung kann für jeglichen Nicht-TCP-Traffic verwendet werden. Aufgrund der TCP-over-TCP-Problematik wird TCP über den SOCKS-Proxy geleitet. Das VPN nimmt somit Nicht-TCP-Traffic an und packt diesen in MPTCP-Pakete ein. Das System kann nach dem in Abbildung 4.14 auf Seite 65 gezeigten Schema aufgebaut werden.

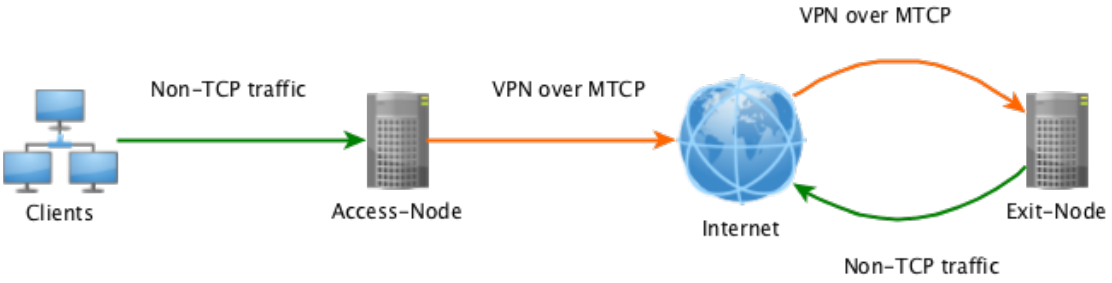

Abbildung 4.14.: MPTCP VPN

4.5.5. Proxy

Ein Proxy kann auf OSI-Layer 7[36] Protokoll-Anfragen bearbeiten, zwischenspeichern und weiterleiten. Im Gegensatz zu anderen Lösungen, wie VPN, muss der Proxy gezielt für ein bestimmtes anderes Layer-7-Protokoll entwickelt worden sein. Proxys werden oft in einer Webumgebung eingesetzt. Sie erlauben es z. B. einem Unternehmen, den direkten Routing-Weg nach aussen zu blockieren. So hat ausschliesslich der Proxy-Server Zugriff auf das Internet.

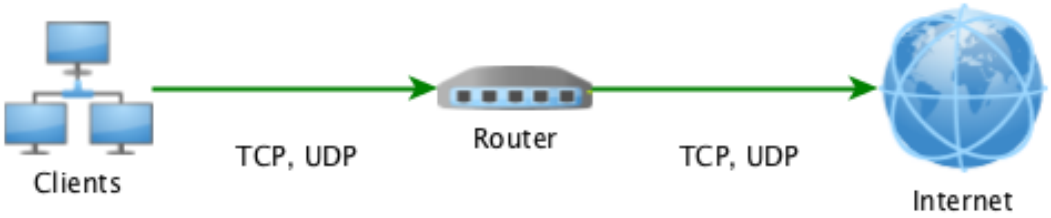

Abbildung 4.15.: Reines Routing

Abbildung 4.15 zeigt, wie bei einem reinen Routing die Clients direkt aufs Internet zugreifen können. Unter Umständen steht noch eine Firewall dazwischen, es kann aber ansonsten frei zugegriffen werden.

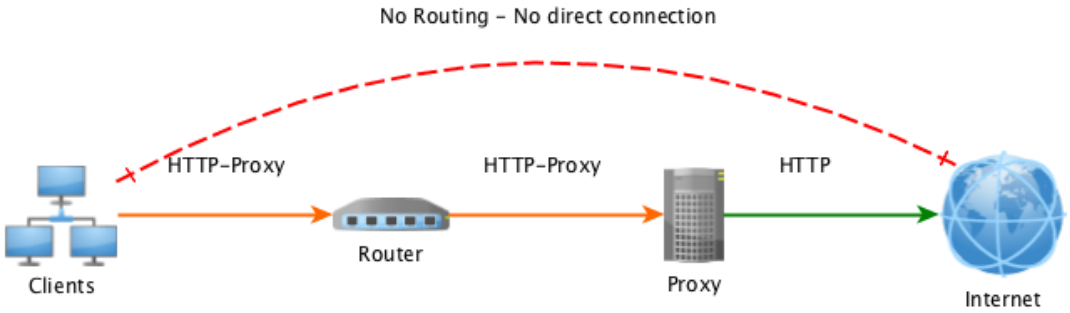

Abbildung 4.16.: Verbindung über HTTP-Proxy

Wie in Abbildung 4.16 gezeigt, kann mit einem Proxy nicht mehr direkt von einem Client

[36] der Applikations-Ebene

auf das Internet zugegriffen werden.

Transparent Proxy

Üblicherweise muss ein Proxy in die Applikation eingetragen werden, die ihn nutzen soll. Bei *MPTCP-WiFi* ist dies jedoch nicht möglich, da die Geräte nicht von vornherein bekannt sind. Stattdessen kann ein Transparent Proxy eingesetzt werden. Der Transparent Proxy leitet z. B. HTTP-Anfragen auf einen bestehenden Proxy um. Dieser Proxy bearbeitet die Anfrage und sie wird transparent wieder zurückgeleitet. So ist zwar keine direkte Authentifizierung möglich, was im Falle von *MPTCP-WiFi* jedoch nicht nötig ist, bzw. an anderer Stelle vorgenommen werden kann.

Proxy Chaining

Beim Proxy-Chaining kann zwischen zwei Standorten eine Weiterleitung von Proxy-Anfragen erreicht werden. Oft ist dies nicht nötig, da die Clients direkt auf den letzten Proxy zugreifen können. Ist dies nicht der Fall, kann ein Proxy Chaining Abhilfe schaffen.

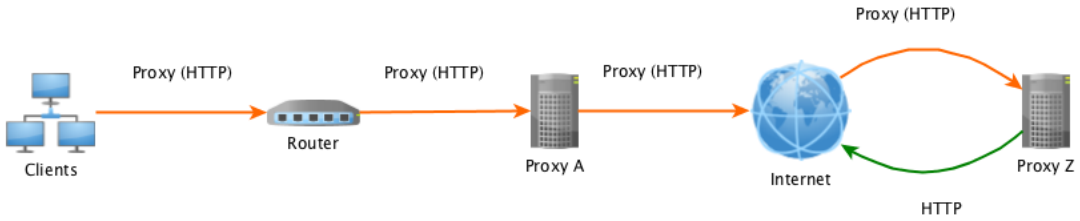

Abbildung 4.17.: Proxy Chaining

Abbildung 4.17 zeigt, wie Proxy-Anfragen von einem Proxy and den nächsten weitergereicht werden. Im Falle von *MPTCP-WiFi* kann für diese Zwischenverbindungen MPTCP eingesetzt werden.

Caching und Logging

Proxys leiten die Anfragen nicht nur blind weiter, sondern können auch Anfragen anpassen oder zwischenspeichern, wie es z. B. bei einem HTTP-Proxy der Fall ist. Diese Anpassung kann aus Privacy-Gründen geschehen[37] oder mit dem Ziel, Anfragen zu blockieren. Proxys können auch Anfragen protokollieren. Beim Zwischenspeichern werden wiederkehrende Anfragen von einem Cache geladen. Bei einer wiederholten Anfrage kann die Antwort direkt beantwortet werden, ohne sie vom Ziel-Server wieder ganz abfragen zu müssen. Hierfür wird geprüft, ob die gecachte Version noch aktuell ist.

Im Szenario von *MPTCP-WiFi* könnten so die Datenbelastung durch Caching reduziert werden und die empfundene Geschwindigkeit durch geringere Latenz verbessert werden.

HTTP-Proxy mit HTTPS

Da ein HTTP-Proxy die HTTP-Verbindungen unterbricht, sind HTTPS[38]-Verbindungen nicht mit dem ursprünglichen Zertifikat möglich. Um dieses Problem zu lösen, kann das Zertifikat des *MPTCP-WiFi*-Betreibers in die Endgeräte eingetragen werden. Dies ist bei fremden Geräten nicht automatisch möglich. HTTP bietet stattdessen den Befehl HTTP Connect[39] an, womit die Verbindung weitergeleitet werden kann, ohne eine HTTPS-Verbindung zu unterbrechen, wobei der Proxy und jegliche Kontrollen umgangen werden.

SOCKS und Redsocks

Ein SOCKS-Proxy ist ein Proxy, der für TCP-Verbindungen geschaffen wurde. Der OpenSSH-Client [11] bietet beispielsweise einen eingebauten SOCKS-Proxy an, bei dem TCP-Verbindungen über eine bestehende SSH-Verbindung geleitet werden können. Anwendungen, die diesen Proxy nutzen sollen, müssen jedoch mit SOCKS kompatibel sein.

[37] Beispielsweise kann die Entropie der Anfrage reduziert werden, in dem z. B. der User-Agent auf ein generischen Wert geändert wird.
[38] Hypertext Transfer Protocol Secure
[39] `http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html`

Um das Problem der Kompatibilität zu umgehen, kann ein Wrapper wie Redsocks verwendet werden. Diese Programme leiten den gesamten TCP-Verkehr über einen SOCKS-Proxy um, ohne dass die Applikationen daran angepasst sein müssen. Die Möglichkeit der Verwendung von Redsocks wurde auch in der Vorarbeit [1] vorgeschlagen.

Nutzen für *MPTCP-WiFi*

Als eine allgemeine Lösung für *MPTCP-WiFi* stellt es sich vorerst als sinnvoll dar, einen allgemeinen TCP-Proxy zu nutzen, der HTTP sowie weitere TCP-basierte Protokolle weiterleiten kann. In einem weiteren Schritt könnte mit einem HTTP- bzw. HTTPS-Proxy Verkehr gesondert behandelt und beschleunigt werden.

Ein Produkt wie Redsocks kann Verbindungen, die nicht direkt mit SOCKS kompatibel sind, weiterleiten. Beim Access Node wird der TCP-Verkehr von Nicht-TCP-Verkehr getrennt und über den Proxy geleitet. Dies verhindert die Problematik des TCP-over-TCP.

4.6. Integrationstests

Integrationstests werden durchgeführt, um die Zusammenarbeit der Einzelkomponenten zu bewerten. Auf diese Weise kann ein funktionsfähiges Gesamtsystem erzeugt werden, welches alle Anforderungen erfüllt.

Die Testfälle werden aus den Use Cases abgeleitet. Als Anregung zur Umsetzung können die Testcases aus der Vorarbeit [1] dienen, die jedoch an die neuen Bedingungen angepasst werden müssen.

Die Tests werden mithilfe von Prototyp-Hardware auf Strecken und in Fahrzeugen von Jungfraubahnen und Postauto durchgeführt. Bei Bedarf wird ein Auto für zusätzliche Tests verwendet.

5. Domainanalyse

5.1. Bestandteile

Das Produkt *MPTCP-WiFi* enthält zwei Netzwerkknoten.

Access Node: Der Access Node befindet sich innerhalb des Fahrzeuges. Mit die Nutzer sind über WiFi mit ihm verbunden.

Exit Node: Der Exit Node befindet sich ausserhalb des Fahrzeuges. Er fungiert als Proxy, um die Pakete der Nutzer ans Internet sowie die Pakete vom Internet an die Nutzer weiterzuleiten.

In Abbildung 5.1 auf Seite 5.1 wird das Produkt in seinem Umfeld gezeigt: Pakete vom Nutzer an das Internet gehen beim Access Node ein, werden von diesem an den Exit Node geleitet und von diesem an das Ziel im Internet. Auf dem Rückweg passieren die Pakete zuerst den Exit Node, dann den Access Node und gelangen von diesem aus zum Nutzer.

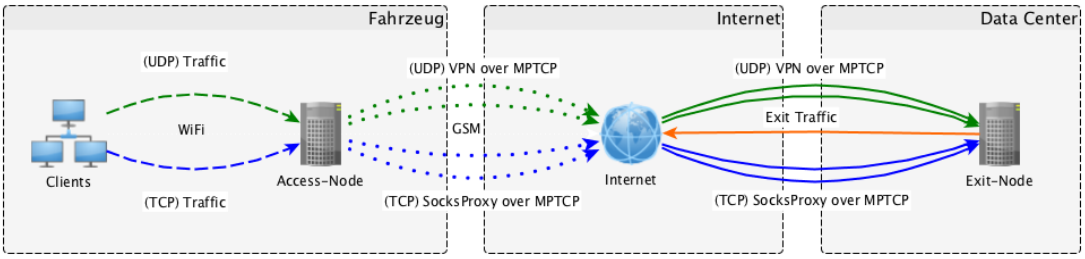

Abbildung 5.1.: Übersicht des Systems

5.2. Umfeld

Das Produkt *MPTCP-WiFi* ist in mehrere Umweltsysteme eingebettet.

Nutzer: Der Nutzer ist eine Person oder ein technisches Gerät. Er beeinflusst das Produkt, indem er TCP- und UDP-Pakete an das Internet schickt und Pakete aus dem Internet erhält.

Funknetze: Access Node und Exit Node sind miteinander über mehrere Funknetze (Mobilfunk und WiFi) verbunden. Auf dieser Strecke wird MPTCP eingesetzt.

Internet-Backbone: Das Internet wird vom Produkt über den Backbone erreicht. Über diesen empfängt es Pakete der Nutzer oder sendet Pakete an die Nutzer.

5.3. Das Produkt im Umfeld

Das Produkt *MPTCP-WiFi* muss auf die Einflüsse der Umweltsysteme reagieren. Diese werden im Folgenden erläutert.

5.3.1. Daten von Nutzern

Wenn ein Nutzer Daten an das Produkt schickt, müssen diese am Access Node bezüglich des Protokolls analysiert werden.

Wird auf OSI-Layer 4 das Protokoll TCP eingesetzt, sind die Pakete bereits MPTCP-fähig. Dort können also die MPTCP-Header ergänzt werden und die Pakete über den SOCKS-Proxy durch die Funknetze an den Exit Node geleitet werden.

Wird TCP nicht eingesetzt, müssen die Pakete über einen TCP-Tunnel geleitet werden, um MPTCP verwenden zu können. Es wird VPN über TCP eingesetzt, um die Nicht-TCP-Pakete über die Mobilfunknetze an den Exit Node zu leiten.

5.3.2. Daten vom Internet

Werden Daten vom Internet an einen Nutzer gesendet, so handelt es sich stets um eine Antwort. Daher kann sie die jeweils noch offenen Kanäle ohne weitere Interaktion des Produkts passieren.

Ein eigener Verbindungsaufbau vom Internet aus zu einem Nutzer ist bereits aus Prinzip nicht möglich, da die NATs in diese Richtung nicht passiert werden können. Daher muss das Produkt diese Richtung des Verbindungsaufbaus auch nicht berücksichtigen.

5.3.3. Änderungen an den Funknetzen

Die Funknetze können sich, da sich das Fahrzeug bewegt, regelmässig ändern. Dabei können mehrere Fälle eintreten:

- Der mögliche Durchsatz über ein Funknetz ändert sich, beispielsweise durch wechselnde Signalstärke.
- Ein Funknetz ist nicht mehr verfügbar, beispielsweise durch ein Funkloch.
- Ein Funknetz ist neu wieder verfügbar.

Wenn ein Funknetz nicht mehr verfügbar ist, kann sich dies über zwei Arten äussern:

- Das Netzwerk-Interface wird automatisch deaktiviert („down“).
- Das Netzwerk-Interface bleibt aktiviert, der Durchsatz sinkt jedoch auf 0.

Auf beide Fälle muss der Access Node reagieren und den Verkehr über die anderen Funknetze leiten.

Ist ein Funknetz wieder verfügbar, kann sich dies ebenfalls über zwei Arten äussern:

- Das Netzwerk-Interface wird wieder automatisch aktiviert („up“).

- Über das weiterhin aktiviert gebliebene Netzwerk-Interface können nun wieder Daten übertragen werden.

Auch hier müssen beide Fälle berücksichtigt werden.

Möglich ist auch der Fall, dass sämtliche Funknetze zeitweise nicht mehr verfügbar sind. Dann muss der Access Node möglichst schnell wieder seinen Betrieb aufnehmen, sobald wieder ein Funknetz verfügbar ist.

5.4. White-Box

5.4.1. Domainmodell

Abbildung 5.2 auf Seite 74 zeigt das Domainmodell.

5.4.2. Bestimmung des Default Interfaces

Damit neue Verbindungen vom Access Node zum Exit Node eröffnet werden können, muss auf dem Access Node immer ein funktionierendes Interface als Default Interface eingestellt sein. Da die verfügbaren Interfaces aufgrund des Umfelds regelmässig wechseln, muss dazu ein Algorithmus implementiert werden.

Dieser Algorithmus ist in Abbildung 5.3 auf Seite 75 dargestellt.

WiFi-Priorisierung

Als Erweiterung der Aufgabenstellung ist in der Anforderungsspezifikation der Vorschlag enthalten, WiFi gegenüber den Mobilfunk-Interfaces zu bevorzugen. In diesem Fall soll immer das WiFi-Interface verwendet werden, sofern es verfügbar ist. Abbildung 5.4 auf Seite 76 zeigt den erweiterten Algorithmus.

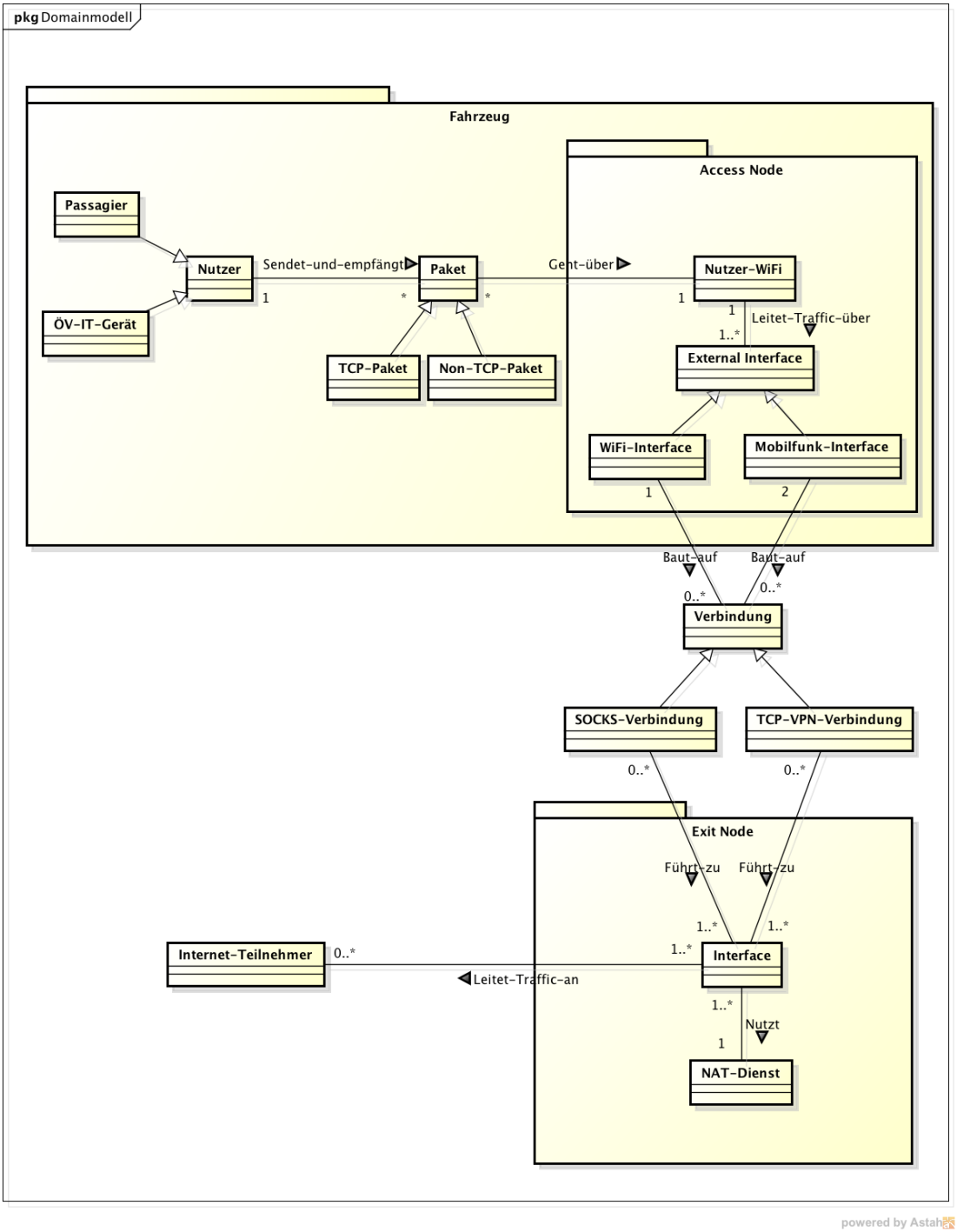

Abbildung 5.2.: Domainmodell

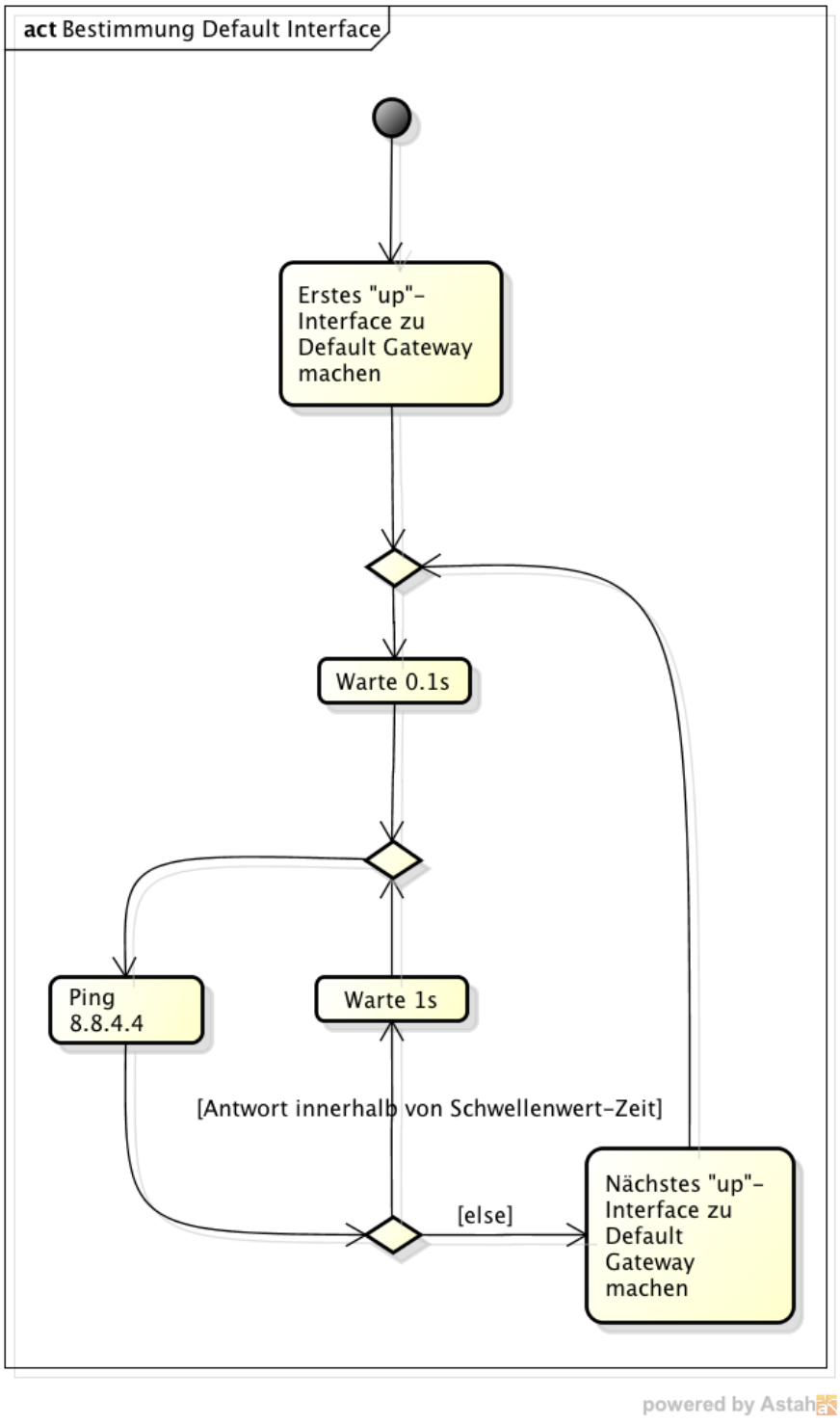

Abbildung 5.3.: Aktionsdiagramm Bestimmung Default Gateway

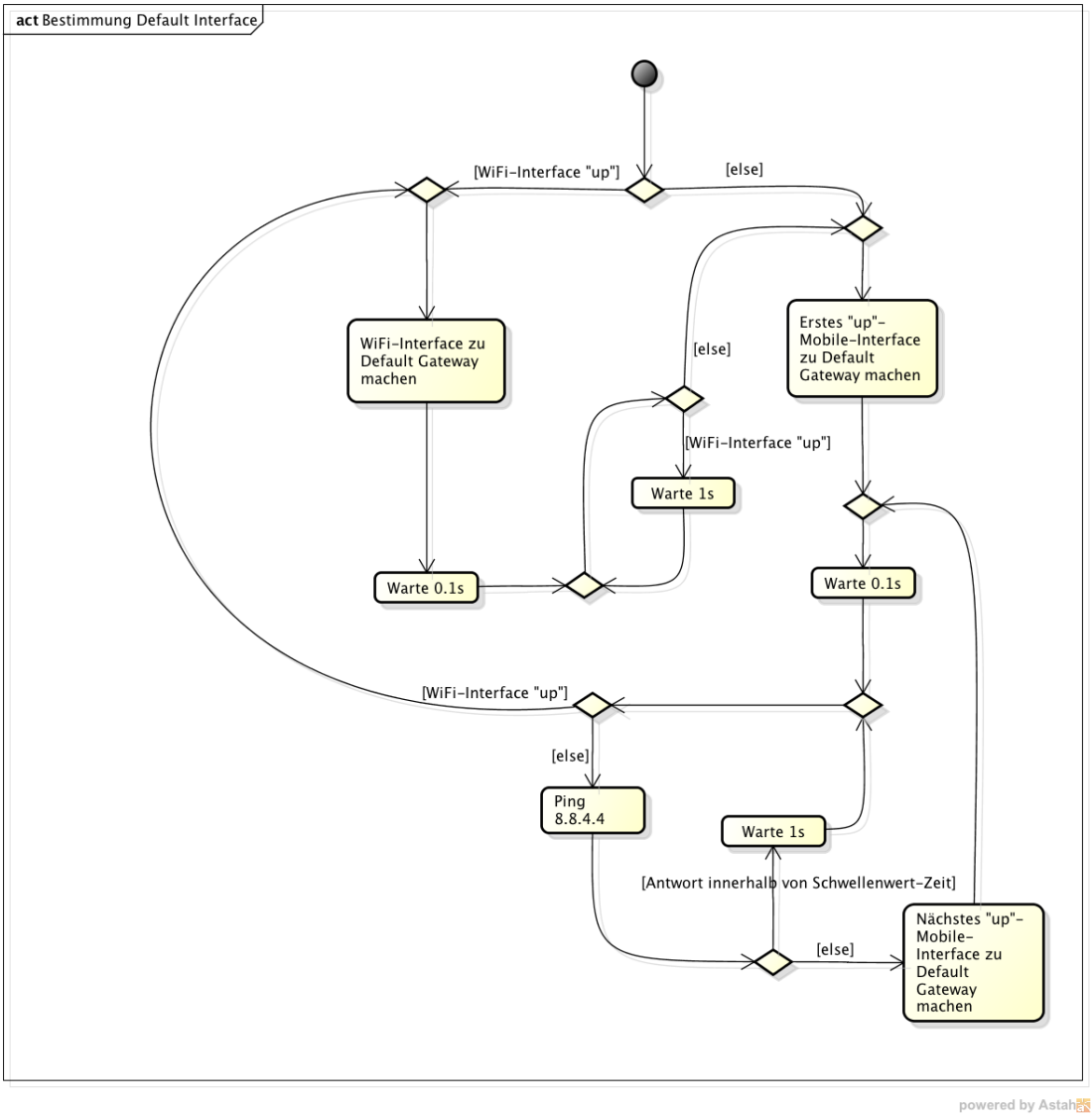

Abbildung 5.4.: Aktionsdiagramm Bestimmung Default Gateway mit WiFi-Priorisierung

5.4.3. Pakethandling

Das Sequenzdiagramm in Abbildung 5.5 auf Seite 77 stellt den Ablauf des Pakethandlings dar. Nicht dargestellt ist die Behandlung der Pakete in die Gegenrichtung: Diese läuft in Gegenrichtung durch die offenen Verbindungen ab.

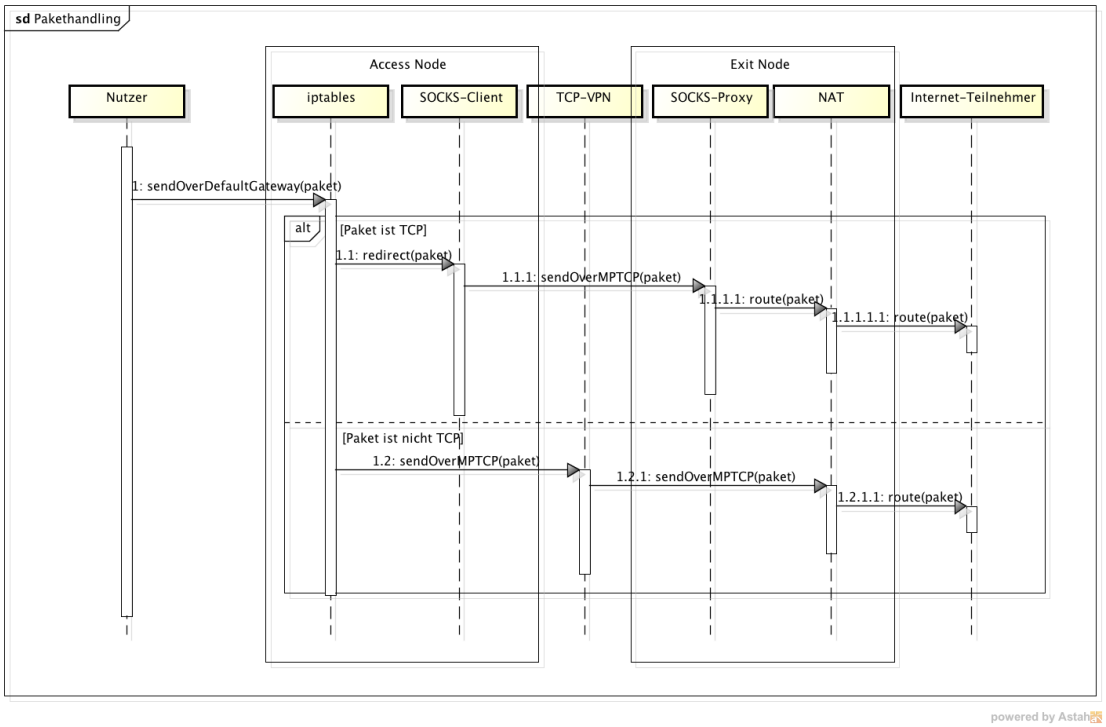

Abbildung 5.5.: Sequenzdiagramm Pakethandling

6. System *MPTCP-WiFi*

6.1. Übersicht zu *MPTCP-WiFi*

6.1.1. Übersicht virtuelle Umgebung

Um das Produkt *MPTCP-WiFi* umzusetzen und auf eine geeignete Weise testen zu können, wurde mithilfe von virtuellen Maschinen eine Umgebung umgesetzt, die der tatsächlich geplanten Umgebung entspricht. So können alle System- und Softwarekomponenten bereits wie in der finalen Umsetzung genutzt werden. Während der Umsetzungsphase hat sich gezeigt, dass der gegebene MPTCP-Kernel nicht auf *IPmotion CAR-A-WAN.coach Plus Vorserie* lauffähig ist, wodurch keine physischen Tests mit der finalen Hardware durchgeführt werden konnten. Da die virtuelle Umgebung bereits alle Komponenten abdeckt, ist das Testen der Umsetzung auch so möglich. Lediglich die Tests unter Verwendung realer Mobilfunknetze müssen getrennt durchgeführt werden.

Abbildung 6.1 auf Seite 79 zeigt die virtuelle Umgebung. Es ist dargestellt, wie die Clients („VMclient“) sich über das Fahrzeugnetzwerk („Bus Net“) mit dem Access Node verbinden und dieser über zwei Mobilfunk-Provider (blau, grün) über das Internet („Simulated Internet“) den Exit Node erreicht. Von dort aus kann der Datenverkehr über ein physisches Netzwerk („BA Net“) an das Internet geleitet werden. Durch diesen Aufbau kann die komplette Umgebung des Produkts *MPTCP-WiFi* vom Client bis ins Internet getestet werden. MPTCP wird zwischen dem Access Node und dem Exit Node eingesetzt. Da weder Client Computer (z. B. Android-Gerät) noch Endserver (z. B. Youtube) üblicherweise MPTCP unterstützen, bieten Access Node und Exit Node einen Daten-Tunnel an. So kann von MPTCP und damit von der parallelen Nutzung mehrerer Internetverbindungen sowie von unterbruchsfreien Wechseln zwischen diesen profitiert werden.

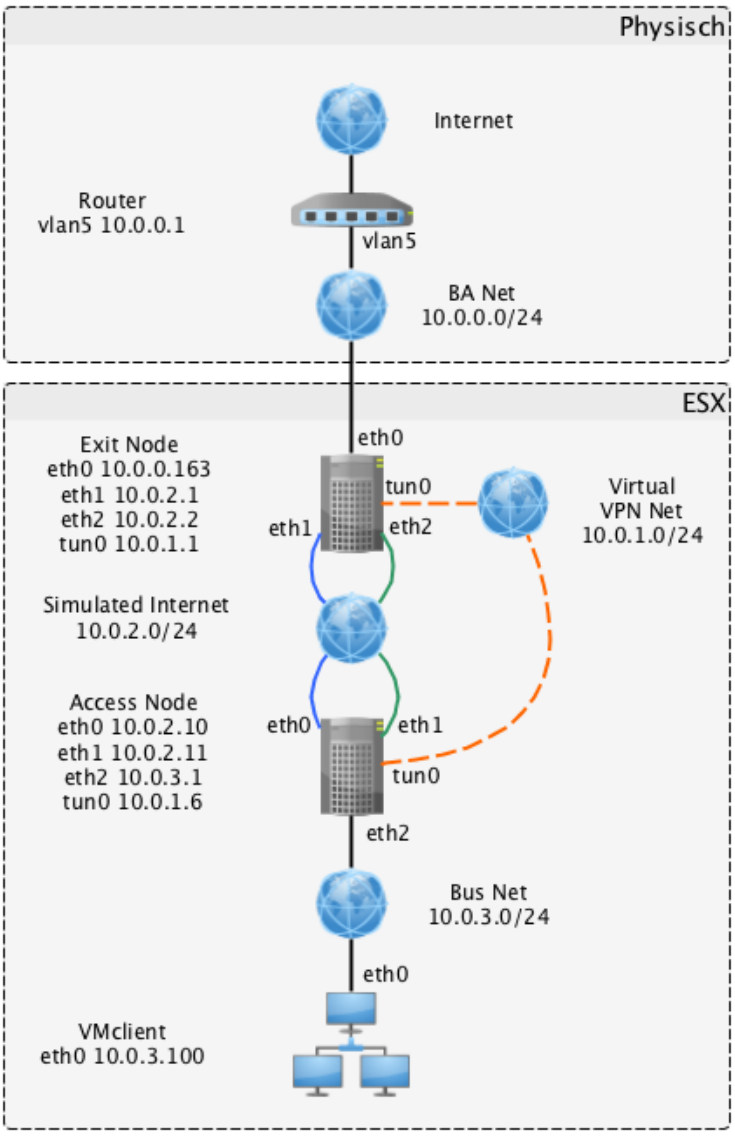

Abbildung 6.1.: Aufbau der virtuellen Umgebung

6.2. Dateneingang: Access Node

Der Access Node stellt gleichzeitig den Router und Wireless Access Point für die Clients, wie auch den Einstiegspunkt der Datenpakete in das System *MPTCP-WiFi* dar. Er ist sowohl mit dem Client-Netzwerk, als auch über mehrere Wege mit dem Internet (Beispiel: 2× Mobilfunk und 1× WiFi) verbunden. Er stellt den Default Gateway für die Clients dar und soll ihre Datenpakete mithilfe von MPTCP über den Exit Node an das Internet senden. Damit können die Clients von den Vorteilen von MPTCP profitieren, ohne selbst mit MPTCP umgehen können zu müssen.

Die eintreffenden Datenpakete werden auf dem Access Node in TCP- und Nicht-TCP unterschieden. Nicht-TCP-Pakete werden über TCP-VPN-Tunnel an den Exit Node geleitet, damit sie MPTCP nutzen können. TCP-Pakete werden wegen der TCP-over-TCP Problematik über SOCKS an den Exit Node gesendet.

Der Access Node überwacht seine Internet-Verbindungen laufend auf Unterbrüche (beispielsweise Funklöcher) und reagiert darauf, um unterbruchsfrei weiterarbeiten zu können. Nur wenn alle Verbindungen unterbrochen sind, ist kein Datenverkehr mehr möglich. Die Funktionalität wird wieder aufgenommen, sobald wieder eine Verbindung zur Verfügung steht.

Abbildung 6.2 zeigt, wie sich der Access Node in das System integriert. Die Clients verbinden sich innerhalb des Fahrzeuges über WiFi mit dem Access-Node. Dieser sendet den Nicht-TCP-Verkehr (beispielsweise UDP) über VPN, den TCP-Verkehr über SOCKS, über seine Internet-Verbindungen (beispielsweise GSM) an den Exit Node. So kann der Datenverkehr an beliebige Server im Internet gelangen.

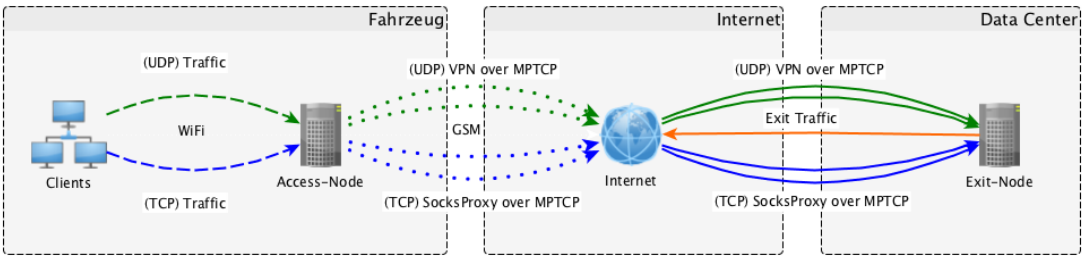

Abbildung 6.2.: Access Node Übersicht

6.2.1. Source Routing

Typischerweise wird der Weg aufgrund der Ziel-IP-Adresse mithilfe der Routing-Tabelle bestimmt. Bei MPTCP kommt der Ansatz von Source Routing zum Zug. Es können für die verschiedenen Interfaces eigene Routing-Tabellen erstellt werden. Bindet sich ein Dienst an ein bestimmtes Interface, werden Pakete über die Routing-Tabelle von diesem Interfaces versendet. Hat ein Interface keine eigene Routing-Tabelle, oder ist ein Dienst nicht an ein bestimmtes Interface gebunden, wird die globale Tabelle verwendet.

Damit MPTCP Pakete parallel über die verschiedenen Interfaces des Access Nodes schicken kann, muss für jedes Interface eine eigene Routing-Tabelle erstellt werden.

6.2.2. iptables

Das Linux-Programm *iptables* wird eingesetzt, um die Pakete je nach Protokoll unterschiedlich zu behandeln. Damit können TCP-Pakete über SOCKS weitergeleitet werden.

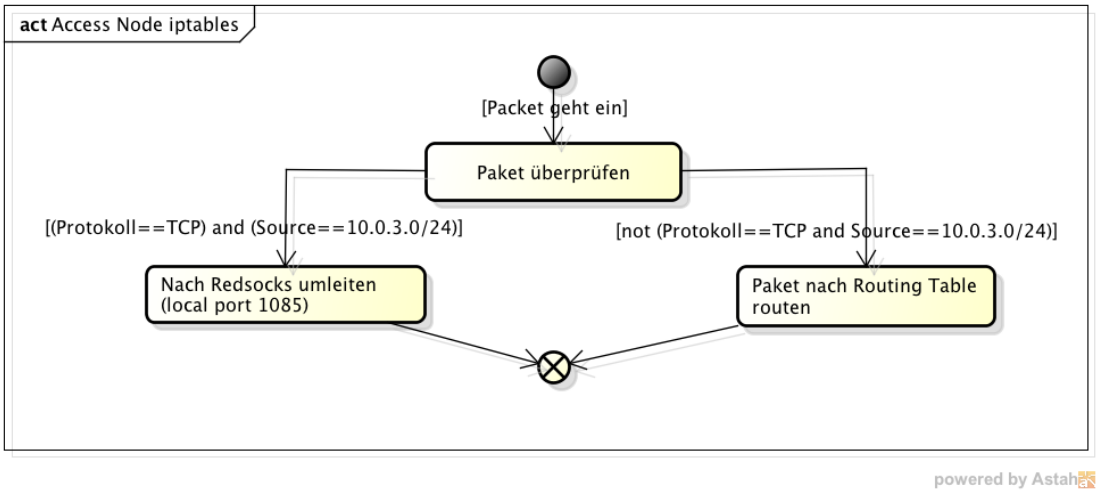

Abbildung 6.3.: Access Node iptables

Abbildung 6.3 auf Seite 81 zeigt den Weg eines Pakets bei Eingang im Access Node. Es wird überprüft, ob das Paket sowohl über TCP verschickt wurde als auch vom Fahrzeugnetz (im Beispiel in der Prototyp-Umgebung 10.0.3.0/24) stammt. Trifft beides

zu, wird das Paket an den lokalen Port 1085 geleitet. Dort nimmt der SOCKS-Client *Redsocks* das Paket entgegen und leitet es über MPTCP an den Exit Node weiter. Treffen die Bedingungen nicht zu, wird der Traffic normal geroutet, was in diesem Fall bedeutet, dass der Traffic über die VPN-Verbindung zum Exit Node geleitet wird.

Hinweis: Das Diagramm ist zugunsten der Übersichtlichkeit vereinfacht worden. Beim vollständigen Ablauf werden noch Ausnahmen für lokale Netzwerke gemacht. Treffen diese Ausnahmen zu, wird der Traffic trotzdem geroutet und nicht über den SOCKS-Dienst geleitet.

Das komplette Skript zur iptables-Konfiguration findet sich im Anhang C.1 auf Seite X.

6.2.3. OpenVPN

OpenVPN überträgt die Nicht-TCP-Daten über ein TCP-VPN an den Exit Node. Da das VPN über eine TCP-Verbindung läuft, kann es, sowie die damit übertragenen Pakete, von MPTCP profitieren, obwohl sie eigentlich nicht MPTCP-fähig sind.

Abbildung 6.1 zeigt, wie die Netzwerke in der Testumgebung konfiguriert sind und an welcher Stelle der VPN-Tunnel zum Einsatz kommt. Der Access Node dient hierbei als VPN-Client. Die Einstellungen sind daher unter /etc/openvpn/client.conf vorgenommen.

```
1 client
2 remote 10.0.2.1 1194 -- IP a Exit Node (VPN-Server)
3 proto tcp
4 client-config-dir ccd
5 route 10.0.3.0 255.255.255.0
6 client-to-client
7 push "route 10.0.3.0 255.255.255.0"
```

Listing 6.1: Erklärung VPN-Einstellungen Client

Listing 6.1 auf Seite 82 zeigt die wichtigsten Einstellungen: Zeile 1 setzt den Client-Modus.

Zeile 2 stellt das Ziel der VPN-Verbindung ein – in diesem Falle IP-Adresse und Port des VPN-Servers auf dem Exit Node. Zeile 3 sorgt dafür, dass der VPN-Tunnel über TCP läuft – nur so kann MPTCP eingesetzt werden. Zeile 4 bestimmt, wo die Zuordnung zwischen VPN Client ID und Netzwerk abgelegt ist. Zeile 5 teilt dem VPN-Server (Exit Node) mit, welches Netzwerk über den eigenen Access Node erreichbar ist. Zeile 6 legt fest, dass verschiedene Clients aufeinander zugreifen können – dadurch ist es möglich, dass von einem Fahrzeug Verbindungen in ein anderes Fahrzeug aufgebaut werden können. Zeile 7 teilt den anderen Clients (Fahrzeugen bzw. deren Access Nodes) mit, welches das eigene Netzwerk ist, damit diese darauf zugreifen können.

Das komplette Skript zur VPN-Konfiguration des Access Nodes findet sich im Anhang C.2 auf Seite XI.

6.2.4. Redsocks

Redsocks ist ein SOCKS-Client und leitet die TCP-Pakete an den Exit Node. Diese können nicht ebenfalls über die TCP-VPN-Verbindung gesendet werden, da eine Schachtelung von TCP-Paketen in TCP-Pakete für Probleme sorgen kann (TCP-over-TCP-Problematik). Dies ist aber auch nicht nötig, da TCP-Pakete direkt um MPTCP erweitert werden können. Über SOCKS können diese daher unmittelbar an den Exit Node weitergeleitet werden, der damit als Proxy dient, um auf der Zwischenstrecke MPTCP nutzen zu können.

Die RedSocks-Konfiguration findet sich im Anhang C.3 auf Seite XI.

6.2.5. DHCP-Server

Um den Clients den Zugriff über den Access Node zu geben, bietet dieser einen DHCP-Server an. Über das WiFi können die Clients so IP-Adressen beziehen und bekommen die IP-Adresse des Access Nodes als Default Gateway mitgeteilt.

Die DHCP-Server-Konfiguration findet sich im Anhang C.4 auf Seite XII. In diesem Beispiel wird das Netz 10.0.3.0/24 verwendet. Im Praxiseinsatz wird jedem Fahrzeug ein eigenes Subnetz zugeteilt. Dies ermöglicht den Fernzugriff auf Geräte, die sich im Bus

befinden, wie z. B. die Kameraanlage. Das Netz 10.0.3.0/24 dient daher nur als Beispiel, je nach Bedürfnissen und Praxisanforderungen können Netzadresse und Range angepasst werden.

Ist der Fernzugriff auf Geräte im Bus nicht erwünscht oder nötig, kann ein NAT auf Seiten des Buses eingesetzt werden. Dadurch kann das gleiche Subnetz innerhalb von allen Bussen eingesetzt werden. Dies würde potenziell den Aufwand im Produktiveinsatz reduzieren.

6.3. Datenausgang: Exit Node

Der Exit Node stellt die Gegenverbindungsstelle des Access Nodes dar. Da heutige Server meist noch kein MPTCP unterstützen, dient der Exit Node als MPTCP-Proxy.

Der Exit Node betreibt sowohl einen VPN- als auch einen SOCKS-Server. Über diese beiden Server empfängt er die Pakete der Clients, die ihm vom Access Node geschickt werden, und leitet sie an das Internet weiter. Die Antwort-Pakete sendet er über die offenen Kanäle zurück.

6.3.1. iptables

Mittels *iptables* wird auf dem Exit Node ein NAT eingerichtet. Da der Exit Node nur über eine beschränkte Anzahl öffentlicher IP-Adressen verfügt, jedoch eine grosse Anzahl Clients über *MPTCP-WiFi* das Internet nutzen, ist ein NAT erforderlich.

Die Abbildung 6.4 auf Seite 85 zeigt den Ablauf beim Transport von Paketen durch *iptables* auf dem Exit Node. Geht ein Paket ein, bei welchem die Quellenadresse 10.0.3.0/24 (Bus-Netz) lautet, wird ein NAT genutzt. In diesem Aufbau ist die „externe“ Adresse des Exit Nodes 10.0.0.163. Somit wird die Quelladdresse der Pakete auf 10.0.0.163 geändert und weitergeleitet. Der Exit Node führt Buch über die Anfragen und wenn eine Antwort zurück kommt, weiss er, an wen er sie weiterleiten muss.

In einem produktiven System wäre die „externe“ Adresse nicht wie in diesem Beispiel 10.0.0.163, sonder eine im Internet erreichbare Adresse.

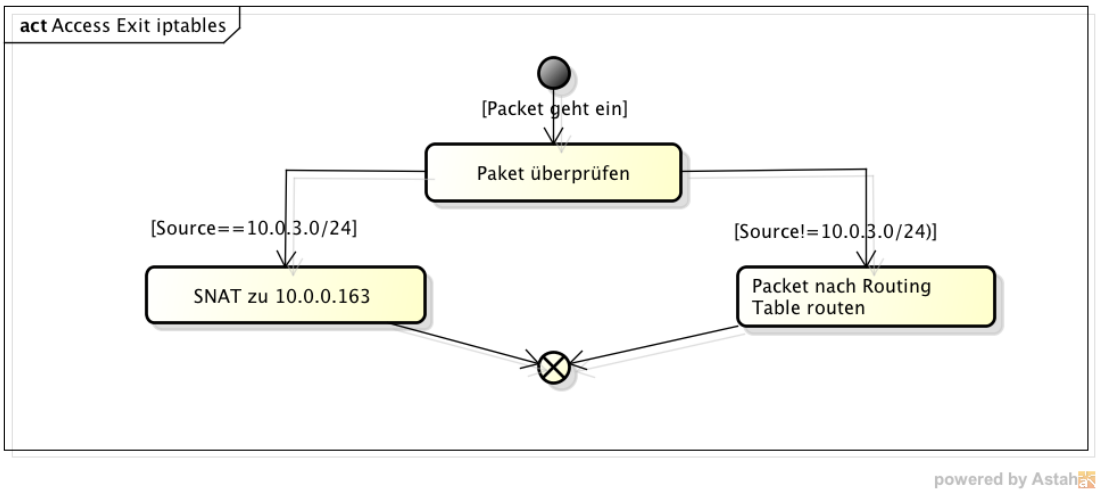

Abbildung 6.4.: Exit Node iptables

Die komplette *iptables*-Konfiguration findet sich im Anhang D.2 auf Seite XV.

6.3.2. OpenVPN

Als VPN-Server wird *OpenVPN* eingesetzt.

Abbildung 6.1 zeigt, wie die Netzwerke in der Testumgebung konfiguriert sind und an welcher Stelle der VPN-Tunnel zum Einsatz kommt. Der Exit Node dient hierbei als VPN-Server. Die Einstellungen sind daher unter /etc/openvpn/server.conf vorgenommen.

```
1 proto tcp
2 server 10.0.1.0 255.255.255.0
3 push "route 10.0.3.0 255.255.255.0"
4 client-config-dir ccd
5 route 10.0.3.0 255.255.255.0
```

Listing 6.2: Erklärung VPN-Einstellungen Server

Listing 6.2 auf Seite 85 zeigt die wichtigsten Einstellungen: Zeile 1 sorgt dafür, dass der VPN-Tunnel über eine TCP-Verbindung abläuft – nur so kann diese MPTCP einsetzen.

Zeile 2 stellt das VPN-Netzwerk ein, um es den Clients bekannt geben zu können. Zeile 3 gibt ein Netzwerk eines Fahrzeugs allen anderen Fahrzeugen bekannt. Zeile 4 gibt den Ort der Zuordnung zwischen VPN Client ID und Netzwerk an. Zeile 5 stellt ein, dass das angegebene Fahrzeugnetz über das VPN-Netzwerk erreichbar ist.

Das komplette Skript zur VPN-Konfiguration des Exit Nodes findet sich im Anhang D.1 auf Seite XIV.

6.3.3. Dante

Der SOCKS-Proxy *Dante* wird auf dem Exit Node als Gegenstelle für die TCP-Verbindungen der Clients eingesetzt, die vom Access Node über SOCKS geleitet werden. Da MPTCP auf den Servern im Internet noch nicht verbreitet ist, ist der Proxy nötig, um auf der Zwischenstrecke MPTCP einsetzen zu können.

Die Dante-Konfiguration findet sich im Anhang D.3 auf Seite XV.

6.4. Software

Um das Reagieren auf die Aussenwelt von *MPTCP-WiFi* umzusetzen, wurden verschiedene Softwarekomponenten umgesetzt.

6.4.1. Evaluierung Umsetzung

Die Softwarekomponenten können auf zwei Wege umgesetzt werden:

Kompiliertes Programm: Es könnte eine Applikation entwickelt werden, die alle Aufgaben übernimmt und als kompiliertes Programm ausgeführt wird. Beispiel: C++.

Shell-Programm: Die Aufgaben könnten per Shell-Skript-Programm umgesetzt werden. Beispiel: Bourne-Shell

Da die Software auf einem festen Gerät ausgeführt wird (*IPmotion CAR-A-WAN.coach Plus Vorserie*) und auf diesem Gerät der Grösste Teil der bestehenden Logik über Shell-Skripte umgesetzt wurden, bietet sich an, die neuen Komponenten auch als Shell-Skripte zu realisieren. Dadurch sind die Komponenten einheitlich zu den bestehenden Komponenten umgesetzt und könnten damit bei einer Portierung nach selbem Prinzip angepasst werden.

Die üblichen Nachteilen von Shell-Skripte treffen in dieser Umgebung nicht zu: Da die Software auf *IPmotion CAR-A-WAN.coach Plus Vorserie* feste, bekannte Versionen trägt, sind auch die Shell-Ausgaben der Befehle bekannt und immer gleich. Auf der Seite der Vorteile kann so ein unnötiger Overhead vermieden werden, da ein C++-Programm zur Konfiguration ebenfalls viele Befehle über Shell absetzen müsste.

6.4.2. Interface up/down

Wenn ein Interface hinzugefügt wird, welches kein Loopback-Interface ist, muss das Routing für dieses Interface konfiguriert werden. Dafür muss für jedes Interface eine eigene Routing-Tabelle hinzugefügt werden, damit MPTCP die Pakete über jedes einzelne Interface korrekt routen kann. Ein Eintrag mit Informationen über das Interface wird in `/tmp/mptcp-wifi/info/(Interfacename)` gespeichert, damit andere Softwarekomponenten diese Informationen auslesen können. Es wird hierbei im networking-Service auf den „if-up“-Event gehört. Das Shell-Programm findet sich im Anhang B.1 auf Seite IV.

Wird ein Interface entfernt, werden die Routing-Tabelle sowie die Informationsdatei wieder gelöscht. Hierzu wird beim networking-Service der „if-down“-Event genutzt. Das Shell-Programm findet sich im Anhang B.2 auf Seite VI.

6.4.3. Default-Route-Überwachung

Da neue TCP-Verbindungen jeweils über die Default Route aufgebaut werden, bevor die Verbindung um MPTCP erweitert wird, muss die Verbindung regelmässig auf Nutzbarkeit hin überprüft werden. So kann gegebenenfalls die Default Route angepasst werden. Dieses Programm muss beim Start des Systems aufgerufen werden, damit es im Hintergrund laufend durchgeführt werden kann.

Es wird im Sekundentakt jeweils ein Ping abgesendet. Kommt bis eine Sekunde später keine Antwort zurück, wird das nächste verfügbare Interface als Default Route gewählt. Somit wird jeweils im Kreis durch die verfügbaren Interfaces gewechselt, bis ein funktionierendes Interface gefunden wird. Nach dem Wechsel des Interfaces wird bis zum nächsten Ping nur 0.1 Sekunde gewartet, um schnell zu prüfen, ob dieses Interface Emfang hat, oder gegebenenfalls weiter zu wechseln. Wurde ein funktionierendes Interface gefunden, wird dies zur Default Route und sekündlich überwacht.

Als Erweiterung gegenüber der Aufgabenstellung kann ein Priority-Interface definiert werden. Wenn dieses Interface „up“ ist, wird nur dieses Interface genutzt und die anderen Interfaces für MPTCP abgeschaltet. Damit kann beispielsweise im Depot, wo kostenloses WiFi verfügbar ist, auf die kostenpflichtigen Mobilfunkinterfaces verzichtet werden.

Das Shell-Programm findet sich im Anhang B.3 auf Seite VI.

6.5. Testcases

Um die Erreichung der Ziele zu überprüfen, wurden zu Beginn der Analyse anhand der technischen Use Cases folgende Test Cases aufgestellt. So kann sichergestellt werden, dass auf alle Einflüsse auf das System korrekt reagiert wird.

6.5.1. Use Case 2: Paket senden

TC01 Es ist möglich, wenn alle 3 Interfaces nutzbar sind, ein Paket über TCP zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC02 Es ist möglich, wenn nur 2 Mobile-Interfaces und kein WiFi-Interface nutzbar sind, ein Paket über TCP zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC03 Es ist möglich, wenn nur 1 Mobile-Interface und 1 WiFi-Interface nutzbar sind, ein Paket über TCP zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC04 Es ist möglich, wenn nur 1 Mobile-Interface und kein WiFi-Interface nutzbar ist, ein Paket über TCP zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC05 Es ist möglich, wenn nur 1 WiFi-Interface und kein Mobile-Interface nutzbar ist, ein Paket über TCP zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC06 Es ist möglich, wenn alle 3 Interfaces nutzbar sind, ein Paket über UDP zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC07 Es ist möglich, wenn nur 2 Mobile-Interfaces und kein WiFi-Interface nutzbar sind, ein Nicht-TCP-Paket zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC08 Es ist möglich, wenn nur 1 Mobile-Interface und 1 WiFi-Interface nutzbar sind, ein Nicht-TCP-Paket zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC09 Es ist möglich, wenn nur 1 Mobile-Interface und kein WiFi-Interface nutzbar ist, ein Nicht-TCP-Paket zu versenden. Erwartet: Paket wird erfolgreich versendet.

TC10 Es ist möglich, wenn nur 1 WiFi-Interface und kein Mobile-Interface nutzbar ist, ein Nicht-TCP-Paket zu versenden. Erwartet: Paket wird erfolgreich versendet.

6.5.2. Use Case 3: Zu hohe Latenz haben

Ausangslage vor jedem Test: Es sind 2 Mobile-Interfaces und kein WiFi-Interface verfügbar.

TC11 TCP-Download wird gestartet. Default Gateway wird abgedeckt. Erwartet: Download läuft weiter.

TC12 UDP-Pakete werden laufend versendet. Default Gateway wird abgedeckt. Erwartet: Pakete werden weiterhin erfolgreich versendet.

TC13 Default Gateway wird abgedeckt. Neue TCP-Verbindung wird aufgebaut. Erwartet: Verbindungsaufbau erfolgreich.

6.5.3. Use Case 4: Keinen Empfang haben

Ausangslage vor jedem Test: Es sind 2 Mobile-Interfaces und 1 WiFi-Interface verfügbar. Das WiFi-Interface ist das Default Gateway.

TC14 TCP-Download wird gestartet. WiFi-Interface wird down genommen. Erwartet: Download läuft weiter.

TC15 UDP-Pakete werden laufend versendet. WiFi-Interface wird down genommen. Erwartet: Pakete werden weiterhin erfolgreich versendet.

TC16 WiFi-Interface wird down genommen. Neue TCP-Verbindung wird aufgebaut. Erwartet: Verbindungsaufbau erfolgreich.

6.5.4. Use Case 5: Default Gateway ändern

TC17 Alle 3 Interfaces sind verfügbar. Erwartet: Das WiFi-Interface ist Default Gateway.

TC18 Alle 3 Interfaces sind verfügbar. Das WiFi-Interface wird down genommen. Erwartet: Ein Mobile-Interface wird Default Gateway.

TC19 2 Mobile-Interfaces sind verfügbar, kein WiFi-Interface. Das Default-Mobile-Interface wird abgedeckt. Erwartet: Das andere Mobile-Interface wird zum Default Gateway.

TC20 1 Mobile-Interface ist verfügbar, 1 Mobile-Interface ist abgedeckt, kein WiFi-Interface. Das WiFi-Interface wird up geschaltet. Erwartet: Das WiFi-Interface wird zum Default Gateway.

TC21 1 Mobile-Interface und 1 WiFi-Interface sind verfügbar, 1 Mobile-Interface ist abgedeckt. Die Abdeckung des Mobile-Interfaces wird entfernt. Erwartet: Das WiFi-Interface bleibt Default Gateway.

6.5.5. Use Case 6: Interface für MPTCP dekonfigurieren

TC22 Das WiFi-Interface ist verbunden und für MPTCP konfiguriert. Es wird down genommen. Erwartet: Das WiFi-Interface ist für MPTCP dekonfiguriert.

6.5.6. Use Case 7: Neu nutzbare Latenz erreichen

TC23 2 Mobile-Interfaces sind abgedeckt, kein WiFi-Interface. Die Abdeckung eines Mobile-Interfaces wird entfernt und eine TCP-Übertragung gestartet. Erwartet: Die Verbindung ist erfolgreich.

TC24 2 Mobile-Interfaces sind abgedeckt, kein WiFi-Interface. Die Abdeckung eines Mobile-Interfaces wird entfernt und ein Nicht-TCP-Paket versendet. Erwartet: Der Versand ist erfolgreich.

TC25 1 Mobile-Interface ist verfügbar, 1 Mobile-Interface ist abgedeckt, kein WiFi-Interface. Ein TCP-Download wird gestartet. Während der Download läuft, wird die Abdeckung des zweiten Mobile-Interfaces entfernt. Erwartet: Es ist auch auf dem zweiten Mobile-Interface eine Übertragung messbar.

6.5.7. Use Case 8: Neu Empfang haben

TC26 2 Mobile-Interfaces sind abgedeckt, kein WiFi-Interface. Das WiFi-Interface wird up geschaltet und eine TCP-Übertragung gestartet. Erwartet: Die Verbindung ist erfolgreich.

TC27 2 Mobile-Interfaces sind abgedeckt, kein WiFi-Interface. Das WiFi-Interface wird up geschaltet und ein Nicht-TCP-Paket versendet. Erwartet: Der Versand ist erfolgreich.

6.5.8. Use Case 9: Interface für MPTCP konfigurieren

TC28 2 Mobile-Interfaces sind abgedeckt, kein WiFi-Interface. Das WiFi-Interface wird up geschaltet. Erwartet: Das WiFi-Interface ist für MPTCP konfiguriert.

6.5.9. Reboot Access Node

TC29 Alle Interfaces sind verfügbar. Der Access Node wird neu gestartet. Erwartet: Alle drei Interfaces sind für MPTCP konfiguriert.

TC30 2 Mobile-Interfaces sind abgedeckt, kein WiFi-Interface. Nach einiger Zeit werden die Abdeckungen der Mobile-Interfaces entfernt und das WiFi-Interface up geschaltet. Erwartet: Alle drei Interfaces sind für MPTCP konfiguriert.

6.6. Testdurchführung

Die Testcases werden auf der virtualisierten Umgebung umgesetzt. Die Mobile und WiFi Interfaces sind somit normale Interfaces in ESXi. Das unterschiedliche Verhalten von z. B. bevorzugen von WiFi interface kann dennoch getestet werden.

Tests werden nach listing 6.3 durchgeführt. Stellvertretend für das Interface ist ethx.

```
1  # a. Kein Durchsatz auf Interface ethx
2  iptables -I OUTPUT -o ethx -j DROP
3  # b. Wieder Durchsatz auf Interface ethx
4  iptables -D OUTPUT 1
5  # c. Latenz zu hoch auf Interface ethx
6  iptables -I OUTPUT -o ethx -j DROP
7  # d. Normale Latenz auf Interface ethx
8  iptables -D OUTPUT 1
9  # e. Interface ethx geht down
10 ifdown ethx
11 # f. Interface ethx ist wieder up
```

```
12 ifup ethx
13 # g. Nicht-TCP-Traffic testen
14 ping google.ch
15 # h. Nicht-TCP-Traffic laufend testen
16 ping google.ch
17 # i. Laufende TCP-Verbindung Testen
18 wget http://goo.gl/HKlDg1 -c --limit-rate=100k
19 # j. TCP-Verbindung Testen
20 w3m google.ch
21 # k. System neustart
22 reboot
23 # l. Auslesen der MPTCP-Konfiguration
24 ip rule show
25 ip route show table ethx
26 # m. Default Route auslesen
27 route
```

Listing 6.3: Befehle für Testdurchführung

Dem Status-Kürzel in der Spalte „Done“ wird der Buchstabe der jeweiligen Action vorgestellt, z. B. a.ok für Erfolgreicher Test mit „Kein Durchsatz auf Interface ethx“. Für welches Interface ethx steht, wird im Test Case beschrieben. Werden mehrere Aktionen ausgeführt, werden sie entsprechend aufgelistet. Beschreibt ein Testcase seine Vorraussetzungen, wie z.Ḃ. ethx im voraus down, wird die Aktion für die Vorbedingung nicht im Statusfeld aufgeführt.

6.6.1. Durchführung 09.06.14

Nr.	Erwartet	Resultat	Done
Use Case 2: Paket senden			
TC01	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC02	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok

Nr.	Erwartet	Resultat	Done
TC03	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC04	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC05	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC06	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC07	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC08	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC09	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC10	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
Use Case 3: Zu hohe Latenz haben			
TC11	TCP-Download läuft weiter	Download friert ein	ia.nok
TC11	Anmerkung: Zu hohe latenz durch droppen der Pakete		
TC12	Ping läuft weiter	Ping läuft weiter	h.ok
TC13	Neue TCP-Verbindung erfolgreich	Neuer Download funktioniert	i.ok
Use Case 4: Kein Empfang haben			
TC14	TCP-Download läuft weiter	Download läuft weiter	i.ok
TC15	Nicht-TCP-Übertragung läuft weiter	Pings an google.ch laufen weiter	h.ok
TC16	Neue TCP-Verbindung möglich	Aufruf von www.google.ch funktioniert	j.ok
Use Case 5: Default Gateway ändern			
TC17	WiFi-Interface ist Default Gateway	WiFi ist Default Gateway	m.ok
TC18	Ein Mobileinterface wird Default Gateway	eth0 wird Default Gateway	e.ok
TC19	Anderes Mobile-Interface wird Default Gateway	Default Gateway wird nicht gewächselt	a.nok
TC20	WiFi-Interface wird Default Gateway	WiFi-Interface wird Default Gateway	f.ok

Nr.	Erwartet	Resultat	Done
TC21	WiFi-Interface bleibt Default Gateway	WiFi-Interface bleibt Default Gateway	d.ok
Use Case 6: Interface für MPTCP dekonfigurieren			
TC22	WiFi-Interface wird für MPTCP dekonfiguriert	WiFi-Interface bleibt konfiguriert	e.ok
TC22	Anmerkung: WiFi Down über ifdown eth2		
Use Case 7: Neu nutzbare Latenz erreichen			
TC23	Verbindung ist erfolgreich	Verbindung ist erfolgreich	b.ok
TC24	Versand ist erfolgreich	Nach ca. 15 Sekunden Verzögerung erfolgreich	bg.nok
TC24	Anmerkung: Die Dauer der Konvergenz ist sehr lange (ca. 15 Sekunden)		
TC25	Es ist ebenfalls auf dem zweiten Mobile-Interface eine Übertragung messbar	Übertragung auf beiden Mobile-Interfaces messbar	ib.ok
Use Case 8: Neu Empfang haben			
TC26	Verbindung ist erfolgreich	TCP-verbindung erfolgreich	fj.nok
TC26	Anmerkung: DNS abfrage schlägt jedoch fehl		
TC27	Versand ist erfolgreich	nicht erfolgreich	fg.nok
TC27	Anmerkung: OpenVPN verliert Routing Eintrag zum Host		
Use Case 9: Interface für MPTCP konfigurieren			
TC28	WiFi-Interface für MPTCP konfiguriert	WiFi-Interface für MPTCP konfiguriert	f.ok
Reboot Access Node			
TC29	Alle drei Interfaces sind für MPTCP konfiguriert	Alle drei Interfaces sind für MPTCP konfiguriert	kl.ok
TC30	Alle drei Interfaces sind für MPTCP konfiguriert	Alle drei Interfaces sind für MPTCP konfiguriert	aafl.ok

6.6.2. Durchführung 11.06.14

Nr.	Erwartet	Resultat	Done
Use Case 2: Paket senden			
TC01	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC02	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC03	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC04	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC05	TCP Paket wird erfolgreich versendet	Aufruf von www.google.ch funktioniert	j.ok
TC06	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC07	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC08	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC09	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
TC10	Nicht-TCP-Paket wird erfolgreich versendet	Ping an google.ch funktioniert	g.ok
Use Case 3: Zu hohe Latenz haben			
TC11	TCP-Download läuft weiter	TCP-Download läuft weiter	ia.ok
TC12	Ping läuft weiter	Ping läuft weiter	h.ok
TC13	Neue TCP-Verbindung erfolgreich	Neuer Download funktioniert	i.ok
Use Case 4: Kein Empfang haben			
TC14	TCP-Download läuft weiter	Download läuft weiter	i.ok
TC15	Nicht-TCP-Übertragung läuft weiter	Pings an google.ch laufen weiter	h.ok
TC16	Neue TCP-Verbindung möglich	Aufruf von www.google.ch funktioniert	j.ok
Use Case 5: Default Gateway ändern			
TC17	WiFi-Interface ist Default Gateway	WiFi ist Default Gateway	m.ok
TC18	Ein Mobileinterface wird Default Gateway	eth0 wird Default Gateway	e.ok

Nr.	Erwartet	Resultat	Done
TC19	Anderes Mobile-Interface wird Default Gateway	Default Gateway wird nicht gewächselt	a.ok
TC20	WiFi-Interface wird Default Gateway	WiFi-Interface wird Default Gateway	f.ok
TC21	WiFi-Interface bleibt Default Gateway	WiFi-Interface bleibt Default Gateway	d.ok
Use Case 6: Interface für MPTCP dekonfigurieren			
TC22	WiFi-Interface wird für MPTCP dekonfiguriert	WiFi-Interface bleibt konfiguriert	e.ok
TC23	Verbindung ist erfolgreich	Verbindung ist erfolgreich	b.ok
TC24	Versand ist erfolgreich	Nach ca. 15 Sekunden Verzögerung erfolgreich	bg.ok
TC25	Es ist ebenfalls auf dem zweiten Mobile-Interface eine Übertragung messbar	Übertragung auf beiden Mobile-Interfaces messbar	ib.ok
Use Case 8: Neu Empfang haben			
TC26	Verbindung ist erfolgreich	TCP-verbindung nicht erfolgreich	fj.nok
TC27	Versand ist erfolgreich	nicht erfolgreich	fg.ok
Use Case 9: Interface für MPTCP konfigurieren			
TC28	WiFi-Interface für MPTCP konfiguriert	WiFi-Interface für MPTCP konfiguriert	f.ok
Reboot Access Node			
TC29	Alle drei Interfaces sind für MPTCP konfiguriert	Alle drei Interfaces sind für MPTCP konfiguriert	kl.ok
TC30	Alle drei Interfaces sind für MPTCP konfiguriert	Alle drei Interfaces sind für MPTCP konfiguriert	aafl.ok

6.7. Schnittpunkte zur Vor-Studienarbeit

Die Analyse, dass der Grundaufbau aus Access Node und Exit Node mit dazwischen liegendem VPN und SOCKS-Umleitung bestehen soll sowie welche Software auf Access Node und Exit Node zum Einsatz kommen soll, wurde aus der Vor-Studienarbeit in die vorliegende Arbeit übernommen.

Das Ziel des Prototyps war hier nicht, als Proof-of-Concept das Funktionieren der Idee zu zeigen, sondern eine funktionsfähige Lösung aufzustellen und zu dokumentieren, auf der für den Praxiseinsatz aufgebaut werden kann. Zu diesem Ziel wurden die Komponenten des Prototyps erweitert sowie zusätzliche Komponenten ergänzt und zudem eine ausführliche Beschreibung inklusive Installationsanleitung hinzugefügt.

Auf dem Access Node wurde gegenüber der vereinfachten, statischen Konfiguration ein automatisches Auslesen und Vornehmen der Einstellungen ergänzt. So steht das System auch nach einem Neustart wieder ohne Änderungen von Hand wieder zum Betrieb bereit und es wird nicht eine feste IP-Adresse pro Interface benötigt. Dies soll den Prototyp praxisgerechter machen.

Zudem wurde der Prototyp um die vorgesehenen Clients ergänzt. Es wird vorgesehen, dass der Access Node als Router fungiert und den Clients im Fahrzeug per DHCP IP-Adressen vergibt. Die Datenpakete für die Tests wurden so direkt von den Clients erzeugt statt auf dem Access Node selbst generiert.

Durch die Beschreibungen und der Installationsanleitung soll sichergestellt sein, dass künftige Projekte auf den Ergebnissen aufbauen können und den Prototyp direkt nachbauen können. Es war ein wichtiges Ziel, dass auch CloudGuard als Auftraggeber einen direkten Nutzen aus dem Prototypen ziehen kann.

7. Installationsanleitung

7.1. Komponenten

MPTCP-WiFi besteht aus zwei Hauptkomponenten, die den Dienst des Internetzugangs erbringen. Der Access Node nimmt den Datenverkehr der Clients im Fahrzeug entgegen und leitet diesen an den Exit Node weiter. Der Exit Node nimmt den Traffic vom Access Node entgegen und leitet den Datenverkehr schlussendlich ins Internet. Dies ist schematisch in Abbildung 7.1 dargestellt.

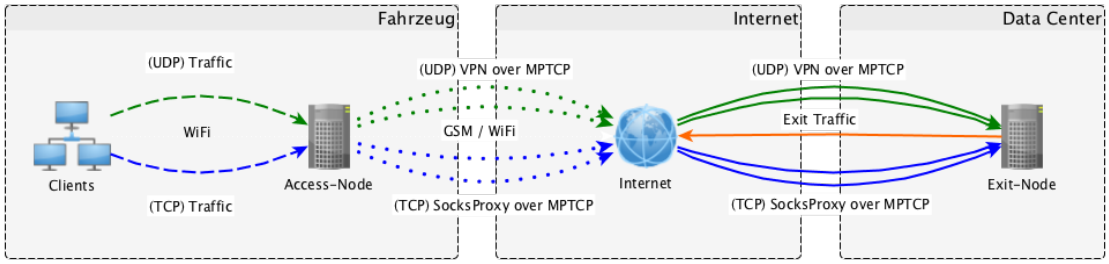

Abbildung 7.1.: Übersicht *MPTCP-WiFi*

Dienste auf den beschriebenen Systemen:

Access Node: MPTCP-Kernel, iptables, dhcpd, routewatcher, redsocks, OpenVPN

Exit Node: MPTCP-Kernel, iptables, dante, OpenVPN

VMclient: dhcp-client

In dieser Arbeit wurde die Umgebung in *VMware ESXi 5.1* umgesetzt. Für den Access Node und den Exit Node wurde als Betriebssystem *Ubuntu Server 13.10 64-Bit* eingesetzt.

Es wird in dieser Installationsanleitung die erfolgte Installation von VMware ESXi sowie von *Ubuntu Server 13.10 64-Bit* als gegeben angenommen.

7.2. Vorbereitung der Komponenten

7.2.1. Erstellen der virtuellen Maschinen

Das VMware-Netzwerk wird nach Abbildung 7.2 in vier Netzwerke aufgeteilt:

BA Net - 10.0.0.0/24 An dieses Netzwerk ist der Exit Node angeschlossen und dieser ist über das Modem mit der IP-Adresse 10.0.0.1 ans Internet angeschlossen.

VPN Net - 10.0.1.0/24 Dieses Netzwerk wird virtuell durch den *OpenVPN*-Dienst auf dem Exit Node erstellt und sichert die Verbindung zwischen dem OpenVPN-Server und OpenVPN-Client. Exit Node und Access Node erhalten eine IP-Adresse in diesem Netzwerk. Der Server leitet den eingehenden Traffic über die IP-Adresse 10.0.1.1 weiter.

Simulated Internet - 10.0.2.0/24 Dieses Netzwerk stellt symbolisch das Internet zwischen dem Access Node und Exit Node dar.

Bus Net - 10.0.3.0/24 Dieses Netzwerk wird vom Access Node verwaltet und den WiFi-Clients im Fahrzeug angeboten.

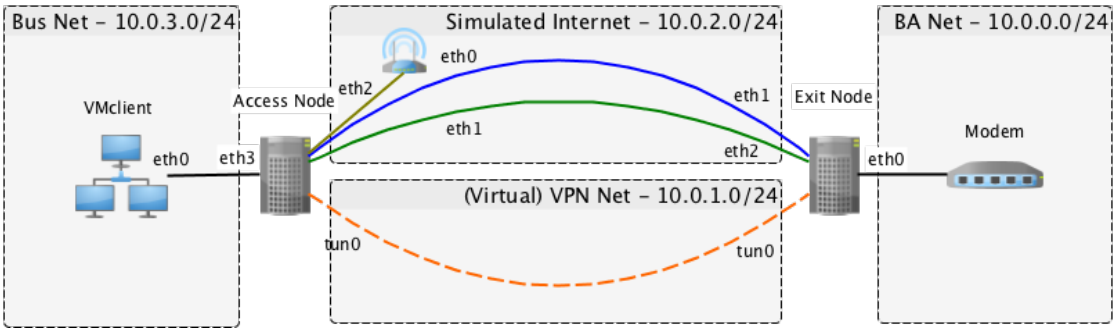

Abbildung 7.2.: Übersicht Netzwerk

Die aufgelisteten Netzwerke (VPN Net ausgeschlossen) werden in VMware konfiguriert und die VMs Abbildung 7.2 auf Seite 100 entsprechend angeschlossen. Hinweis: Der WiFi-Access-Point ist nur symbolisch eingetragen. Die WiFi Verbindung vom Access Node ist virtuell, wie die anderen Anschlüsse. Über den *vSphere Client 5.1* wird, wie in Abbildung 7.3 auf Seite 101 dargestellt, die Verbindung mit dem ESXi Server aufgebaut. Anschliessend werden unter Configuration → Networking die Netzwerke eingetragen.

Nach der Erstellung der Netzwerke werden drei unabhängige VMs erstellt und nach der folgenden Beschreibung mit den entsprechenden Netzwerken verbunden.

Access Node: 512 MB Memory, 5 GB HD, Interface 1 (eth0) verbunden mit Simulated Internet (eth1), Interface 2 verbunden mit Simulated Internet, Interface 3 (eth2) verbunden mit Simulated Internet, Interface 4 (eth3) verbunden mit Bus Net.

Exit Node: 512 MB Memory, 5 GB HD, Interface 1 (eth0) verbunden mit BA Net, Interface 2 (eth1) verbunden mit Simulated Internet, Interface 3 (eth2) verbunden mit Simulated Internet.

vmclient: 512 MB Memory, 5,GB HD, Interface 1 (eth0) verbunden mit Bus Net.

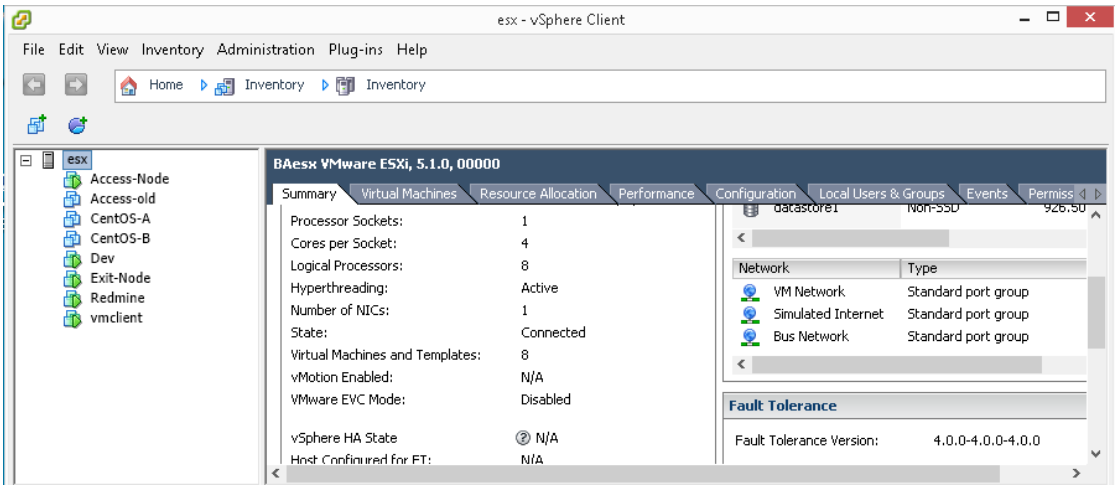

Abbildung 7.3.: vSphere Client 5.1

Die Stiftung Switch bietet auf ihrem Mirror[1] das ISO-Abbild für die Installation von Ubuntu an. Mit diesem ISO-Abbild werden die drei VMs aufgesetzt.

Hinweis: Wenn vom Access Node gesprochen wird, wird grundsätzlich auf den ESXi Access Node in Abschnitt 7.3.4 auf Seite 104 eingegangen. Der Fusion Access Node in Abschnitt 7.3.7 auf Seite 111 beschreibt optional die Umsetzung auf einem Mac Laptop und die Verwendung von USB-Modems und WiFi-Adaptern. Diese Modems sind jedoch nur für Testzwecke und nicht for die finale Hardware vorgesehen. Durch Herausforderungen mit Empfang und MPTCP-Kompatibilität von dem Mobile-Anbieter Orange wurde der ESXi Access Node dem Fusion Access Node vorgezogen.

7.3. Installation von MPTCP-WiFi

7.3.1. Installation des MPTCP-Kernels

Um von MPTCP Gebrauch machen zu können, muss der MPTCP-fähige Linux-Kernel installiert werden. Die Université catholique de Louvain bietet eine kompilierte Version dieses Kernels über ein Repository für Ubuntu 13.10 an. MPTCP wird auf dem Access Node und auf dem Exit Node installiert.

```
1 wget -q -O - http://multipath-tcp.org/mptcp.gpg.key | sudo
    apt-key add -
2 # Datei /etc/apt/sources.list.d/mptcp.list erweitern um:
3 deb http://multipath-tcp.org/repos/apt/debian saucy main
4 # Update des Repository mit MPTCP
5 sudo apt-get update
6 # Installation von MPTCP
7 sudo apt-get install linux-mptcp
```

Listing 7.1: MPTCP auf Ubuntu 13.10 Installieren

Bei einem Neustart wird das Linux mit dem neuen Kernel gestartet. Dies kann durch den in Listing 7.2 auf Seite 103 beschriebenen Befehl überprüft werden.

[1] `http://mirror.switch.ch/ftp/ubuntu-cdimage/13.10/ubuntu-13.10-server-amd64.iso`

```
1 # Antwort bei erfolgreicher Installation: 3.11.0-88-mptcp
2 uname -r
```

Listing 7.2: Kernel überprüfen

7.3.2. Nutzen von tcpdump

Durch die Installation von MPTCP mithilfe des Repositorys der Université catholique de Louvain werden auch manche Programme wie *tcpdump* erneuert, um sie ebenfalls MPTCP-fähig zu machen. Nach der installation fehlt jedoch eine Abhängigkeit für tcpdump, die Bibliothek *libsmi2*. Diese kann nach über den Befehl in Listing 7.3 auf Seite 103 nachinstalliert werden.

```
1 sudo apt-get install libsmi2ldbl
```

Listing 7.3: Abhängigkeit libsmi2 für tcpdump

7.3.3. Installation des MPTCP-Konfigurationsscripts

MPTCP teil des Linux-Kernels und agiert auch in diesem. MPTCP benötigt jedoch zusätzliche Informationen, um von den verschiedenen Interfaces aus unabhängig ins Internet routen zu können. Die *MPTCP-WiFi*-Skripte nehmen diese Konfigurationen vor, wie in Listing 7.4 auf Seite 103 gezeigt ist.

```
1 # Die Konfig. Datein nach Access Node kopieren
2 scp mptcp_wifi_up mptcp_wifi_down access:.
3 sudo cp mptcp_wifi_up /etc/network/if-up.d/mptcp_wifi_up
4 sudo cp mptcp_wifi_down /etc/network/if-down.d/mptcp_wifi_down
5 # Skripte ausfuehrbar machen
6 sudo chmod u+x /etc/network/if-up.d/mptcp_wifi_up
7 sudo chmod u+x /etc/network/if-down.d/mptcp_wifi_down
8 # Abhaengigkeit ipcalc
```

```
9  sudo apt-get install ipcalc
10 # Netzwerk neustarten
11 sudo service networking restart
```

Listing 7.4: MPTCP-WiFi Scripts

Nach dem Neustart ist MPTCP nutzbar. Zur Überprüfung von MPTCP kann der Link `http://amiusingmptcp.com/` genutzt werden. Diese Webseite prüft, ob dem TCP-Verbindungsaufbau die MPTCP-Optionen angehängt sind und zeigt das Ergebnis an. Hinweis: Unbekannte TCP-Optionen wie die Nummer 30, die von MPTCP genutzt wird, ist in vielen Einrichtungen blockiert, was auch an der HSR der Fall ist. In diesen Umgebungen kann MPTCP daher trotz korrekter Installation nicht genutzt werden.

7.3.4. Konfigurieren vom Access Node (VMwware ESXi)

Netzwerk

In der Konfigurationsdatei `/etc/network/interfaces` werden die Einstellung der jeweiligen Interfaces eingetragen, wie in Listing 7.5 auf Seite 104 gezeigt.

```
1 # Die Konfigurationsdateien auf den Access Node kopieren
2 scp interfaces_access.conf access:.
3 sudo cp interfaces_access.conf /etc/network/interfaces
4 # Netzwerk neustarten
5 sudo service networking restart
```

Listing 7.5: Access Node Interfaces

OpenVPN-Server und -Client

Die Datei openvpn_client.tar.gz aus dem Abgabe-Ordner muss auf den Access Node in den Ordner `/etc/openvpn` kopiert werden, wie in Listing 7.6 auf Seite 105 gezeigt. Der letzte Befehl des Listings startet OpenVPN.

```
1  # OpenVPN installieren
2  sudo apt-get install openvpn
3  # Die Konfigurationsdateien auf den Access Node kopieren
4  scp openvpn_client.tar.gz access:.
5  sudo cp openvpn_client.tar.gz /etc/openvpn/.
6  cd /etc/openvpn/
7  # Die Konfig auspacken
8  sudo tar -xvf openvpn_client.tar.gz
9  # OpenVPN starten
10 sudo service openvpn start
```

Listing 7.6: OpenVPN Client Installation

SOCKS-Server Redsocks

Die Datei redsocks.conf aus dem Abgabe-Ordner muss auf den Access Node kopiert werden, wie in Listing 7.7 auf Seite 105 gezeigt.

```
1  # Redsocks installieren
2  sudo apt-get install redsocks
3  # Die Konfig. Datein nach Access Node kopieren
4  scp redsocks.conf redsocks_default.conf access:.
5  sudo cp redsocks.conf /etc/.
6  sudo cp redsocks_default.conf /etc/default/.
7  # Redsocks starten
8  sudo service redsocks start
```

Listing 7.7: Redsocks-Installation

IPtables

Um den TCP-Traffic von den Clients an Redsocks zu leiten, wird *iptables* eingesetzt.

```
# iptables sollte bereits installiert sein
# ansonsten nachinstallieren mit:
sudo apt-get install iptables
# Die Konfigurationsdateien auf den Access Node kopieren
scp iptables_access.save access:.
sudo cp iptables_access.save /etc/.
# iptables manuell aktivieren
iptables-restore < /etc/iptables_access.save
```

Listing 7.8: Access Node iptables

Die iptables-Regeln werden bei einem Neustart, durch das Laden des Netzwerks, mit den Anweisungen in `/etc/network/interfaces` konfiguriert.

DHCP-Server und Routing

Der Access Node verwaltet das im Fahrzeug angebotene Netzwerk. Über DHCP verteilt er die Addressen und trägt sich als Default Gateway ein. Dadurch wird jeglicher Traffic von den Clients an den Access Node gesendet. Damit Linux eingehenden Traffic weiterroutet, muss es im Kernel aktiviert sein. Die notwendigen Befehle zur Installation sowie zur Konfiguration sind in Listing 7.9 auf Seite 106 dargestellt.

```
# /etc/sysctl.conf mit Texteditor bearbeiten
# folgende Zeile auskomentieren
# net.ipv4.ip forward=1
sudo reboot

# DHCPd installieren
sudo apt-get install isc-dhcp-server
# Die Konfigurationsdateien auf den Access Node kopieren
scp dhcpd.conf access:.
sudo cp dhcpd.conf /etc/dhcp/.
# dhcpd starten
sudo service isc-dhcp-server start
```

Listing 7.9: DHCP-Server-Installation

Nach der erfolgreichen Konfiguration des DHCP-Servers und dem Aktivieren des Routings, sollte in der Datei `/etc/sysctl.conf` die Routingzeile auskommentiert sein, wie in Abbildung 7.4 auf Seite 107 gezeigt.

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Abbildung 7.4.: Aktivieren von Routing in `/etc/sysctl.conf`

MPTCP-WiFi Route Watcher

Der Route Watcher überwacht den bestehenden Default Gateway und prüft, ob weiterhin neue MPTCP-Verbindungen darüber gestartet werde können. Sind neue Datenübertragungen innerhalb eines Zeitrahmens (1000ms RTT[2]) nicht mehr möglich, wird ein anderes Interface zum Default Gateway konfiguriert. Die Installation und das Starten des Route Watchers ist in Listing 7.10 auf Seite 107 dargestellt.

```
1  # Dateien auf Access Node kopieren
2  scp routwatcher.tar.gz 10.0.0.163:.
3  # Dateien entpacken
4  tar -xvf routewatcher.tar.gz
5  # Service kopieren
6  sudo cp routewatcher.service /usr/sbin/routewatcher
7  # init script kopieren
8  sudo cp routewatcher.init /etc/init.d/routewatcher
9  # ausfuehrbar machen
10 sudo chmod u+x /etc/init.d/routewatcher
11 sudo chmod u+x /usr/sbin/routewatcher
12 # init script aktivieren
13 sudo update-rc.d routewatcher defaults
```

[2]Round Trip Time

```
14 # Dienst starten
15 sudo service routewatcher start
```

Listing 7.10: Route-Watcher-Installation

7.3.5. Konfigurieren vom Exit Node

Netzwerk

In der Konfigurationsdatei /etc/network/interfaces werden die Einstellung der jeweiligen Interfaces eingetragen.

```
1 # Die Konfig. Datein nach Exit Node kopieren
2 scp interfaces_exit.conf exit:.
3 sudo cp interfaces_exit.conf /etc/network/interfaces
4 # Netzwerk neustarten
5 sudo service networking restart
```

Listing 7.11: Exit Node Interfaces

OpenVPN Server

Die Datei openvpn_exit.tar.gz muss aus dem Abgabe-Ordner auf den Exit-Node in den Ordner /etc/openvpn kopiert werden, wie in Listing 7.12 auf Seite 108 gezeigt. Mit dem letzten Befehl des Listings startet der OpenVPN-Server. Hinweis: Es ist aus Sicherheitsgründen empfohlen, neue Zertifikate für die Clients und den Server zu erstellen.

```
1 # OpenVPN installieren
2 sudo apt-get install openvpn
3 # Die Konfigurationsdateien auf den Exit Node kopieren
4 scp openvpn_exit.tar.gz exit:.
5 sudo cp openvpn_exit.tar.gz /etc/openvpn/.
6 cd /etc/openvpn/
```

```
7  # Die Konfig auspacken
8  sudo tar -xvf openvpn_exit.tar.gz
9  # OpenVPN starten
10 sudo service openvpn start
```

Listing 7.12: OpenVPN Client Installation

SOCKS-Server Dante

Der SOCKS-Server *Dante* nimmt den SOCKS-Datenverkehr vom Access Node entgegen und leitet ihn ans Internet weiter. Mit den Befehlen in Listing 7.13 auf Seite 109 wird der Dienst installiert und gestartet.

```
1 # dante installieren
2 sudo apt-get install dante-server
3 # Die Konfigurationsdateien nach Exit Node kopieren
4 scp danted.conf exit:.
5 sudo cp danted.conf /etc/.
6 # Dante starten
7 sudo service danted start
```

Listing 7.13: SOCKS-Server Konfiguration Installation

iptables konfigurieren

Der Traffic vom Access Node wird vom Exit Node entgegengenommen und ins Internet geleitet. Da der geroutete Traffic vom Access Node im privaten IP-Bereich liegt, setzt der Exit Node NAT mit *iptables* ein.

```
1 # iptables sollte bereits installiert sein
2 # ansonsten nachinstallieren mit:
3 sudo apt-get install iptables
4 # Die Konfigurationsdateien auf den Exit Node kopieren
```

```
5 scp iptables_exit.save exit:.
6 sudo cp iptables_exit.save /etc/.
7 # iptables manuell aktivieren
8 iptables-restore < /etc/iptables_exit.save
```

Listing 7.14: Access iptables

Die iptables-Regeln werden bei einem Neustart, durch das Laden des Netzwerks, mit den Anweisungen in /etc/network/interfaces konfiguriert.

Routing aktivieren

Damit der zu routende Traffic vom Access Node durch den Exit Node geroutet werden kann, muss dies im Kernel von Linux konfiguriert werden. Listing 7.15 auf Seite 110 zeigt die Installation und Konfiguration.

```
1 # /etc/sysctl.conf mit Texteditor bearbeiten
2 # folgende Zeile auskommentieren
3 # net.ipv4.ip_forward=1
4 # Aktivieren von Routing zur Laufzeit
5 sysctl -w net.ipv4.ip_forward=1
6 sudo reboot
```

Listing 7.15: Routing aktivieren

Nach der erfolgreichen Konfiguration des Routing, sollte in der Datei `/etc/sysctl.conf`, wie in Abbildung 7.4 auf Seite 107 gezeigt, die Routingzeile auskommentiert sein.

7.3.6. Konfigurieren vom VMclient

Der VMclient ist unabhängig von der MPTCP-Implementation des Access und Exit Nodes und muss somit nicht selbst über eine solche Installation verfügen. Der VMclient erhält über den DHCP-Server des Access Nodes seine IP-Konfiguration.

Netzwerk

Der VMclient ist standardmässig bereits auf DHCP eingestellt. Die Konfigurationsdatei sollte somit wie in Listing 7.16 aussehen.

```
1 # interfaces(5) file used by ifup(8) and ifdown(8)
2 auto lo
3 iface lo inet loopback
4
5 auto eth0
6 iface eth0 inet dhcp
```

Listing 7.16: VMclient Interfaces

7.3.7. Optional: Konfigurieren von mobilem Access Node (VMware Fusion)

Der mobile Access Node wird, im Gegensatz zum ESXi-Access-Node auf einem Laptop mit VMware Fusion betrieben. Dies erlaubt das Anschliessen von Modems und dem WiFi-Adapter an die VM.

Netzwerk

Die Netzwerk-Konfiguration beim mobilen Access Node beinhaltet die Verbindung zum Host-System (OS X), WiFi-Interface und die USB-Modems.

Damit der mobile Access Node seinen Datenverkehr nicht über den Host (OS X) leitet, aber trotzdem Remote-Zugriff vom Host aus weiter möglich ist, wird das konfigurierte VMware-Fusion-Interface auf „host-only“, wie in Abbildung 7.5 auf Seite 112 dargestellt. Daraus folgt, dass die VM ihre Daten nur über die physischen Interfaces, wie das WiFi Interface, leitet.

Anschliessend können die physischen Interfaces nach Listing 7.17 konfiguriert und angeschlossen werden.

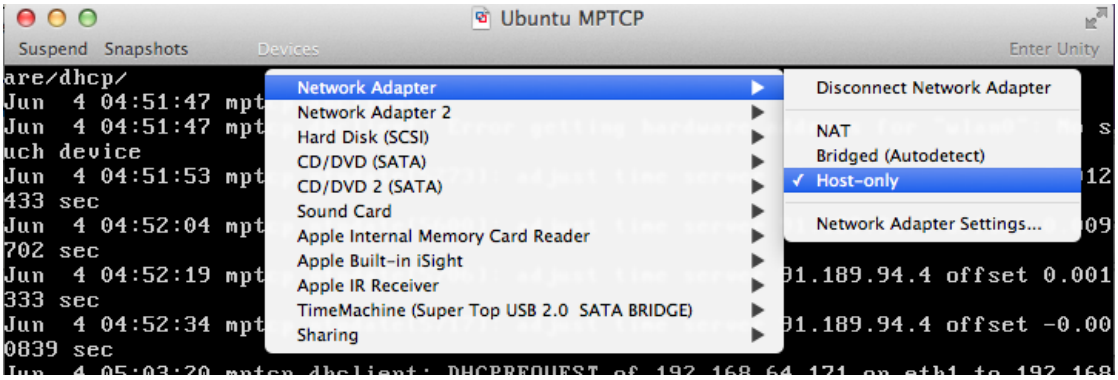

Abbildung 7.5.: VMware Fusion Host Only

```
1  # Software Installieren
2  sudo apt-get install wvdial wpasupplicant wireless-tools
3  # Die Konfig. Datein nach Access Node kopieren
4  scp configs_mobile_access.tar.gz mobile-access:.
5  # Konfig. Dateien entpacken
6  tar -xvf configs_mobile_access.tar.gz
7  # Interfaces Konfiguration
8  sudo cp interfaces /etc/network/interfaces
9  # HSR WiFi Konfiguration kopieren
10 sudo cp wpa_supplicant.conf
       /etc/wpa_supplicant/wpa_supplicant.conf
11 # Mobile Konfiguration kopieren
12 sudo cp wvdial.conf /etc/wvdial.conf
13 # Netzwerk neustarten
14 sudo service networking restart
```

Listing 7.17: Mobile Acess Node Interfaces

Die Datei /etc/wpa_supplicant/wpa_supplicant.conf ist bereits für das HSR-Secure WiFi-Netzwerk vorkonfiguriert , es muss noch mit einem aktuellen Username / Passwort ergänzt werden.

Verbinden des WiFi Adapters

Für diese Arbeit wurde der Ralink RT2870 Wireless Adapter verwendet. Dieser Adapter war direkt mit *Ubuntu Server 13.10 64-Bit* kompatibel. Der Adapter wird an den Laptop angeschlossen und beim anschliessenden Dialog, wie in Abbildung 7.6 auf Seite 113, mit „Connect to Linux“ bestätigt.

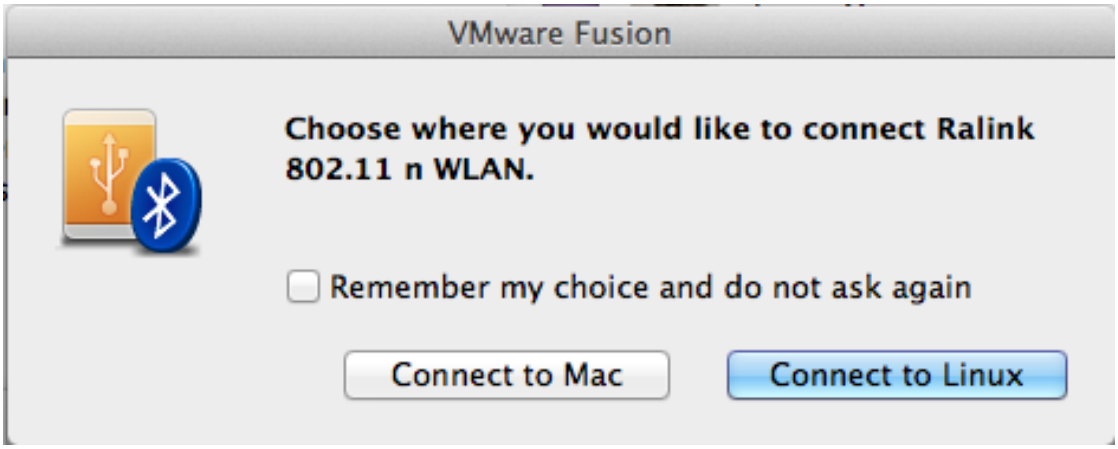

Abbildung 7.6.: WiFi Adapter Verbinden

Verbinden der USB Modems

Der Anschluss der *4G Systems XS-Stick W100* muss genau nach der folgenden Reihenfolge geschehen, sonst stimmen die device IDs im /dev/ Ordner nicht überein mit den Einstellungen in der Datei /dev/wvdial.conf. Zudem werden die Modems nicht in den „Modem-Modus“ über gehen. Diese Problematik ist beschrieben im Abschnitt 4.3.2 auf Seite 53.

1. Swisscom-Modem per USB anschliessen, Dialog „Connect to Mac“ wählen
2. Swisscom-Modem der VM zuweisen, siehe Abbildung 7.7 auf Seite 114
3. Orange-Modem per USB anschliessen, Dialog „Connect to Mac“ wählen
4. Orange-Modem der VM zuweisen, siehe Abbildung 7.7 auf Seite 114

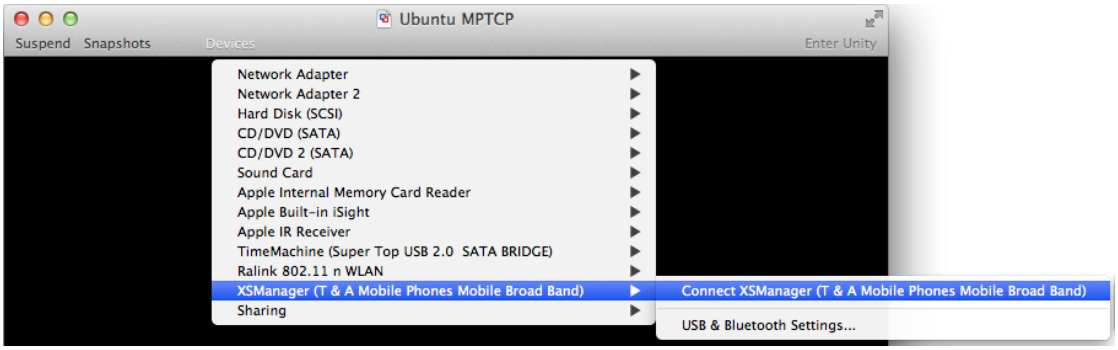

Abbildung 7.7.: *4G Systems XS-Stick W100* an VM anschliessen

Hinweis: Für das Anschliessen vom Orange-Modem, muss das „XSManager(T & A Mobile Mobile Phones Mobile Broad Band)“ ausgweählt werden, welches im Menu „Connect XSManager..“ zeigt. Dies ist hilfreich, da die beiden Modems sonst nicht unterschieden werden können.

Manuelles Starten der Modems

In dieser Arbeit wurde beobachtet, dass wenn diese Adapter statisch über die if-up scripte & /etc/interface Datei in Ubuntu konfiguriert werden, können sie aus der Interface Auflistung in ifconfig wieder verschwinden. Es hat sich jedoch gezeigt, dass sie sich stabiler verhalten, wenn sie zusätzlich noch manuel gestartet werden, siehe Listing 7.18 auf Seite 114.

```
1 # In Terminal 1
2 sudo wvdial swisscom
3 # In Terminal 2
4 sudo wvdial orange
```

Listing 7.18: Manuelles Starten der Modems

Da die Modems nicht der finalen Hardware entsprechen, kann diese Problematik mit diesem Workaround behandelt werden.

Hinweis: Wenn nach Listing 7.18 die Modems gestartet werden, müssen sie nach einem Neustart wieder Konfiguriert werden.

OpenVPN Server / Client

Die Datei openvpn_mobile_access.tar.gz aus dem Abgabe-Ordner muss auf den Access-Node in den Ordner /etc/openvpn kopiert werden.

```
1  # OpenVPN installieren
2  sudo apt-get install openvpn
3  # Die Konfig. Datein nach Access Node kopieren
4  scp openvpn_mobile_access.tar.gz access:.
5  sudo cp openvpn_mobile_access.tar.gz /etc/openvpn/.
6  cd /etc/openvpn/
7  # Die Konfig auspacken
8  sudo tar -xvf openvpn_mobile_access.tar.gz
9  # OpenVPN Starten
10 sudo service openvpn start
```

Listing 7.19: OpenVPN Client Installation

Die Datei /etc/client2.conf muss noch mit einem Texteditor geöffnet werden und die „remote your-ip your-port“ Anweisung mit den Angaben des im Internet erreichbaren Exit Node ergänzt werden. Wenn z.B. die IP-Addresse des Exit Node 8.8.8.8 ist und der Port für den OpenVPN-Dienst 9999 ist, dann wäre der Eintrag:„remote 8.8.8.8 9999“.

SOCKS Server Redsocks

Die redsocks.conf Datei muss aus dem Abgabe-Ordner auf den Access Node kopiert werden.

```
1  # Redsocks installieren
2  sudo apt-get install redsocks
```

```
3 # Die Konfig. Datein nach Access Node kopieren
4 scp redsocks.conf redsocks_default.conf access:.
5 sudo cp redsocks.conf /etc/.
6 sudo cp redsocks_default.conf /etc/default/.
7 # redsocks Starten
8 sudo service redsocks start
```

Listing 7.20: Redsocks Installation

Die Datei /etc/redsocks.conf muss noch mit einem Texteditor geöffnet werden und die „ip“ und „port“ Anweisungen mit den Angaben des im Internet erreichbaren Exit Node ergänzt werden. Wenn z.B. die IP-Addresse des Exit Node 8.8.8.8 ist und der Port für den SOCKS-Dienst 5555 ist, dann wäre der Eintrag:„ip = 8.8.8.8 “ und „port = 5555“.

MPTCP-WiFi Route-Watcher

Der Route-Watcher überwacht den konfigurierten Default Gateway und prüft, ob weiterhin neue MPTCP-Verbindungen darüber gestartet werde können. Sind neue Datenübertragungen innerhalb eines Zeitrahmens (1000ms RTT) nicht mehr möglich, wird ein anderes Interface zum Default Gateway konfiguriert und wiederum getestet.

```
1  # Dateien auf Access-Node kopieren
2  scp routwatcher.tar.gz 10.0.0.163:.
3  # Dateien entpacken
4  tar -xvf routewatcher.tar.gz
5  # service kopieren
6  sudo cp routewatcher.service /usr/sbin/routewatcher
7  # init script kopieren
8  sudo cp routewatcher.init /etc/init.d/routewatcher
9  # ausfuehrbar machen
10 sudo chmod u+x /etc/init.d/routewatcher
11 sudo chmod u+x /usr/sbin/routewatcher
12 # init script aktivieren
13 sudo update-rc.d routewatcher defaults
14 # Dienst starten
```

```
15 sudo service routewatcher start
```

Listing 7.21: Route-Watcher Installation

8. Schlussfolgerungen

8.1. Umgesetzte Umgebung

Die umgesetzte Umgebung entspricht der Planung. Die Clients innerhalb des Fahrzeuges sind über WiFi mit dem Access Node des Fahrzeugs verbunden. Dieser leitet die Pakete der Clients über SOCKS (im Falle von TCP-Paketen) oder TCP-VPN (im Falle von Nicht-TCP-Paketen) an den Exit Node im Rechenzentrum. Auf dieser Strecke wird MPTCP eingesetzt. Der Exit Node leitet die Pakete weiter an das Ziel im Internet. Auf dem Rückweg wird der umgekehrte Weg auf den noch offenen Kanälen genutzt.

Auf diese Weise kann von den Vorteilen von MPTCP profitiert werden, allen voran die dynamische Aufteilung des Verkehrs auf die gerade vorhandenen Funkverbindungen des Access Nodes, obwohl die meisten Clients und Server zum Zeitpunkt dieser Arbeit kein MPTCP implementiert haben.

Die umgesetzte Umgebung besteht aus mehreren Komponenten:

Auf dem Access Node wurde ein SOCKS-Client (*Redsocks*) sowie ein VPN-Client (*OpenVPN* im TCP-Modus) konfiguriert. Über *iptables* werden die Pakete dynamisch auf SOCKS oder VPN aufgeteilt. Drei Applikationen behandeln die Interface-Dynamik: Beim Interface-Up-Ereignis werden die nötigen Einstellungen für MPTCP bestimmt und berechnet und eine Routing-Tabelle für dieses Interface angelegt. Beim Interface-Down-Ereignis werden die Einstellungen wieder entfernt. Ein Route-Watcher untersucht laufend, ob das momentane Default Interface das Internet erreichen kann und stellt gegebenenfalls ein neues Default Interface ein. Dabei wird das WiFi-Interface bevorzugt, sodass bei vorhandenem WiFi-Zugang kein Datenverkehr über Mobilfunk gesendet wird.

Auf dem Exit Node sind als Verbindungs-Gegenstelle ein SOCKS-Proxy (*dante*) sowie ein

VPN-Server konfiguriert. Ein NAT sorgt für die Umschreibung der internen IP-Adressen auf IP-Adressen, die im Internet geroutet werden können.

8.2. Umgesetzte Anforderungen

Alle in der Anforderungsspezifikation (siehe Kapitel 2 auf Seite 16) definierten Muss-Anforderungen, User Stories und Use Cases konnten umgesetzt werden. Dazu wurde die Analyse der Vorarbeit hinterfragt und fortgeführt sowie der Proof-of-Concept-Prototyp um Dynamik sowie diverse Komponenten erweitert und neu implementiert. Darüber hinaus konnte auch die Erweiterung, WiFi über Mobilfunk zu priorisieren, realisiert werden. Die durchgeführten Tests sind in Abschnitt 6.5 auf Seite 88 dokumentiert.

8.3. Angetroffene Herausforderungen und Erweiterungsmöglichkeiten

Nicht umgesetzt werden konnte die Einbeziehung der Fahrzeughardware (*IPmotion CAR-A-WAN.coach Plus Vorserie*) in den Prototyp, da die dort verwendeten Hardwarekomponenten den MPTCP-Linux-Kernel nicht unterstützen,[1] weil MPTCP in der Implementation der Université catholique de Louvain auf einen neueren Linux-Kernel aufbaut. In einer Folgearbeit könnte der Prototyp des Access Nodes auf eine neue Version der Fahrzeughardware portiert oder eine alternative Fahrzeughardware evaluiert werden.

MPTCP setzt für seine Optionen das TCP Options Kind 30 ein. Es existieren Firewalls, die ihnen unbekannte Optionen aus TCP-Paketen entfernen. So konnte in verschiedenen Umgebungen – der HSR und im Netz des Mobilfunk-Betreibers Orange – beobachtet werden, dass die MPTCP-Optionen entfernt wurden. Damit können Access Node und Exit Node keine MultiPath-Verbindung aushandeln, MPTCP kann in diesen Netzwerken nicht zum Einsatz kommen. Möglicherweise kann hier eine Lösung mit den Netzbetreibern ausgehandelt werden.

[1] Ergeben durch Abklärung mit dem Hersteller.

Weitere Erweiterungsmöglichkeiten sind:

- Implementation von Quality of Service unterschiedliche Behandlung des Datenverkehrs nach Audio, Video oder Daten.
- Dynamische Aufteilung der Daten je nach Qualität und Kosten der Mobilfunkverbindungen durch Implementierung eines eigenen Path-Managers als Teil des MPTCP-Kernels.
- Tests des Prototyps auf geeigneter Hardware in Fahrzeugen und auf Strecken von Postauto und Jungfraubahnen.
- Erarbeitung eines IP-Adresskonzepts für den Produktiveinsatz der in diesem Projekt erarbeiteten Technik.
- Implementierung der Produktivumgebung.
- Spätestens, wenn die MPTCP-Spezifikation nicht mehr als „Experimental“ gekennzeichnet ist: Prüfung, ob die Umgebung noch auf dem neusten Stand des MPTCP-Standards aufbaut.

Literaturverzeichnis

[1] Caspar, Nils/ Huber, Simon: „Multipath over Wireless Networks für mobiles WiFi“. Studienarbeit, HSR Hochschule für Technik Rapperswil, 2013.

[2] IETF: „Multipath TCP (mptcp)“. Sammlung von Standards über sowie im Umfeld von MPTCP. `http://datatracker.ietf.org/wg/mptcp/`, Abgerufen am 17. März 2014.

[3] Apple Inc.: „iOS: Multipath TCP Support in iOS 7“. `http://support.apple.com/kb/HT5977`, 27. Januar 2014, Abgerufen am 24. März 2014.

[4] Bonaventure, Olivier: „Apple seems to also believe in Multipath TCP“. `http://perso.uclouvain.be/olivier.bonaventure/blog/html/2013/09/18/mptcp.html`, 18. September 2013, Abgerufen am 27. März 2014.

[5] Ford, A. et al.: „TCP Extension for Multipath Operation with Multiple Addresses“. IETF, RFC6824, Januar 2013. Zu finden unter: `http://tools.ietf.org/html/rfc6824`, Abgerufen am 24. März 2014.

[6] Raiciu, A. et al.: „Coupled Congestion Control for Multipath Transport Protocols“. IETF, RFC6356, Oktober 2011. Zu finden unter: `http://tools.ietf.org/html/rfc6356`, Abgerufen am 31. März 2014.

[7] Paasch, Christoph: „Multipath TCP“. Vortragsaufzeichnung. `http://www.youtube.com/watch?v=VWN0ctPi5cw`, 26. April 2012, Abgerufen am 27. März 2014.

[8] Bonaventure, Olivier: „Decoupling TCP from IP with Multipath TCP“. Präsentationsfolien. `http://multipath-tcp.org/data/MultipathTCP-netsys.pdf`, März 2013, Abgerufen am 27. März 2014.

[9] IPmotion: CAR-A-WAN.automotive Plus, CAR-A-WAN.coach Plus. Manual. https://www.ipmotion.de/fileadmin/user_upload/doc/manuals/v4/Manual_CAR-A-WAN.automotive_v4_Revision_4.pdf, August 2013, Abgerufen am 23. März 2014.

[10] Meeting mit Michael Schneider, CloudGuard Software AG, Zürich, 7. März 2014.

[11] OpenSSH: Specifications implemented by OpenSSH. http://www.openssh.com/specs.html, November 2013, Abgerufen am 30. März 2014.

A. Aufgabenstellung bei Ausschreibung

(Originalzitat von Ausschreibung der Bachelorarbeit, Prof. B. Stettler und CloudGuard Software AG, 2013.)

A.1. Ausgangslage

Um die Reisezeit zu verkürzen und gegenüber dem Privatverkehr an Attraktivität zuzulegen, werden zur Zeit viele Fahrzeuge des öffentlichen Verkehrs mit WiFi ausgerüstet („Mobile WiFi“). Zwar verfügen Smartphones auch über 3G/4G Datenverbindungen, die Dämpfungseigenschaften von Zügen/Bussen führen aber zu einem stark reduzierten Datendurchsatz und zu einer erhöhten Strahlenbelastung der Passagiere. Zudem ist Mobile WiFi meist gratis und für viele Handy-Abos daher die attraktivere Variante (siehe auch `http://www.postauto.ch/pag-startseite/pag-ueberuns/pag-medien/post-archive/2013/pag-news-nat-postauto-gratis-wifi/pag-medienmitteilungen.htm`).

A.2. Problemstellung

Der im Fahrzeug generierte Internetverkehr wird zur Zeit über (nur) eine einzelne UMTS-Aussenantenne zu Mobilfunkprovidern abgeführt. Allerdings führen Funklöcher und überlastete Zellen dazu, dass das „Nutzererlebnis“ vielerorts beeinträchtigt wird. Es drängt sich daher auf, möglichst viele Handynetze parallel zu nutzen und den Verkehr dynamisch auf die verschiedenen Providernetze (z. B. Swisscom, Sunrise, Orange) aufzuteilen.

Um diese Vision in die Tat umzusetzen, müssen verschiedene technische Hürden genommen werden:

- Erkennen/Messen der aktuell verfügbaren Kapazität auf den verschiedenen Handy-Netzen
- Dynamisches Verteilen des Verkehrs, ohne die User-Anwendungen zu beeinträchtigen (wie durch TCP-Restarts)
- Aufrechterhalten von Tunnelverbindungen trotz wechselnder IP-Adressen und NAT

Eine Studienarbeit hat bereits einen Proof-of-Concept basierend auf Multipath-TCP (`http://datatracker.ietf.org/wg/mptcp/`) gelegt, einem neuen TCP-Standard, der automatisch alle verfügbaren Links verwendet.

A.3. Ziele

In dieser Folgearbeit geht es um die Neuerstellung eines Prototypen und um die Durchführung von konkreten Tests bei Postauto und den Junfraubahnen:

- Der vorhandene Prototyp soll erweitert werden, so dass die Verkehrsverteilung auf unterschiedlich gute oder mit unterschiedlichen Kosten verbundene Mobilfunkverbindungen dynamisch gesteuert wird.
- Die Implementation soll dahingehend erweitert werden, dass unterschiedliche Verkehrsströme (z. B. Audio, Video, Daten) individuell auf die verschiedenen Uplinks verteilt werden können.
- Die MPTCP-Linux-Implementation der Université catholique de Louvain (`http://mptcp.info.ucl.ac.be`) soll in die Fahrzeugplattformen integriert werden (heutiger Prototyp basiert noch auf Standard-Linux)

- Tests auf Bussen von Postauto und Zügen der Junfraubahnen sollen die Praxistauglichkeit beweisen.

B. Shell-Programme

B.1. mptcp_wifi_up.sh

```
1  #!/bin/sh
2  # ipcalc is needed (sudo apt-get install ipcalc)
3  # The ifconfig output used depends on MPTCP. If MPTCP isn't
       installed this script won't work.
4  # Copy this script into /etc/network/if-up.d/
5
6  set -e
7
8  ####
9  DEFAULT_GATEWAY_BLACKLIST1="eth3" # Ignore this interface for
       Default Gateway watcher
10 DEFAULT_GATEWAY_BLACKLIST2="eth999" # Ignore this interface
       for Default Gateway watcher
11 DEFAULT_GATEWAY_BLACKLIST3="eth999" # Ignore this interface
       for Default Gateway watcher
12 ####
13
14 if [ "$IFACE" = lo -o "$MODE" != start -o "$ADDRFAM" != inet
       ]; then
15         exit 0
16 fi
17
18 # make a table-alias
```

```
19 if [ `grep $IFACE /etc/iproute2/rt_tables | wc -l` -eq 0 ];
   then
20         num=`cat /etc/iproute2/rt_tables | wc -l`
21         echo "$num  $IFACE" >> /etc/iproute2/rt_tables
22 fi
23
24 # get route info
25 ip_address=`ifconfig $IFACE | grep -oP
   "\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b" | head -n 1`
26 netmask=`ifconfig $IFACE | grep -oP
   "\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b" | tail -n 2 | head
   -n 1`
27 network=`ipcalc -nb $ip_address/$netmask | grep -oP
   "\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/\d{1,2}\b"` #
   calculate network/mask
28 gateway_address=`ipcalc -nb $ip_address/$netmask | grep -oP
   "\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b" | head -n 5 | tail
   -n 1` # assumed as first ip address in network
29 if [ `ip route | grep -P "default via
   \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3} dev $IFACE" | wc -l` = 1
   ]; then
30     gateway_address=`ip route | grep -oP
   "\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}" | head -n 1` # main
   default gateway
31 fi
32
33 # configure route tables
34 ip route add $network dev $IFACE scope link table $IFACE
35 ip route add default via $gateway_address dev $IFACE table
   $IFACE
36 ip rule add from $ip_address table $IFACE
37
38 if [ "$IFACE" != "$DEFAULT_GATEWAY_BLACKLIST1" -a "$IFACE" !=
   "$DEFAULT_GATEWAY_BLACKLIST2" -a "$IFACE" !=
   "$DEFAULT_GATEWAY_BLACKLIST3" ]; then
39     mkdir -p "/tmp/mptcp-wifi/info"
```

```
40      touch /tmp/mptcp-wifi/info/$IFACE
41      echo "$ip_address\n$network\n$gateway_address" >
        /tmp/mptcp-wifi/info/$IFACE
42  fi
```

Listing B.1: mptcp_wifi_up

B.2. mptcp_wifi_down

```
1  #!/bin/sh
2  # Copy this script into /etc/network/if-down.d/
3
4  set -e
5
6  if [ "$IFACE" = lo -o "$MODE" != stop ]; then
7          exit 0
8  fi
9
10 ip rule del table $IFACE
11 ip route flush table $IFACE
12
13 rm /tmp/mptcp-wifi/info/$IFACE
```

Listing B.2: mptcp_wifi_down

B.3. mptcp_route_watcher

```
1  #!/bin/sh
2  # Start at boot on Access Node with elevated rights
3
4  ####
```

```
5  CHECK_IP_ADDRESS="10.0.2.2" # IP address to check reachability
6  VPN_SERVER_IP_ADDRESS="10.0.2.1" # IP address to connect to
       the VPN server
7  PRIORITY_INTERFACE="eth2" # if up, use only this interface
       (e.g. to use WiFi instead of LTE)
8  BACKUP_INTERFACE1="eth0" # backup interfaces will not be used
       if PRIORITY_INTERFACE is up (e.g. to not use LTE if WiFi is
       up)
9  BACKUP_INTERFACE2="eth1"
10 PING_TIMEOUT_SECONDS="1" # after how many seconds without icmp
       echo reply the interface will be considered unusable
11 ####
12
13 echo "Starting mptcp_route_watcher."
14
15 default_interface=`ip -4 route list 0/0 | cut -d ' ' -f 5` #
       get current default interface
16 current_default_interface="$default_interface"
17 if [ $default_interface ]; then
18    echo "Current default interface is $default_interface"
19 else # if there is currently no default interface use random
       interface name as starting value
20    default_interface="$PRIORITY_INTERFACE"
21    current_default_interface="not_$default_interface" # force
       interface change
22    echo "There is currently no default interface"
23 fi
24
25 ip_address=`cat /tmp/mptcp-wifi/info/$default_interface | tail
      -n 1`
26 while true; do
27
28    if [ -e /sys/class/net/$PRIORITY_INTERFACE/operstate ] && [
       `cat /sys/class/net/$PRIORITY_INTERFACE/operstate` = "up"
       ]; then
29       default_interface="$PRIORITY_INTERFACE"
```

```
30    else
31       if [ ! -e /sys/class/net/$default_interface/operstate ]
      || [ `cat /sys/class/net/$default_interface/operstate` !=
      "up" ] || ! ping -c 1 -w $PING_TIMEOUT_SECONDS -I
      $default_interface $CHECK_IP_ADDRESS > /dev/null; then
32          echo "Current default interface $default_interface
      doesn't work"
33          interfaces=`ls /tmp/mptcp-wifi/info/`
34          interfaces_length=`echo "$interfaces" | wc -w`
35          current_interface_id=`echo "$interfaces" | grep -n
      "$default_interface" | cut -f1 -d:`
36          interface_id=`expr $current_interface_id + 1`
37          if [ "$interface_id" -gt "$interfaces_length" ]; then
38             interface_id=1
39          fi
40          default_interface=`echo "$interfaces" | cut -d ' ' -f
      1 | head -n $interface_id | tail -n 1`
41          if [ ! $default_interface ]; then
42             echo "Whoops, looks like every interface is down.
      Doing nothing."
43             default_interface="$current_default_interface"
44          fi
45       fi
46    fi
47
48    if [ "$default_interface" != "$current_default_interface"
      ]; then
49       ip_address=`cat /tmp/mptcp-wifi/info/$default_interface
      | tail -n 1`
50       ip route del default
51       ip route add default via $ip_address dev
      $default_interface
52       ip route del $VPN_SERVER_IP_ADDRESS
53       ip route add $VPN_SERVER_IP_ADDRESS via $ip_address dev
      $default_interface
54       echo "Switched current interface to $default_interface"
```

```
55      if [ "$current_default_interface" =
    "$PRIORITY_INTERFACE" ]; then # if switching away from
    PRIORITY_INTERFACE
56          ip link set dev $BACKUP_INTERFACE1 multipath on
57          ip link set dev $BACKUP_INTERFACE2 multipath on
58          echo "Enabled multipath on $BACKUP_INTERFACE1 and
    $BACKUP_INTERFACE2"
59      else
60          if [ "$default_interface" = "$PRIORITY_INTERFACE" ];
    then # if switching to PRIORITY_INTERFACE
61              ip link set dev $BACKUP_INTERFACE1 multipath backup
62              ip link set dev $BACKUP_INTERFACE2 multipath backup
63              echo "Set $BACKUP_INTERFACE1 and
    $BACKUP_INTERFACE2 to multipath backup"
64          fi
65      fi
66      current_default_interface="$default_interface"
67      sleep 0.1s
68  else
69      sleep 1s
70  fi
71
72 done
```

Listing B.3: mptcp_route_watcher

C. Konfigurationen Access Node

C.1. Konfiguration iptables

```
1  #!/bin/bash
2  # Create new chain
3  iptables -t nat -N REDSOCKS
4
5  # Ignore LANs and some other reserved addresses.
6  iptables -t nat -A REDSOCKS -d 0.0.0.0/8 -j RETURN
7  iptables -t nat -A REDSOCKS -d 10.0.0.0/8 -j RETURN
8  iptables -t nat -A REDSOCKS -d 127.0.0.0/8 -j RETURN
9  iptables -t nat -A REDSOCKS -d 169.254.0.0/16 -j RETURN
10 iptables -t nat -A REDSOCKS -d 172.16.0.0/12 -j RETURN
11 iptables -t nat -A REDSOCKS -d 192.168.0.0/16 -j RETURN
12 iptables -t nat -A REDSOCKS -d 224.0.0.0/4 -j RETURN
13 iptables -t nat -A REDSOCKS -d 240.0.0.0/4 -j RETURN
14
15 # Anything else should be redirected to port 1085
16 iptables -t nat -A REDSOCKS -p tcp -j REDIRECT --to-ports 1085
17
18 # Any tcp connection made by 'cruiser' should be redirected.
       Works fror local user, redsocks local_ip must be localhost
19 #iptables -t nat -A OUTPUT -p tcp -m owner --uid-owner cruiser
       -j REDSOCKS
20 #iptables -t nat -A OUTPUT -p tcp -s 10.0.3.0/24 -j REDSOCKS
21 iptables -t nat -A PREROUTING -i eth2 -s 10.0.3.0/24 -p tcp -j
       REDSOCKS
```

Listing C.1: iptables_access.sh (für Access Node)

C.2. Konfiguration VPN-Client

```
client
dev tun
proto tcp
remote 10.0.2.1 1194
resolv-retry infinite
nobind
user nobody
group nogroup
persist-key
persist-tun
ca ca.crt
cert client1.crt
key client1.key
ns-cert-type server
comp-lzo
verb 3
```

Listing C.2: /etc/openvpn/client.conf (für Access Node)

C.3. Konfiguration RedSocks

```
base {
    log_debug = on;
    log_info = on;
    log = "syslog:daemon";
    daemon = off;
```

```
6
7       user = redsocks;
8       group = redsocks;
9
10      redirector = iptables;
11  }
12
13  redsocks {
14      local_ip = 0.0.0.0;
15      local_port = 1085;
16      ip = 10.0.2.1;
17      port = 1080;
18      type = socks5;
19  }
```

Listing C.3: redsocks.conf (für Access Node)

C.4. Konfiguration DHCP-Server

```
1  ddns-update-style none;
2  authoritative;
3  default-lease-time 600;
4  max-lease-time 7200;
5  log-facility local7;
6
7  subnet 10.0.3.0 netmask 255.255.255.0 {
8  interface eth2;
9  range 10.0.3.50 10.0.3.99;
10 option routers 10.0.3.1;
11 }
12
13 host vmclient {
14   hardware ethernet 00:0c:29:6b:1d:92;
15   fixed-address 10.0.3.50;
```

```
16 }
```

Listing C.4: dhcpd.conf (für Access Node)

D. Konfigurationen Exit Node

D.1. Konfiguration VPN-Server

```
1  local 10.0.2.1
2  port 1194
3  proto tcp
4  dev tun
5  ca ca.crt
6  cert server.crt
7  key server.key  # This file should be kept secret
8  dh dh2048.pem
9  server 10.0.1.0 255.255.255.0
10 ifconfig-pool-persist ipp.txt
11 client-config-dir ccd
12 route 10.0.3.0 255.255.255.0
13 push "redirect-gateway def1 bypass-dhcp"
14 client-to-client
15 push "route 10.0.3.0 255.255.255.0"
16 keepalive 10 120
17 comp-lzo
18 user nobody
19 group nogroup
20 persist-key
21 persist-tun
22 status openvpn-status.log
23 verb 3
```

Listing D.1: /etc/openvpn/server.conf (für Exit Node)

D.2. Konfiguration iptables

```
1 # Generated by iptables-save v1.4.18 on Fri May 16 16:43:01
      2014
2 *nat
3 :PREROUTING ACCEPT [2328:163961]
4 :INPUT ACCEPT [903:68249]
5 :OUTPUT ACCEPT [608:50366]
6 :POSTROUTING ACCEPT [1120:83662]
7 -A POSTROUTING -s 10.0.3.0/24 -j SNAT --to-source 10.0.0.163
8 COMMIT
9 # Completed on Fri May 16 16:43:01 2014
```

Listing D.2: iptables_exit.save (für Exit Node)

D.3. Konfiguration Dante

```
1  logoutput: stderr /var/log/danted
2  user.privileged: proxy
3  user.notprivileged: nobody
4  user.libwrap: nobody
5
6  internal: eth1 port = 1080
7  external: eth0
8  clientmethod: none
9  method: none
10
11 client pass {
12         from: 10.0.0.0/24 to: 0.0.0.0/0
```

```
13      log: error # connect disconnect
14  }
15  #allow connections form local network (10.0.1.0/24)
16  client pass {
17          from: 10.0.1.0/24 to: 0.0.0.0/0
18          log: error connect # connect disconnect
19  }
20  client pass {
21          from: 10.0.2.0/24 to: 0.0.0.0/0
22          log: error connect # connect disconnect
23  }
24  client pass {
25          from: 10.0.3.0/24 to: 0.0.0.0/0
26          log: error # connect disconnect
27  }
28  #allow connections from local network (192.168.1.0/24)
29  client pass {
30          from: 192.168.1.0/24 to: 0.0.0.0/0
31          log: error # connect disconnect
32  }
33
34  # server opration access rules
35  # allow the rest
36  pass {
37      from: 0.0.0.0/0 to: 0.0.0.0/0
38      command: bind connect udpassociate
39      log: error
40  }
41
42  pass {
43          from: 0.0.0.0/0 to: 0.0.0.0/0
44          command: bindreply udpreply
45          log: error # connect disconnect iooperation
46  }
```

Listing D.3: danted.conf (für Exit Node)