HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ins institute for networked solutions

WLAN Vernetzung einer Gondelbahn

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2016

Autor(en):	Etienne Georgy, Fabian Wirz
Betreuer:	Prof. Beat Stettler
Gegenleser:	Prof. Dr. Farhad Mehta
Experte:	Basile Bluntschli

Aufgabenstellung

Ausgangslage

Bergbahnen haben die Vorschrift, dass sie bei Störungen Durchsagen an die Passagiere machen können müssen. Dies wurde in der Vergangenheit meist über Lautsprecher an den Masten gelöst, was aber gerade bei Sturm nicht gut verstanden wird. Deshalb möchte man in Zukunft Lautsprecher in den Kabinen installieren, was die Möglichkeit mit sich bringt, auch gleich noch weitere Services anzubieten (Gegensprech, Informationsbildschirme, Internet-Access usw.)

Die Schwierigkeit bei der Vernetzung von Gondelbahnen sind:

- Zu grosse Distanzen für viele Funktechnologien wie WLAN zwischen den Gondeln und der Berg- oder Talstation. Das heisst z.B. für WLAN, dass ein Mesh-Netzwerk verwendet muss
- Keine fixe Reihenfolge der Gondeln, das heisst "Nachbarn" können nicht fix konfiguriert werden
- Ein dynamisches Mesh-Protokoll (wie OLSR) hat das Problem, dass die entgegenkommenden Gondeln viele Topologiewechsel verursachen, die nichts bringen.
- Keine ständige Stromversorgung, das heisst energiehungrige Lösungen funktionieren nicht.

Diese Studienarbeit beinhaltet die folgenden Arbeitsschritte:

- Analyse der Aufgabe sowie der damit verbundenen technischen Schwierigkeiten
- Evaluierung von Lösungs-Optionen mit Hilfe von verfügbaren Funktechnologien
- Implementierung eines Prototypen basierend auf Raspberry Pi's
- Durchführen von Tests
- Dokumentation der Resultate

Datum: 21.12.16

Unterschrift: 

Abstract

Viele Gondelbahnen benutzen heutzutage ein Lautsprechersystem ausserhalb der Gondeln, um die Passagiere über allfällige Störungen zu informieren. Diese Durchsagen sind dadurch oft schwer verständlich. Über die Übertragungstechnologie WLAN und die Mesh-Technologie 802.11s werden die Gondeln nun untereinander vernetzt, um Sprachsignale direkt in der Gondel abspielen zu können. In der Gondel ist dafür als Mesh-Knoten ein Raspberry Pi eingebaut, der die Sprachdurchsagen direkt über einen Lautsprecher ausgibt.

In einer ersten Phase wurden die verschiedenen Mesh-Protokolle analysiert und klassifiziert. Anhand der zuvor definierten Anforderungen fiel die Wahl auf ein reaktives Protokoll. Das ausgewählte Mesh-Protokoll 802.11s wurde danach mit Wireshark untersucht und mit verschiedenen Tests auf Funktionsfähigkeit überprüft. Nebst klassischen Failover-Tests wurde das Protokoll auf spezifische Gondelbahn-Szenarien überprüft. Das Resultat ergab, dass gerade das Kreuzen von zwei Gondeln ein Problem verursacht. In diesem Fall sehen sich die beiden Mesh-Knoten nur für eine kurze Zeit und ein Datenaustausch ist daher nicht gewünscht. Ein entwickeltes Skript blockiert diese Verbindung und baut nur zu Mesh-Knoten eine Verbindung auf, die länger sichtbar sind. Weiter wurde untersucht, wie sich ein Multicast-Strom und VoIP-Telefonie im Mesh-Netzwerk verhalten. Mit einer im Modell simulierten Gondelbahn wurde danach getestet, ob das gesamte System auch funktioniert.

Über das aufgebaute Mesh-Netzwerk können alle Knoten miteinander Daten austauschen. Die Sprachsignale werden in der Station auf die Gondeln übertragen und diese senden das Signal an ihre Nachbarknoten weiter, bis alle Gondeln erreicht wurden. Das verwendete Protokoll bietet eine gute Grundlage für den Aufbau eines solchen Netzwerkes; es musste aber mit zusätzlichen Skripten und Konfigurationen an die jeweilige Gondelbahn angepasst werden. Zudem müssen die verwendeten Applikationen dafür ausgelegt sein, Daten im Mesh-Netzwerk zu versenden.

Management Summary

Ausgangslage

Hat eine Gondelbahn in der heutigen Zeit einen Defekt, müssen die Passagiere schnell und verständlich per Sprachdurchsage informiert werden können. Gerade die Verständlichkeit ist aber ein grosses Problem. Die Passagiere in der Gondel werden heute meistens über Lautsprecher an den Masten informiert. Grosse Distanzen zu den Masten oder auch Wettereinflüsse können jedoch die Verständlichkeit dieser Sprachsignale stark beeinflussen. Ein Lautsprecher in jeder Gondel wäre die Lösung des Problems. Die Sprachinformationen müssen dadurch aber zwangsläufig in die Gondeln transportiert werden. Da die Daten nicht über ein Kabel übertragen werden können, gibt es nur noch die Möglichkeit, die Daten per Funk zu übertragen. Da das traditionelle WLAN mit einem Access Point als Sender nur eine gewisse Reichweite hat, könnte die gesamte Gondelbahn gar nicht von einem zentralen Punkt aus abgedeckt werden. Theoretisch könnten an den Seilbahnmasten weitere Access Points aufgebaut werden, die das WLAN-Signal an die Gondeln verteilen. Für die Realisierung einer solchen Lösung sind jedoch bauliche Massnahmen nötig. Zusätzlich dürfte die Distanz zwischen den Masten auch nicht zu gross sein. Gerade heute ist es aber das Ziel, eine Gondelbahn mit möglichst wenigen Masten zu bauen, um die Umwelt weniger zu belasten. Gondeln, die sich direkt miteinander verbinden, könnten dieses Problem ideal lösen. Die Datenpakete werden dabei von jeder Gondel empfangen und weitergeleitet, bis sie das Ziel erreichen. Diese Topologie wird Mesh-Netzwerk genannt.

Vorgehen / Technologien

In einem ersten Schritt wurden bestehende Gondelbahnen analysiert. Dadurch konnten unterschiedliche Situationen und Problemzonen definiert werden. Gerade der Gondelabstand und die Gondelbahngeschwindigkeit waren wichtige Indikatoren, um die verschiedenen Situationen besser einschätzen zu können. Zusätzlich wurden Anforderungen sowohl aus Kunden- wie auch technischer Sicht definiert. Als primäre Ziele wurden „Sprachdurchsage“ und „Audioübertragung“ definiert.

Anhand der festgestellten Problemzonen und definierten Anforderungen wurde eine theoretische Lösungsskizze erstellt. Mit Hilfe der theoretischen Lösungsskizze konnten die Anforderungen an das Mesh-Protokoll und den zu verwendenden Algorithmus besser formuliert werden.

Die Protokollvielfalt im Mesh-Netzwerk ist sehr gross; daher mussten die verschiedenen Protokolle klassifiziert und anhand von Eigenschaften geordnet werden. In kabellosen Mesh-Netzwerken sind die Gegebenheiten ausserdem anders als in normalen kabelgebundenen Netzwerken. Die zwei Hauptkategorien bilden dabei die reaktiven und die proaktiven Protokolle. Bei den reaktiven Protokollen wird ein Pfad zum Ziel im Netzwerk erst gesucht, wenn auch effektiv Daten gesendet werden müssen. Die proaktiven hingegen sind vergleichbar mit den klassischen kabelgebundenen Netzwerken und warten die Routing-Einträge zu ihren Zielen kontinuierlich.

Da im Gondelbahn-Netzwerk nicht kontinuierlich Daten übertragen werden müssen und auch der Energieverbrauch eine wichtige Rolle spielt, fiel der Entscheid auf das 802.11s Protokoll. Das 802.11s Protokoll ist im WLAN-Standard integriert und findet heute eine immer grössere Verwendung.

Nach dem Entscheid 802.11s zu verwenden, wurde ein Testlabor aufgebaut, um das Protokoll mittels Testszenarien hinsichtlich der festgestellten Problemzonen zu prüfen. Aufgrund der Testresultate wurde erkannt das insbesondere das Kreuzen von Gondeln ein Problem verursacht. Es wurde daher ein Skript entwickelt, welches dieses Problem zu lösen weiss.

Ergebnisse

Die verschiedenen Tests zeigten auf, dass mit dem Protokoll 802.11s das dynamische Gondelbahn-Netzwerk aufgebaut und betrieben werden kann. Das implementierte Skript behebt ausserdem das Problem mit den sich kreuzenden Gondeln und sorgt dabei für einen optimaleren Datenfluss im Mesh-Netzwerk.

Über das aufgebaute Mesh-Netzwerk können die Gondeln mit der Talstation Daten austauschen. Sprachdurchsagen können somit über das Netzwerk versendet und empfangen werden. Je nach Störungen im WLAN variiert jedoch die Verständlichkeit solcher Durchsagen. Mit zusätzlichen Optimierungsmassnahmen könnten sicher noch bessere Resultate erzielt werden.

Eine Gegensprechanlage kann über das Netzwerk ebenfalls betrieben werden. Die Qualität während den Tests war gut und die End-zu-End-Verzögerung gering. Im Notfall können somit Passagiere durchaus mit der Talstation Kontakt aufnehmen.

1 Inhaltsverzeichnis

I Technischer Bericht

2 Einleitung

2.1 Ausgangslage

Bergbahnen haben die Vorschrift, bei einer allfälligen Störung der Bahn, die Passagiere über Lautsprecher informieren zu können. Bisher wurde dies meistens mit Lautsprechern an den verschiedenen Seilmasten gelöst. Je nach Wetter und Distanz zu den Masten wurden die Durchsagen verzerrt und waren dadurch für die Passagiere nicht mehr verständlich. Ein Lautsprecher innerhalb einer Gondel könnte dieses Problem beheben.

Eine Möglichkeit wäre, die Gondeln mit Funkgeräten auszurüsten und über diese die Informationen zu verteilen. Dabei könnten aber lediglich Sprachsignale übertragen werden. Es gibt jedoch auch das Bedürfnis, die Gondeln mit anderen Informationsarten zu versorgen. Somit wäre es zum Beispiel auch möglich, interaktive Informationen auf Bildschirmen in der Gondelbahn anzuzeigen oder Internet-Zugriff über WLAN für die Passagiere anzubieten.

Die grundlegende Idee ist nun, verschiedene Informationen bei der Tal- bzw. Bergstation an eine Gondel zu übertragen. Die Gondel empfängt die Information und leitet sie an die weiteren Gondeln weiter. Am Schluss sollen alle Gondeln die ausgesendete Information erhalten.

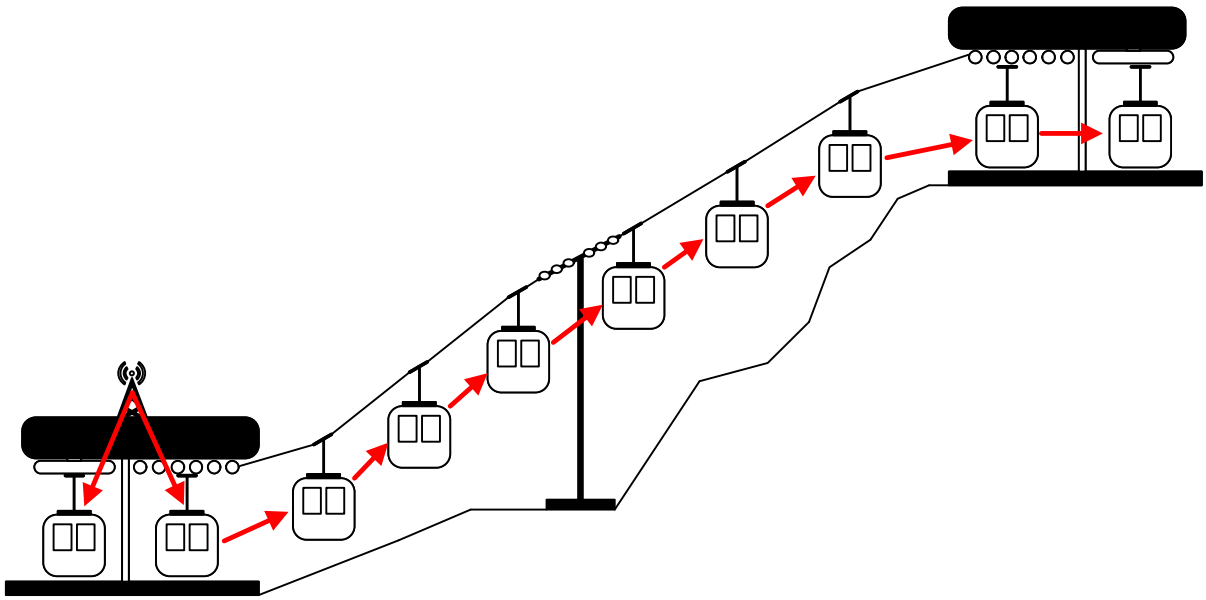


Abbildung 1: Big Picture Gondelbahn

2.2 Herausforderungen

Um die Informationen auf die vorgesehene Art von Gondel zu Gondel zu übertragen, wird ein Routing-Protokoll beziehungsweise ein Routing-Algorithmus benötigt, der die Informationen in einer solch dynamischen Umgebung korrekt weiterleiten kann. Die speziellen Herausforderungen an dieses Routing-Protokoll werden unter anderem in den technischen Anforderungen (Kapitel 3.3) und in den Problemzonen (Kapitel 4.4) beschrieben.

3 Analyse

3.1 Bestehende Gondelbahnen

Um die theoretischen Aspekte besser mit den realen Verhältnisse verknüpfen zu können, wurden mehrere Gondelbahnen auf diverse Eigenschaften untersucht. Die Gondelbahnen sind bewusst sehr unterschiedlich gewählt, damit die Problemstellung eine gewisse Breite bekommt.

Ein entscheidender Wert ist der Abstand zwischen zwei Gondeln. Dieser wird benötigt um herauszufinden, wie weit eine drahtlose Verbindung mindestens strahlen muss, damit noch eine Verbindung zwischen diesen möglich ist. Zusätzlich ist der Wert „Anzahl Gondeln" ein wichtiger Indikator dafür, wie viele Gondeln sich im gleichen Netzwerk befinden.

	Flumserberg Unterterzen - Oberterzen	**Flumserberg Oberterzen - Tannenbodenalp**	**Jungfrauregion Grindelwald-Holenstein**	**Jungfrauregion Holenstein-Männlichen**	**Lenzerheide Scharmoin**	**London Emirates Air Line**
Länge in Meter	1198	2231	3167	3073	1331	1103
Höhendifferenz in Meter	233	721	682	598	414	70
Anzahl Stützen	8	17	30	26	8	3
Stationshöhe Talstation in Meter	-	-	942	1624	1496	-
Stationshöhe Bergstation in Meter	-	-	1624	2222	1910	-
Anzahl Gondeln	23	42	230	230	42	34
Kapazität der Gondeln	8	8	4	4		10
Max. Fahrgeschwindigkeit	5 m/s	5 m/s	4 m/s	4 m/s	6 m/s	6 m/s
Durchschnittlicher Gondelabstand	104.17	194.00	275.39	267.22	115.74	95.91

Tabelle 1: Existierende Gondelbahnen und ihre Spezifikationen[1]

Da die Werte zum Teil sehr variieren, ist es schwierig, im späteren Verlauf der Arbeit einen Prototyp zu realisieren, der alle Werte abdeckt. Deshalb wird während der Arbeit von folgenden Werten ausgegangen:

- Für die theoretische Analyse des Problems wird ein Gondelabstand von 50 – 100 Meter angenommen.
- Es wird zudem mit einer Gondelanzahl von 40 Gondeln gerechnet. Dies entspricht etwa dem Wert einer normalen Gondelbahn.
- Die Fahrgeschwindigkeit beträgt 5 m/s.

[1] Alle Spezifikationsgrössen stammen von der Webseite http://www.seilbahntechnik.net

3.2 Anforderungen aus Kundensicht

Aufgrund der definierten Aufgabenstellung ergeben sich folgende Anforderungen aus Kundensicht.

Nummer	Anforderung
A01	Alle fahrenden Gondeln müssen erreicht werden können.
A02	Es muss die Möglichkeit bestehen, Sprachdurchsagen an die Gondeln zu übertragen.
A03	Es muss eine energieeffiziente Lösung sein, da keine permanente Stromversorgung vorhanden ist.
A04	Es muss die Möglichkeit offengehalten werden, andere Services, wie zum Beispiel Internet-Zugang, Gegensprechanlage und Informationsbildschirme anbieten zu können.

Tabelle 2: Kundenanforderungen

3.3 Technische Anforderungen

Aus den kundenspezifischen Anforderungen wurden die Technischen Anforderungen formuliert.

Nummer	Anforderung
T01	Das drahtlose Signal muss 100 Meter weit senden können, um zwei Gondeln miteinander zu verbinden. Je weiter die Gondeln auseinanderliegen, desto schlechter wird die Bandbreite und somit der Durchsatz im gesamten Netzwerk.
T02	Der Routing-Algorithmus sollte nicht mehr als 0.5 Sekunden benötigen, um die Routingtabelle zu erstellen. Dauert die Erstellung der Tabelle zu lange, kann sich das Netzwerk nie stabilisieren.
T03	Der Routing-Algorithmus sollte mit mindestens 40 Knoten umgehen können. Eine durchschnittliche Gondelbahn hat etwa 40 Gondeln und somit auch 40 Knoten.
T04	Die Bandbreite sollte nicht kleiner als 1 Mbit/s sein. Zur Übermittlung einer Sprachnachricht würden zwar bereits 128 Kbit/s reichen, jedoch möchte man noch andere Dienste anbieten und daher wird eine höhere Bandbreite benötigt.
T05	Der Empfänger in der Gondel soll die empfangenen Daten auch verarbeiten können, das heisst Sprachnachrichten ausgeben und Textnachrichten anzeigen.
T07	Routing-Tabellen sollten nur, wenn nötig geändert werden. Jede Änderung macht das Netzwerk langsamer. Zusätzlich könnten Applikationen auf dem OSI Layer 7 Probleme mit vielen Netzwerkwechseln bekunden.

Tabelle 3: Technische Anforderungen

Erläuterung zu Nummer T02:

Dieser Wert ist vor allem für Routing-Protokolle wichtig, die nicht angepasst werden können. Der nächste Hop von Gondel 1 wird etwa 25 Meter lang die Gondel auf der gegenüber liegenden Seite sein. Der Unterschied, dass die Distanz zur Gondel schräg ist und nicht gerade, kann vernachlässigt werden.

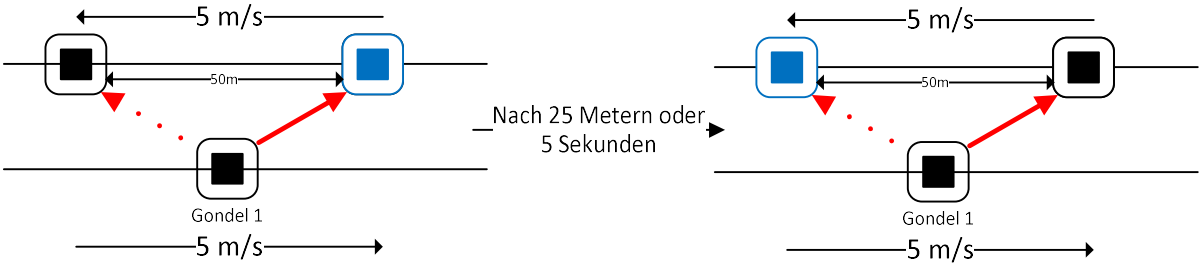


Abbildung 2: Kreuzende Gondeln

Anhand dieser Daten kann ausgesagt werden, dass alle fünf Sekunden jede Gondel einen neuen Nachbarn bekommt. Damit auch Daten übertragen werden können und nicht immer nur die Routing-Tabellen angepasst werden, soll die Konvergenzzeit weniger als 10% betragen. Dieser Prozentwert ist aber lediglich eine sinnvolle Annahme und sollte nicht als harte Grenze angeschaut werden.

3.4 Situationen

Durch die Analyse wurden verschiedene Situationen eruiert. In diesen Situationen muss das Netzwerk korrekt funktionieren. Eine mögliche Lösung muss die folgenden Situationen beachten.

3.4.1 Gondelbahn fährt

Dies ist der normale Fall, alle Gondeln sind in Bewegung. Die Daten bzw. Signale muss jede Gondel empfangen können und danach weitersenden. Bewegungen im System führen aber unweigerlich zu Topologie-Änderungen. Kreuzt eine Gondel, die sich auf der Fahrt nach oben befindet, mit einer Gondel, die sich auf der Fahrt nach unten befindet, kann es zu Problemen kommen. Für kurze Zeit ist diese Gondel der ideale Verbindungspartner. Häufige Topologie-Wechsel sind aber nicht gut für ein Netzwerk. Die Gondeln sollten sich daher nur immer mit jenen verbinden, welche sich in gleicher Richtung bewegen.

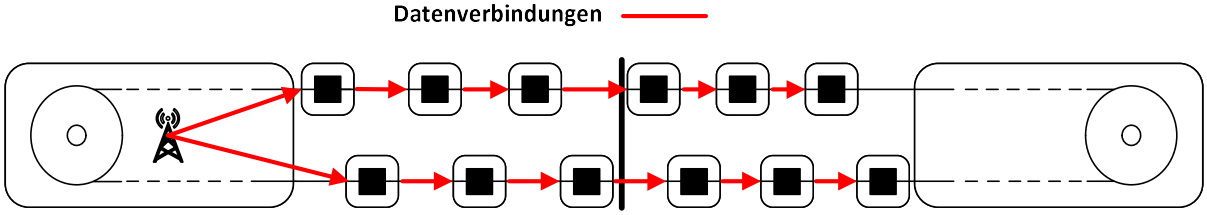


Abbildung 3: Idealfall - Alle Gondeln in einer Richtung sind miteinander verbunden

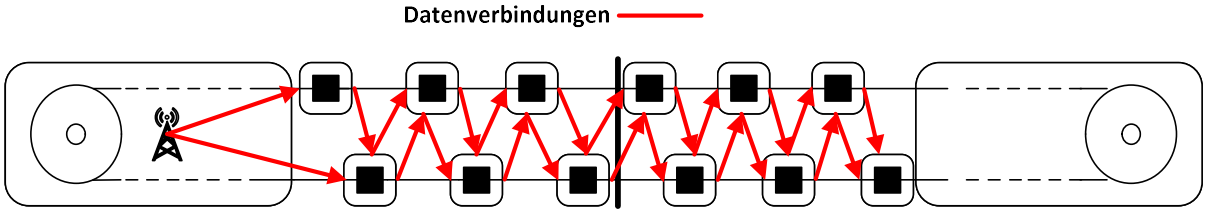


Abbildung 4: Kein Idealfall - Daten werden überkreuzt übertragen

3.4.2 Gondelbahn steht

Gerade wenn die Gondelbahn stillsteht, ist es wichtig, die Passagiere informieren zu können. Ein funktionierendes Netzwerk ist deshalb Pflicht. Bei einem Stillstand werden sich die Gondeln aber nicht mehr nur kurz kreuzen, sondern nebeneinander stehen bleiben. Es kann nicht mehr unterschieden werden, ob die Gondeln in die gleiche oder in die Gegenrichtung fahren. Die Hop-Anzahl wird dadurch grösser, bis alle Gondeln erreicht wurden. Die Topologie sollte aber trotzdem stabil bleiben und die Informationen sollten alle Passagiere erreichen.

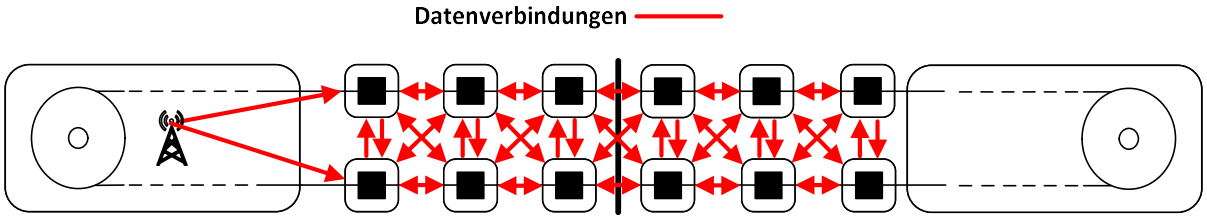


Abbildung 5: Gondelbahn steht still

3.4.3 Gondel in der Station

Kommen die Gondeln in der Tal- bzw. Bergstation an, werden sie automatisch abgebremst und fahren langsamer, damit die Passagiere ein- und aussteigen können. In der Station befinden sich dadurch zwangsläufig immer mehrere Gondeln, die sich über eine längere Zeit sehen. Die Topologie sollte auch in dieser Situation stabil bleiben. Wichtig ist, dass sich alle wegfahrenden bzw. ankommenden Gondeln nahtlos ins Netzwerk einfügen.

3.4.4 Gondel im Depot

Herrscht schlechtes Wetter oder ist gerade Nebensaison, wird die Gondelbahn weniger stark frequentiert. Es ist deswegen auch nicht nötig, alle Gondeln fahren zu lassen. Mehrere Gondeln könnten sich somit noch im Depot befinden. Die Gondeln im Depot sollten aber die fahrenden Gondeln nicht stören.

4 Anforderungsspezifikation

Die Anforderungsspezifikation soll anhand eines Szenarios aufzeigen, wie ein geeignetes Routing- Protokoll bzw. ein geeigneter Routing-Algorithmus auszusehen hat, um mit einem solchen dynamischen Netzwerk klar zu kommen.

4.1 Szenario Mesh-WLAN-Network

Alle Knoten der Gondelbahn sind immer in Bewegung, jedoch sind alle Bewegungen vorhersagbar. Wie lange das Kreuzen dauert oder wie lange eine Gondelbahn in einer Station verweilt, ist fest definiert. Die Knoten an sich sind dynamisch, aber das System als Ganzes bleibt immer gleich. Unter normalen Umständen hat eine Gondel immer den gleichen Vorgänger und den gleichen Nachfolger. Lediglich der Knoten an der Berg- bzw. Talstation bleibt an seinem Standort und sieht ständig andere Knoten. Jedoch gelten diese Bedingungen nur für einen Tag, am nächsten Tag können die Gondeln andere Vorgänger und Nachfolger haben. Daher kann nichts fest in den Gondeln programmiert werden.

4.2 Abgrenzungen

Um eine mögliche Lösungsskizze zu entwerfen, wird von einer optimalen Gondelbahn ausgegangen. Diese Gondelbahn enthält folgende Eigenschaften:

- Es wird angenommen, dass immer gleich viele Gondeln auf der Strecke sind. Somit ist gewährleistet, dass sich der Abstand zwischen zwei Gondeln nicht verändert.
- Die Anzahl Gondeln auf der Bahn beträgt 40.
- Der Abstand zwischen zwei Gondeln misst zwischen 50 und 100 Meter.
- Durchschnittlich braucht eine Gondel 30 Sekunden für die Durchfahrt einer Station.
- Alle Gondeln funktionieren zu jeder Zeit, es gibt keine Ausfälle.
- Eine Gondel hat zu jeder Zeit Funkkontakt zu ihrem Vorgänger und ihrem Nachfolger.
- Der Sender befindet sich in der Talstation.

4.3 Analyse

Aufgrund der definierten Abgrenzungen lässt sich eine Analyse erstellen, die es vereinfachen soll, einen geeigneten Algorithmus zu finden.

- Eine Gondel sieht während 30 Sekunden den Knoten, welcher Zugang zum Internet hat.
- Eine Gondel sieht während fünf Sekunden eine entgegen kommende Gondel.
- Eine Gondel sieht während der gesamten Zeit ihren Vorgänger und ihren Nachfolger.

4.4 Problemzonen

Das gesamte Problem lässt sich aufgrund der Situationsbeschreibungen aus Kapitel 3.4 und der Szenario-Analyse aus Kapitel 4.3 auf vier Problemzonen eingrenzen. Ein geeigneter Routing-Algorithmus muss im Stande sein, alle diese Problemzonen abzudecken. Dadurch, dass jedoch jede einzelne Problemzone ihre eigenen Anforderungen hat, ist es schwierig, einen globalen Algorithmus zu finden, der zu jedem Fall passt.

4.4.1 Problemzone 1: Gondel ist unterwegs in der Luft

Diese Problemzone stellt die gewöhnliche Situation dar. Eine Gondel fährt auf der Seilbahnstrecke hinauf- bzw. hinunter und kreuzt dabei andere Gondeln.

Problem	Anforderung
Kreuzende Gondeln	• Das Routing darf nicht über die entgegenkommenden Gondeln führen, da die Verbindung zu diesen schnell wieder abbricht. • Routing-Tabelle innerhalb einer Gondel darf als Next Hop nicht eine kreuzende Gondel enthalten.

Tabelle 4: Problembeschrieb kreuzende Gondeln

4.4.2 Problemzone 2: Gondel befindet sich in Durchfahrt Talstation

Diese Problemzone beschreibt den Fall, dass eine Gondel die Talstation durchfährt.

Problem	Anforderung
Stationärer Zugangspunkt	• Das Routing zu einem Knoten ausserhalb des Mesh-Netzwerkes muss zwangsweise über den stationären Zugangspunkt geschehen. • Die Routing-Tabelle des stationären Zugangspunktes muss dauernd den Next Hop für ein bestimmtes Ziel ändern, da die Gondeln nach einer gewissen Zeit die Stationen wieder verlassen und für den Zugangspunkt daher nicht mehr sichtbar sind.
Viele Gondeln, die sich über längere Zeit sehen	• Das Routing soll immer über die beste Verbindung geschehen.

Tabelle 5: Problembeschrieb Talstation

4.4.3 Problemzone 3: Gondel befindet sich in Durchfahrt Bergstation

Diese Problemzone definiert die gleichen Probleme wie bei der Durchfahrt einer Gondel durch die Talstation, mit Ausnahme, dass hier kein stationärer Zugangspunkt vorhanden ist.

4.4.4 Problemzone 4: Gondelbahn steht still

Diese Problemzone definiert den optimalen Fall eines Mesh-Routings, wo sich alle Knoten stationär an einem bestimmten Ort aufhalten.

Problem	Anforderung
Kreuzende Gondeln sind über längere Zeit sichtbar	• Das Routing soll über die schnellste und zuverlässigste Verbindung im Mesh-Netzwerk führen.

Tabelle 6: Problembeschrieb stehende Gondelbahn

4.5 Theoretische Lösungsskizze

Im folgenden Abschnitt wird eine theoretische Lösung beschrieben und überprüft, ob sie auch mit den vorgängig definierten Problemzonen umgehen kann. Diese Lösung ist unabhängig von den vorhandenen Technologien und soll aufzeigen, ob alle vorher definierten Problemzonen überhaupt behoben werden können. Zur allgemeinen Verständlichkeit wird die Gondel in der Lösungsskizze nur noch als Knoten bezeichnet.

4.5.1 Algorithmus-Vorschlag:

Ein Knoten hat nur eine bestimmte Reichweite. Innerhalb dieser Reichweite kann er alle anderen Knoten sehen. Je länger ein Knoten permanent in dieser Reichweite ist, desto vertrauenswürdiger ist die Verbindung.

Ablauf

1. Ein Knoten sucht in seiner Umgebung nach anderen Knoten.
2. Knoten, die nur kurz sichtbar sind, werden nicht beachtet. Dies sind kreuzende Knoten. Je nach Bahngeschwindigkeit und Gondelabstand sollten unterschiedliche Werte gewählt werden.
3. Alle Knoten die länger sichtbar sind, sollen mit einer Metrik bewertet werden. Die Metrik soll die Bandbreite und den Paketverlust beachten.
4. Die Knoten werden zusammen mit der Metrik in einer Tabelle eingetragen.
5. Diese Tabelle wird an alle Nachbarknoten im Netzwerk weitergeleitet.
6. Die Nachbarknoten berechnen anhand dieser Informationen ihre eigene Routing-Tabelle.
7. Am Ende haben alle Knoten eine Routing-Tabelle, in der alle anderen Knoten aufgelistet sind. Der Knoten sieht anhand dieser Tabelle, wohin er ein Paket senden muss, um einen anderen Knoten zu erreichen.

Ergänzung zu Punkt 2: Beträgt der Gondelabstand 50 Meter und die Bahngeschwindigkeit 5 m/s, dann sollte der Knoten alle anderen Knoten, die weniger lang als fünf Sekunden sichtbar sind, nicht beachten. Die Zeit darf aber auch nicht zu grosszügig gewählt werden, da sonst der Knoten den einzigen statischen Knoten in der Berg- bzw. Talstation nicht als gültig ansieht.

4.5.2 Beurteilung des Vorschlags

Das Netzwerk braucht eine gewisse Aufwärmphase. Sobald alle Knoten ihre Vorgänger und Nachfolger kennen und die Informationen ausgetauscht wurden, bleibt der Rechenaufwand gering. Der definierte Algorithmus würde in einem geschlossenen System mit nur bewegenden Gondeln perfekt funktionieren. Nur wurde bereits in der Problemzone 2 erwähnt, dass es auch Knoten gibt, die fest installiert sind und sich nicht innerhalb dieses Systems bewegen. Verlässt eine Gondel zum Beispiel eine Station, wird die Routing-Tabelle neu berechnet. Da es Änderungen gegeben hat, werden diese Informationen an die Nachbarknoten weitergeleitet. Auch diese Knoten berechnen die Tabelle neu und leiten die neuen Informationen weiter. Letztlich müssen alle Knoten die Tabelle neu berechnen, obwohl nur eine Gondel die Station verlassen hat. Wenn dies alle fünf Sekunden passiert, ist der Aufwand für ein sauberes Netz relativ hoch. Ein Verfahren, dass die Tabelle bereits vor dem ersten Versenden berechnet, macht somit wenig Sinn.

4.5.3 Alternative

Der vorgeschlagene Algorithmus kann die Problemzonen nicht abdecken. Es fliessen zu viele Kontrollpakete, obwohl vielleicht gar keine Daten fliessen. Es braucht ein Verfahren, welches die Pfade nur berechnet, wenn sie gebraucht werden.

Ablauf

1. Sobald ein Knoten Daten zu einem anderen Knoten senden will, sucht er den Pfad im Netzwerk.
2. Eine Meldung vom Sender geht durch das Netzwerk, dass er den Pfad zum Empfänger sucht.
3. Alle Knoten leiten diese Meldung weiter, bis der Empfängerknoten gefunden wurde.
4. Der Empfängerknoten sendet die Meldung zum Sender zurück.
5. Der Sender kennt nun den Pfad zum Empfänger und kann die effektiven Nutzdaten senden.

Im Gegensatz zum ersten Vorschlag werden in dieser Lösung nicht ständig Kontrollpakete ausgetauscht. Es gibt keine Routing-Tabelle, die ständig aktuell gehalten werden muss. Somit kann die Tabelle auch nicht ändern, wenn eine Gondel aus der Station rausfährt. Insbesondere diese Problemzone wird durch dieses Verfahren besser abgedeckt.

5 Evaluation & Design

Eine Möglichkeit, die theoretische Lösungsskizze umzusetzen, wäre es, ein Mesh-Protokoll einzusetzen. Ein Mesh-Protokoll regelt den Datenverkehr zwischen verschiedenen Netzwerkknoten, welche über WLAN miteinander verbunden sind. Man spricht hierbei auch von einem Mesh-Netzwerk.

Die Grundlagen solcher Mesh-Netzwerke und die Herausforderungen an ein Mesh-Protokolls hinsichtlich des Gondelbahnproblems sollen in diesem Kapitel erläutert werden. Ausserdem werden verschiedene Mesh-Protokolle evaluiert und schlussendlich eine Entscheidung getroffen, welches Protokoll sich für das Gondelbahnproblem am besten eignet.

5.1 Mesh-Grundlagen

Ein Mesh-Netzwerk zeichnet sich dadurch aus, dass der Datenfluss zwischen zwei Clients durch ein Netzwerk von Netzwerkknoten zu Netzwerkknoten fliesst und nicht wie im traditionellen WLAN Netzwerk von einem zentralen Netzwerkknoten (Access Point) zum Client. Das Mesh-Protokoll übernimmt hierbei die gesamte Routing Logik, um den besten Pfad durch das Netzwerk zu finden und zu warten.

5.1.1 Mobile Ad-Hoc Network (MANet)

Das MANet ist eine Unterkategorie von Mesh-Netzwerken und bezeichnet ein Ad-Hoc Netzwerk, bei dem sich die einzelnen Netzwerkknoten untereinander selbst organisieren. Diese sind nicht fix an einen Standort gebunden und können sich in ihrer Umgebung frei bewegen.

Um eine Verbindung zwischen zwei Knoten aufzubauen, wird der normale Ad-Hoc-Modus verwendet. Eine Ad-Hoc Verbindung ist jedoch immer Peer-2-Peer und nicht Multi-Hop fähig. Daher muss ein Routing-Protokoll die Logik übernehmen, um Knoten miteinander vernetzen zu können, die nur indirekt miteinander verbunden sind. Es werden dynamische Routing-Verfahren wie OLSR oder AODV benötigt.

5.1.2 Layer 2 vs. Layer 3

Beim Routing in Mesh-Netzwerken wird zwischen Layer 2 und Layer 3 Routing unterschieden. Die meisten Mesh-Routing-Protokolle arbeiten auf Layer 3. Sie tauschen sich gegenseitig Informationen mittels UDP Datenpakete aus und befüllen normale Kernel Routing Tabellen. Layer 2 Mesh-Protokolle hingegen benützen Ethernet-Frames um ein Routing zu ermöglichen. Sie benötigen dafür eine eigene Art Routing-Tabelle. Man spricht dabei oft von einer sogenannten Forwarding Table.

Ein Layer 2 Protokoll besitzt einige Vorteile gegenüber einem Layer 3 Protokoll. Unteranderem besitzt es eine bessere Performance, da der gesamte Overhead eines Layer 3 Protokolls entfällt. Ein weiterer Vorteil von Layer 2 Routing ist, dass danach auf dem Layer 3 unterschiedliche Protokolle einsetzbar sind. Es ist zum Beispiel möglich, IPv6 zu verwenden. Beim Layer 3 Routing kann nur jenes Protokoll verwendet werden, welches auch im Routing verwendet wird. Durch eine Layer 2 Implementation kann aber eine sehr grosse Broadcast Domäne entstehen, was nicht wirklich ein Vorteil ist.

5.2 Routing-Grundlagen

Sollen Daten vom einen Netzwerkgerät zum nächsten gesendet werden, muss der bestmögliche Pfad bekannt sein. Da in einem Ad-hoc-Netzwerk das Routing viel dynamischer ist als bei einem klassischen Netzwerk, braucht es neue Routing-Strategien. Die traditionellen Routing-Verfahren wie Link State oder Distance Vector müssen erweitert werden.

5.2.1 Proaktiv

Beim proaktiven Verfahren wird der Pfad schon ermittelt, bevor erste Nutzdaten übertragen werden. Das heisst, es werden ständig Routinginformationen zwischen den Knoten ausgetauscht. Dies belastet das Netzwerk, aber der Pfad ist dadurch bereits bekannt, wenn allfällige Nutzdaten übertragen werden wollen. Es kann jedoch sein, dass der Pfad keine Gültigkeit mehr besitzt und somit die Nutzdaten trotzdem nicht beim Ziel ankommen. Die Routing-Informationen werden in einer Routing-Tabelle gespeichert. Diese muss ständig aktuell sein, was insbesondere bei grösseren Netzwerken schwierig wird.

5.2.2 Reaktiv

Beim reaktiven Verfahren wird der Pfad zu einem Knoten erst ermittelt, wenn auch wirklich Nutzdaten übertragen werden sollen. Das bedeutet, die Routing-Tabellen werden erst aufgebaut, sobald ein Knoten Daten zu einem anderen Knoten senden will. Ansonsten werden im Netzwerk keine Routing-Control-Pakete versendet. Das Netzwerk wird dadurch weniger belastet, aber das Senden von Nutzdaten dauert länger, da noch zuerst der Pfad zum Ziel ermittelt werden muss.

5.2.3 Weitere Möglichkeiten

Wenn reaktive sowie proaktive Verfahren gemeinsam verwendet werden, nennt man das ein hybrides Verfahren. Dies wird vor allem verwendet, wenn das Netzwerk sehr gross ist und in Zonen aufgeteilt werden kann. In einer lokalen Zone könnte immer noch ein proaktives Verfahren verwendet werden, während im ganzen Netzwerk das reaktive Verfahren besser geeignet ist. Ein Beispiel für ein solches Protokoll ist das Zone-Routing-Protokoll (ZRP).

Während die proaktiven und reaktiven Routing-Verfahren als flaches Routing bezeichnet werden, gibt es noch das hierarchische Routing. Beim hierarchischen Routing werden die Netzwerkteilnehmer in Regionen eingeteilt. Innerhalb einer Region kennt jedes Gerät das andere. Netzwerkgeräte in anderen Regionen werden zu einem Eintrag zusammengefasst. Dadurch entsteht eine Hierarchie. Die Netzwerkgeräte unter sich kennen einander nicht mehr. Ein Beispiel für ein solches Protokoll ist das Cluster Based Routing-Protokoll (CBRP). Diese Klassifizierungen sind im Dokument [1] genauer beschrieben.

Während proaktiv und reaktiv topologiebasierte Routingverfahren sind, gibt es noch die positionsbasierten Routingverfahren. Anhand von Standortinformationen, abgefragt zum Beispiel mit einem GPS-Empfänger, kann auch der beste Pfad zwischen zwei Knoten ermittelt werden. Das Location Aided Routing (LAR) ist ein solches Protokoll.

5.2.4 Überblick

Wie Abbildung 6 zeigt, lassen sich die verschiedenen Routing-Protokolle klassifizieren. Die wichtigsten Routing Protokolle werden in diesem Kapitel noch genauer beschrieben.

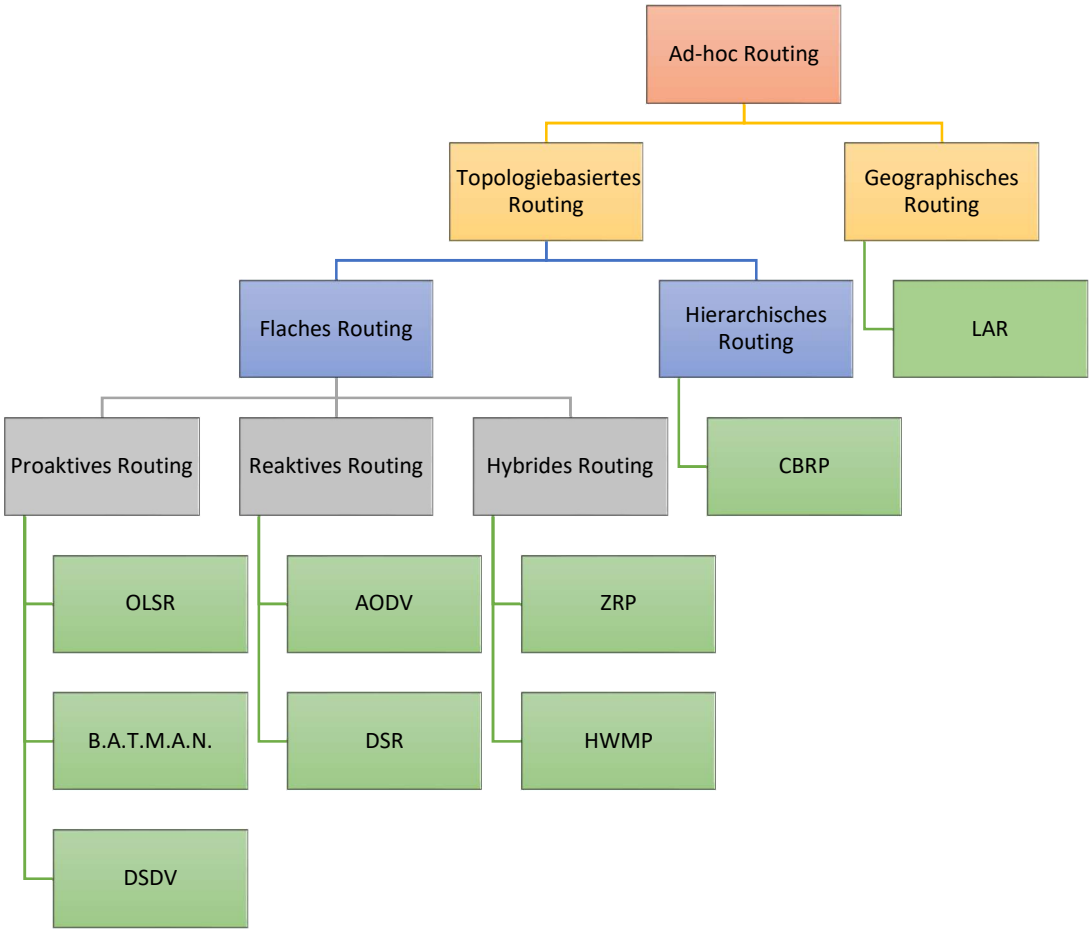


Abbildung 6: Übersicht Ad-Hoc-Protokolle

5.3 Entscheidungsmerkmale

Damit die Routing-Verfahren miteinander verglichen werden können, wurden sie auf verschiedene Entscheidungskriterien untersucht.

5.3.1 Skalierbarkeit

Jede Gondelbahn hat eine unterschiedliche Anzahl an Gondeln im Umlauf. Zudem müssen nicht immer gleich viele Gondeln im Umlauf sein. Das Routing-Protokoll sollte sich bei unterschiedlicher Knotenanzahl zuverlässig verhalten.

5.3.2 Metriken

Die Berechnung der Metrik in einem Mesh-Netzwerk ist ein wichtiger Bestandteil, um den besten Weg im Netzwerk zu finden. In einem drahtlosen Netzwerk gibt es jedoch viel mehr Einflüsse als in einem drahtgebundenen Netzwerk. Insbesondere externe Faktoren wie Wetter und örtliche Begebenheiten müssen beachtet werden. Die vorgestellten Metriken und ihre Eigenschaften sind in dieser Literatur [2] genauer beschrieben.

5.3.2.1 Minimal Hop Count Metrik

Diese Angabe bestimmt, über wie viele Knoten ein Paket gehen muss, damit es das Ziel erreicht. Keinen Einfluss auf diese Metrik hat die Bandbreite oder der Paketverlust. Es besteht die Gefahr, dass der kürzeste, aber nicht der schnellste und performanteste Weg als primärer Weg gewählt wird.

5.3.2.2 Airtime Link Metrik

Die Airtime Link Metrik (ALM) [3, pp. 1381-1382] berechnet den Kanal-Ressourcen-Verbrauch. Für die Berechnung werden unter anderem die Paket-Verlust-Rate sowie die Bandbreite benötigt. Die Formel zur Berechnung der Airtime Link Metrik ist wie folgt definiert:

$$c_a = \left[O_{ca} + O_p + \frac{B_t}{r}\right] * \frac{1}{1 - e_{fr}}$$

Formel 1: Berechnung Airtime Link Metrik

Der Channel Access Overhead c_{ca}, der MAC Protocol Overhead O_p sowie die Anzahl Bits in einem Testframe B_t sind Konstanten. Ihre Werte basieren auf der eingesetzten Übertragungstechnologie (b/g/n). Die Transmission Bit Rate r, angegeben in Mbit/s, ist die Rate, mit welcher ein Mesh Point ein Testframe der Grösse B_t und der Frame Error Rate e_{fr}, zu den derzeit gegebenen äusserlichen Umständen, übermittelt.

5.3.2.3 Expected Transmission Count Metrik

Die Expected Transmission Count Metrik (ETX) misst die Qualität zwischen zwei Knoten. Als Wert wird dabei angegeben, wie viele Übertragungsversuche nötig sind, damit das Paket ankommt. Dafür werden zwischen den Knoten regelmässig Pakete ausgetauscht. Eine Metrik von 1 bedeutet, dass auf dem Link alle Pakete ankommen. Bei einer Metrik von 2 würde nur noch die Hälfte aller Pakete ankommen. Die Berechnung erfolgt mit folgender Formel:

$$ETX = \frac{1}{Df * Dr}$$

Formel 2: Berechnung ETX Metrik

Df: Forward Delivery Ratio, Dr: Reverse Delivery Ratio

5.3.2.4 Expected Transmission Time Metrik

Die Expected Transmission Time Metrik (ETT) ist eine Verbesserung der ETC Metrik. Bei dieser Metrik haben zusätzlich die Paketgrösse und die Bandbreite einen Einfluss.

$$ETT = ETX * \frac{S}{B}$$

Formel 3: Berechnung ETT Metrik

S: Paketgrösse, B: Bandbreite

5.3.3 Stabilität

Die Gondelbahn weist eine hohe Dynamik aus. Damit die Daten aber ohne Unterbruch übertragen werden können, muss das Netzwerk stabil bleiben. Das Routing-Protokoll sollte nicht bei jedem Topologie-Wechsel die ganzen Tabellen neu berechnen.

5.3.4 Geeignet für Gondelbahn

Anhand eines konkreten Projektes wird noch bewertet, ob das Routing-Verfahren verwendet werden kann oder diverse Probleme auftreten könnten.

5.4 Routing Protokolle

In mobilen Ad-hoc-Netzwerken gibt es im Moment über 70 verschiedene Entwürfe für das Routing durch das Netzwerk. Nachfolgend sind die wichtigsten erklärt. Sehr viele Routing-Protokolle sind aber erst theoretisch spezifiziert und wurden in Simulatoren getestet. Eine praktisch verwendbare Implementierung fehlt.

5.4.1 Optimized Link State Routing (OLSR)

Das Optimized Link State Routing (OLSR) [4] ist ein Protokoll für mobile Ad-Hoc-Netzwerke. Es ist ein proaktives Routing-Protokoll, das den Link-State-Algorithmus im mobilen Netzwerk optimiert. Der Routing-Pfad wird somit schon vor dem ersten Senden von Nutzdaten ermittelt. Im Netzwerk werden dadurch ständig Routing-Informationen ausgetauscht. Durch die Zusatzfunktion Multipoint Relay (MPR) wird der stetige Austausch von Routing-Information aber optimiert.

5.4.1.1 Multipoint Relay

Jeder Knoten im Netzwerk wählt Nachbarknoten aus, die Kontrollpakete vom Knoten weitersenden sollen. Diese Nachbarknoten werden als Multipoint Relays bezeichnet. Alle anderen Nachbarknoten bekommen die Kontrollpakete, sollen aber diese Informationen nicht weitersenden. Somit ist sichergestellt, dass ein Kontrollpaket über zwei Hops nur immer von einem Knoten weitergesendet wird. Die Anzahl an Kontrollpaketen im Netzwerk kann dadurch reduziert werden.

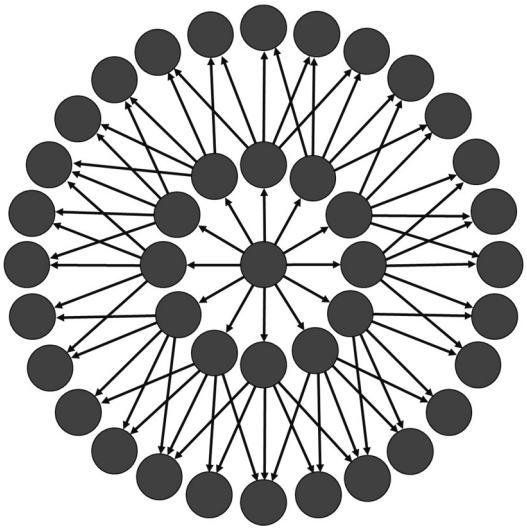


Abbildung 7: OLSR - Ohne optimiertes Flooding

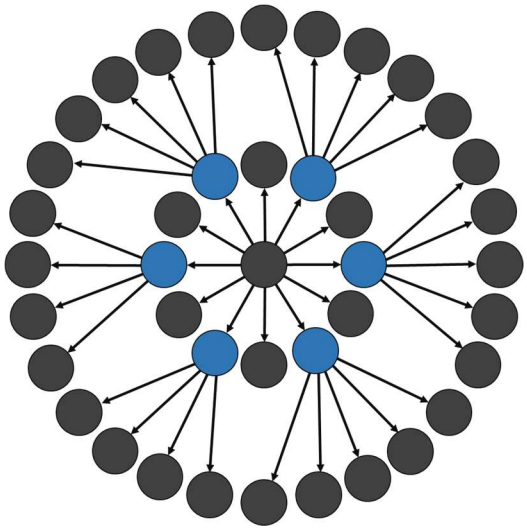


Abbildung 8: OLSR - Mit optimiertem Flooding (MPR)

5.4.1.2 Hello Message

Damit nun der Knoten erkennen kann, welche seine Multipoint Relays sind, wird an alle Nachbarknoten eine Hello-Nachricht versendet. Das Standardintervall beträgt zwei Sekunden. Die Nachbarknoten senden als Antwort eine Liste ihrer Nachbarknoten zurück. Dadurch kann der Knoten die optimalen Multipoint Relays auswählen. Danach muss der Knoten allen Nachbarknoten mitteilen, welches seine Mulitpoint Relays sind.

5.4.1.3 Topology Control Message

Informationen zur Topologie werden in Topology Control Messages versendet. Diese Nachrichten enthalten Informationen zu den Nachbarknoten und eine Sequenznummer, damit die Knoten wissen, welche Informationen im Moment aktuell sind. Die TC Nachrichten werden nur von Knoten versendet, die auch von einem anderen Knoten als Multipoint Relay verwendet werden.

5.4.1.4 OLSRv2

Seit dem April 2014 gibt es den überarbeiteten OLSRv2 Standard. Dieser ist beschrieben im RFC 7181 [5]. Er basiert auf den gleichen Mechanismen und Algorithmen, jedoch ist zur Auswahl der kürzesten Route nicht mehr der Hop Count entscheidend, sondern die ETX-Metrik.

5.4.1.5 Fazit

Bei OSLR werden die Routing-Informationen ständig ausgetauscht. Je mehr Knoten im Netzwerk vorhanden sind, desto besser funktioniert die Multipoint Relay-Optimierung. Die Skalierbarkeit ist dadurch immer noch möglich. Da der Pfad bereits zu der Zeit bekannt ist, wo Nutzdaten übertragen werden sollen, gibt es keine Verzögerung beim Senden von Informationen.

Kriterium	Bewertung
Skalierbarkeit	Je mehr Knoten, desto mehr Control-Nachrichten werden versendet. Durch die Optimierung können diese Nachrichten aber klein gehalten werden, was eine Skalierung bei hoher Knotendichte immer noch möglich macht.
Routing Metrik	OLSR: Next Hop OLSRv2: ETX
Stabilität	Bei hoher Dynamik braucht es mehr Control Messages, die dadurch entstehende Algorithmen-Berechnung macht das Netzwerk mit der Zeit instabil.
Gondelbahn geeignet	OLSR ist besser für statische Mesh-Netzwerke geeignet. Die Gondelbahn ist aber sehr dynamisch, und darum ist ein OLSR ohne spezielle Anpassungen nicht geeignet.
Referenz Implementation	olsrd (OLSR Daemon)

Tabelle 7: Bewertung OLSR

5.4.2 Ad-hoc On-Demand Distance Vector (AODV)

Das Ad-hoc On-Demand Distance Vector (AODV) - Routing-Protokoll [6] steuert ebenfalls das Routing der mobilen Geräte in einem Ad-hoc-Netzwerk. Es ist ein reaktives Routing-Protokoll, welches schnell auf Link-Änderungen reagiert und das Netzwerk wenig belastet. Der Link wird, wie der Name schon sagt, On-Demand berechnet. Inaktive Routen werden nicht gewartet und verbrauchen daher keine Systemressourcen. Das Protokoll braucht daher wenig Prozessorleistung. Damit keine Schleifen im Netzwerk entstehen, arbeitet AODV mit Ziel-Sequenz-Nummern. Die Ziel-Sequenz-Nummer wird dabei vom Zielknoten generiert und an alle angefragten Knoten zurückgesendet. Die angefragten Knoten wählen die Verbindung mit der besten Sequenznummer aus.

5.4.2.1 Message Types

Im AODV gibt es insgesamt vier verschiedene Message Formate die über UDP mit normalen IP-Headern übertragen werden.

Message Type	Funktion
Route Request (RREQ)	Die Message wird verwendet, sobald ein Knoten eine Verbindung zu einem anderen Zielknoten aufbauen will. Es beginnt der sogenannte Route Discovery-Prozess. Beim Route Discovery-Prozess wird das Netzwerk mit RREQ Messages geflutet. Eine Sequenznummer hilft, dass ein Paket von einem Knoten nicht mehrmals weitergeleitet wird.
Route Reply (RREP)	Sobald ein Knoten eine RREQ-Nachricht erhält, wird ein Route Reply zum Sender zurückgesendet. In diesem Paket befindet sich zudem ein Lifetime-Feld, um festzustellen, ob die Route noch Gültigkeit besitzt. Die Antwort wird als Unicast versendet.
Route Error (RERR)	Fällt ein Link zwischen zwei Knoten aus, melden die Knoten mit der Route Error-Nachricht den Nachbarn, dass der Link nicht mehr zur Verfügung steht. Der Nachbar sendet die Information an alle anderen betroffenen Zielknoten weiter, damit diese die Routing-Informationen löschen können.
Route Reply Acknowledgment (RREP-ACK)	Diese Nachricht ist optional, wird aber verwendet, um den Erhalt einer Route Reply Message zu bestätigen.

Tabelle 8: AODV Message Types

Um feststellen zu können, ob ein Link zwischen zwei Knoten noch aktiv ist, wird die Hello Message verwendet. Die Hello Message ist somit ein Indikator für einen Link Failure und ein Auslöser für die Route Error Message.

5.4.2.2 Fazit

AODV ist für mobile Ad-Hoc-Netzwerke gestaltet und kann bis zu 1000 Knoten bedienen. Die Datenraten können sehr unterschiedlich sein. Es wird versucht, den Kontrollverkehr möglichst klein zu halten. Erst bei einer Kommunikation mit Nutzdaten wird die Route zwischen den Knoten ermittelt. Dies wirkt sich positiv auf die Leistung und die Skalierbarkeit aus. Jedoch hat dies auch einen Einfluss auf die Antwortzeit. Bevor Nutzdaten überhaupt übertragen werden können, muss der Pfad ermittelt werden, was Zeit benötigt. Für sehr hohen Datendurchsatz ist AODV nur bedingt geeignet.

Kriterium	Bewertung
Skalierbarkeit	Sehr gut, da keine Statusinformationen ausgetauscht werden. Bei kontinuierlichem Datenstrom könnte es aber mit der Zeit Verzögerungen geben.
Routing Metrik	Next Hop.
Stabilität	Sofern die Datenübertragung nicht länger anhält, ist die Stabilität im Netzwerk gewährleistet.
Gondelbahn geeignet	Bei kurzer Datenkommunikation zu einzelnen Gondeln wäre das Protokoll geeignet, jedoch für längere Datenübertragungen mit vielen Topologie-Wechseln nicht.
Referenz Implementation	Keine eigene Implementation. Kommt jedoch in einer abgeänderten Variante in HWMP vor.

Tabelle 9: Bewertung AODV

5.4.3 B.A.T.M.A.N.

B.A.T.M.A.N. «Better Approach to Mobile Ad-Hoc Networking» [7] ist wie OLSR ein proaktives Protokoll bzw. ein Algorithmus für mobile Ad-Hoc-Netzwerke. Es wurde entwickelt, um die Schwachstellen von OLSR zu verbessern.

Die Grundidee von B.A.T.M.A.N. ist es, das Wissen über die besten End-zu-End-Pfade im Mesh-Netzwerk auf alle teilnehmenden Knoten zu verteilen. Jeder Knoten verwaltet nur die Informationen über den nächstbesten Hop zu allen anderen Knoten. Dadurch wird ein globales Wissen über lokale Netzwerkänderungen überflüssig. Ausserdem sorgt ein ereignisbasierter Flooding-Mechanismus dafür, dass keine widersprüchlichen Topologie-Informationen entstehen (Routing-Loops).

5.4.3.1 Algorithmus

1. Jeder Knoten versendet Broadcast-Nachrichten (OGM – Originator Messages) um die Nachbarknoten über seine Existenz zu informieren.
2. Diese Nachbarknoten broadcasten diese Nachricht nach speziellen Regeln weiter, um wiederum die Nachbarknoten über die Existenz des Originators zu informieren.
3. Somit wird das ganze Netzwerk mit diesen OGMs geflutet. Diese OGMS sind relativ klein. Ein einzelnes Paket besitzt inkl. IP und UDP Overhead 52 Byte. Dabei enthalten sie folgende Informationen:
    a. Adresse des Originators
    b. Adresse des übermittelnden Knoten
    c. TTL (Time to Life)
    d. Sequenznummer.
4. OGMs, die einem Pfad folgen, der über schlechte Qualität verfügt, sind meistens länger unterwegs bzw. haben Paketverluste. Dadurch kann erkannt werden, welcher Weg durchs Netzwerk besser geeignet ist.
5. Für das Erkennen, ob ein OGM bereits empfangen wurde, dient die Sequenznummer, welche vom Originalknoten vergeben wurde.
6. Jeder Knoten broadcastet jede empfangene OGM höchstens einmal und dies auch nur, wenn die OGM von jenem Nachbarknoten empfangen wurde, der als derzeit bester Next Hop zum Originator des OGM identifiziert ist.

Durch diesen Algorithmus erfahren alle Knoten die Existenz der anderen Knoten. Ein Knoten X erfährt die Existenz von einem entfernten Knoten Y über das Empfangen von dessen OGM. Falls Knoten X mehr als einen Nachbarknoten hat, kann dieser aufgrund mehrerer empfangener Nachrichten entscheiden, welches der nächste Hop zu Knoten Y sein soll. Meistens ist es natürlich jener, wo das OGM schneller bzw. zuverlässiger empfangen wurde. Anschliessend wird die Routing-Tabelle mit dem Next Hop zum Originalknoten konfiguriert.

5.4.3.2 B.A.T.M.A.N. Advanced

B.A.T.M.A.N. Advanced [8] ist eine Implementierung des B.A.T.M.A.N. Algorithmus auf Layer 2. Das Routing geschieht dabei anhand von MAC-Adressen und normalen Ethernet Frames. Durch die Implementierung auf Layer 2 entstehen folgende Vorteile:

- Unabhängig vom Netzwerk-Layer – es kann IPv4, IPv6, DHCP, IPX, etc. eingesetzt werden.
- Knoten können im Mesh ohne IP Adresse teilnehmen.
- Einfache Integration von Nicht-Mesh-Clients.
- Roaming von Nicht-Mesh-Clients.
- Optimierung des Datenflusses durch das Mesh (Bsp. Multicast, Forward Error Correction, Interface-Alternating, Interface-Bonding, etc.).
- Protokolle, welche auf Broadcast oder Multicast basieren, können eingesetzt werden (Streaming-Dienste, mDNS, Windows Neighborhood, etc.).

5.4.3.3 Performance

Aus einer bestehenden [9] Evaluation zweier Layer-2 Mesh Protokolle geht hervor, dass B.A.T.M.A.N Advanced enorme Probleme beim Neugenerieren der Routing-Tabelle hat, sobald ein Knoten ausfällt. Dies liegt daran, dass ein Knoten erst dann als unerreichbar gilt, wenn ein sogenannter PURGE_TIMEOUT Intervall abgelaufen ist. Dies ist ein vordefiniertes Zeitfenster, in dem von einem Originator keine neuen OGMs mehr empfangen werden. Der Default Wert des PURGE_TIMEOUT liegt bei 200 Sekunden.

Ausserdem besitzt B.A.T.M.A.N Advanced trotz der Implementierung auf Layer 2 einen grossen Overhead, da er Ethernet Frames mit einem eigenen Header kapselt.

5.4.3.4 Bewertung

Kriterium	Bewertung
Skalierbarkeit	Skalierbarkeit gut, da das Wissen über die besten End-zu-End-Pfade über mehrere Knoten verteilt ist.
Routing Metrik	Transmission Quality (TQ) - Anzahl empfangener OGMs auf Link.
Stabilität	Bei einem Ausfall des Knotens bricht die Verbindung über diesen Knoten zusammen und es dauert einige Zeit (PURGE_TIMEOUT), bis eine neue Verbindung gefunden wird.
Für Gondelbahn geeignet	Gerade das Beispiel mit der hohen Konvergenzzeit zeigt auf, dass dieses Protokoll für ein dynamisches Netzwerk nicht geeignet ist.
Referenz Implementation	batman-adv, batmand

Tabelle 10: Bewertung B.A.T.M.A.N.

5.4.4 802.11s

IEEE 802.11s definiert einen Standard, der die Funktionsweise von drahtlosen Mesh-Netzwerken auf dem Data-Link-Layer regelt. Anders als die übrigen Mesh-Technologien führt 802.11s das Routing dabei auf der MAC-Ebene durch und nicht auf dem klassischen Network Layer. Dadurch kann die Bearbeitungszeit einzelner Datenpakete optimiert werden. Der Overhead, der ein Layer-3 Mesh-Protokoll mit sich bringt, entfällt.

Die mögliche Grösse eines 802.11s Mesh-Netzwerkes ist mit 32 Knoten angegeben. Dieses Limit sollte aber nicht als zu strikt angesehen werden. In Wirklichkeit heisst es nur, die Technologie sei nicht für grosse Netzwerke mit mehreren hunderten Knoten ausgelegt. [10]

5.4.4.1 Grundlagen

Folgendes Bild zeigt die einfache Struktur eines Mesh-Netzwerkes auf Basis von 802.11s.

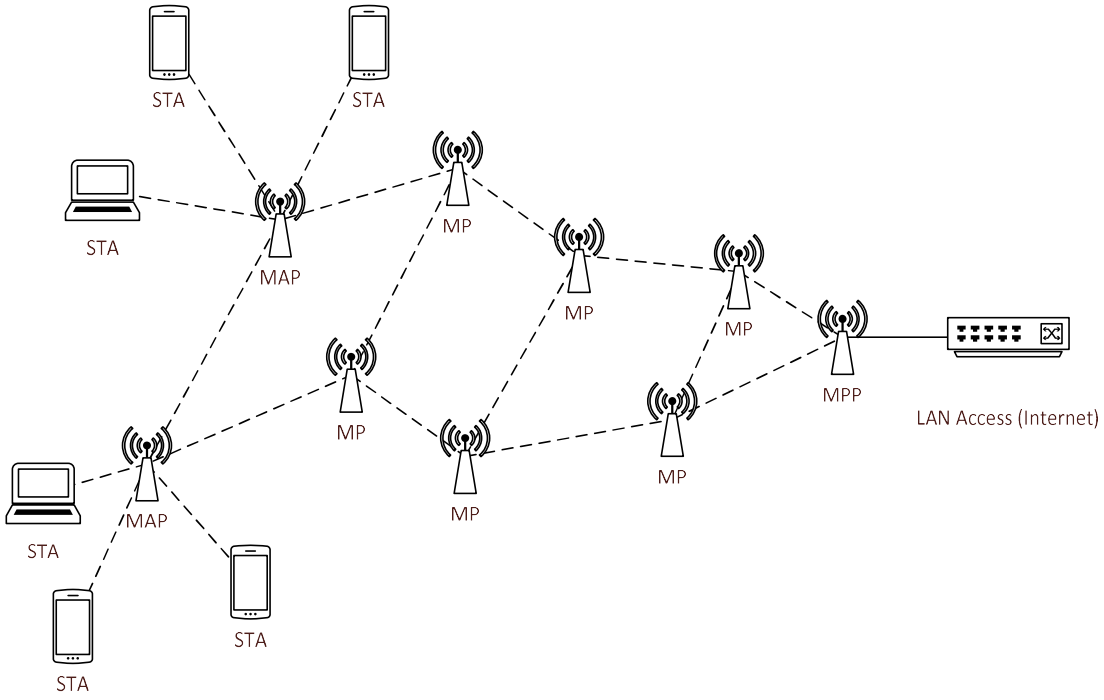


Abbildung 9: Mesh-Netzwerk auf Basis von 802.11s

Dabei werden den Netzwerkgeräten, die das Mesh aufbauen, verschiedene Rollen zugewiesen.

Rolle	Beschreibung
MP: Mesh Point	Baut eine Verbindung zu anderen Mesh Points auf.
MAP: Mesh Access Point	Gleiche Funktion wie ein MP. Zusätzlich bietet der MAP eine Zugriffsmöglichkeit für Stationen mit dem Mesh zu kommunizieren (Access-Point).
MPP: Mesh Point Portal	Durch ein MPP kann das Mesh-Netzwerk an ein kabelgebundenes Netzwerk gekoppelt werden. Beispielsweise um Internetzugriff zu ermöglichen.

STA: Station	Die Station ist ein Synonym für Client. Stationen verbinden sich mit dem MAP, um Zugriff auf das Mesh-Netzwerk zu erhalten. STA selber haben keine Mesh-Funktion und müssen 802.11s dadurch auch nicht unterstützen.

Tabelle 11: Terminologie 802.11s Komponenten

5.4.4.2 Routing

Für das Routing innerhalb von 802.11s-Mesh-Netzwerken kommt standardmässig das Hybrid Wireless Mesh-Protokoll (HWMP) [3, pp. 1382-1413] zum Einsatz. Hybrid bedeutet hierbei, dass sowohl ein reaktiver, als auch proaktiver Modus verwendet wird. Im proaktiven Modus hat ein Mesh Point die Rolle eines Root-Knotens und stellt anhand einer Baumtopologie Pfade zu allen anderen Mesh-Teilnehmern her. Das Verwenden des proaktiven Modus kann konfiguriert werden.

HWMP basiert auf dem Radio-Metric Ad-hoc On-Demand Distance Vector Routing-Protokoll (RM-AODV), einer Adaption von AODV. Während AODV auf Layer-3 mit IP-Adressen routet und Hop-Counts als Metrik benützt, arbeitet RM-AODV auf Layer-2 und benützt die Airtime Link Metrik für die Pfadbestimmung.

HWMP benutzt Ziel-Sequenznummern, um abgelaufene Routing-Informationen zu erkennen. Neu erhaltene Routing-Informationen mit kleineren Sequenznummern als die bereits vorhandenen werden ignoriert und verworfen.

Einträge in der Routing-Tabelle sind mit einer Lifetime verknüpft. Dies ermöglicht die Löschung ungenützter Pfade. Die Lifetime wird bei jedem Übermitteln eines neuen Datenframes bzw. beim Empfangen von neuen Routing-Protokoll-Nachrichten neu gesetzt.

5.4.4.3 Algorithmus für Pfadselektion

On-Demand:

1. Source broadcastet Path Request (PREQ) mit MAC-Adresse der Destination.
2. Alle Mesh-Stationen, die den Request erhalten, erstellen oder updaten ihren Pfad zur Source (aber nur, wenn der PREQ eine Sequenznummer enthält, welche grösser als der bisherige Pfad ist, oder falls die Sequenznummer gleich ist, eine bessere Metrik aufweist).
3. Jede Mesh-Station muss, bevor es den PREQ weiter verteilt (Broadcast), das Metrik-Feld aktulisieren. In diesem Feld befindet sich somit immer der kumulierte Metrik-Wert zur Source.
4. Sobald die Destination den PREQ erhält, sendet diese ein Unicast Path Reply (PREP). Falls die Destination mehrere PREQ erhält mit besseren Metrik-Werten (bei gleicher oder höherer Sequenznummer), so sendet sie erneut einen PREP über den aktualisierten Pfad.
5. Die Mesh-Stationen zwischen Source und Destination leiten das PREP über den besten Pfad (welcher über das PREQ Flooding ermittelt wurde) zur Source weiter. Sobald die Source den PREP empfangen hat, kann ein bidirektionaler Pfad aufgebaut werden, um Daten zu übermitteln.
6. Falls mehr als ein PREP von der Source empfangen wurde, werden diese nur beachtet, falls sie noch aktuell sind und einen besseren Metrik-Wert aufweisen.

Proaktiv:

Mit dem proaktiven Modus wird eine Baumartige Struktur im Netzwerk aufgebaut. Dafür wird ein Mesh Point als Root konfiguriert. Idealerweise ist dies ein Mesh Point Portal. Es wird zwischen drei verschiedenen Mechanismen unterschieden um diesen Baum aufzubauen. [11]

Mechanismus	Beschreibung
Proaktive PREQ (registration mode)	Root Knoten sendet periodisch PREQs aus. Diese werden wie im On-Demand Modus an die anderen Mesh-Teilnehmer versendet, welche mit einem PREP antworten. Somit entsteht zwischen der Root und den einzelnen Mesh Points jeweils ein bidirektionaler Pfad.
Proaktive PREQ (non-registration mode)	Wie im Registration Mode sendet der Root Knoten periodisch PREQs aus. Die Mesh-Teilnehmer antworten jedoch nicht mit einem PREP. Es entsteht ein baumartiger Pfad von den Mesh Points zur Root hin, ohne das die Root weiss, wie die einzelnen Mesh Points erreichbar sind.
Proaktive RANN	Root Knoten sendet periodisch Root Announcements aus. Aufgrund dieser Root Announcements suchen die anderen Mesh Points im Netzwerk mittels normalem On-Demand Modus den besten Pfad zur Root.

Tabelle 12: Proaktive Mechanismen in 802.11s

5.4.4.4 open80211s

Open80211s ist eine Referenz-Implementierung des 802.11s Standards. Diese wurde in der Zeit entwickelt, als sich der Standard noch im Draft befand. Die Idee hinter der Implementierung war es, die vielen verfügbaren Mesh-Protokolle zu einem einzigen Mesh-Protokoll zu vereinen, das auf einem Standard basiert. Ausserdem sollte dieses Protokoll dazu dienen, Mesh-Netzwerke an sich besser zu verstehen. Die Implementierung von 802.11s findet sich seit dem Jahre 2012 im Linux Kernel.

5.4.4.5 Fazit

Der 802.11s Standard mit dem HWMP-Protokoll besitzt dadurch, dass er auf Layer-2 Ebene agiert, einige Vorteile gegenüber anderen Mesh-Routing-Protokollen. Der Wegfall des Overheads macht das Routing besonders effizient hinsichtlich Verarbeitungszeit, und dadurch indirekt auch für die Performance bzw. den Stromverbrauch.

Da es sich um einen Standard handelt, muss 802.11s auch herstellerübergreifend einsetzbar sein. Dies wird durch das standardmässig integrierte HWMP-Protokoll ermöglicht.

5.4.4.6 Bewertung

Kriterium	Bewertung
Skalierbarkeit	Ausgelegt für ca. 32 Knoten, es sind aber auch mehr möglich
Routing Metrik	Airtime Link Metrik.
Stabilität	Durch das eingesetzte Protokoll HWMP, welches ein proaktives wie reaktives Routing unterstützt, kann eine bessere Stabilität erreicht werden.
Für Gondelbahn geeignet	Da es ein Hybrides-Protokoll ist, hat es sowohl seine Vorteile, wenn die Gondelbahn fährt wie auch wenn sie steht. Eigene Anpassungen brauchen tiefe Protokollkenntnisse und Kernelwissen.
Referenz Implementation	open20811s, in Linux Kernel integriert.

Tabelle 13: Bewertung 802.11s

5.4.5 Weitere Routingprotokolle

Wie schon beschrieben wurde, gibt es sehr viele Routing Protokolle im Mesh Bereich. Die folgenden wurden kurz angeschaut aber auf Grund der Anforderungen relativ schnell wieder verworfen. Der Vollständigkeit halber werden sie hier noch kurz erwähnt und beschrieben.

5.4.5.1 Destination Sequenced Distance Vector (DSDV)

Destination-Sequenced Distance-Vector Routing (DSDV) [12] basiert auf dem Bellman-Ford-Algorithmus und ist ein proaktives Routing-Verfahren. Jeder Knoten wartet seine eigene Routing-Tabelle. Ändert die Topologie im Netz, werden entweder «full-dumps» oder nur inkrementelle Updates durchs Netzwerk gesendet. Bei jedem Topologie-Wechsel wird zudem eine neue Sequenznummer generiert und das Netzwerk neukonvergiert. Die Sequenznummer ist zudem wichtig, damit keine Loops im Netzwerk entstehen. Es ergibt sich sonst eine schlechte Skalierung bei vielen Änderungen, da die Routing-Tabellen immer neu berechnet werden müssen.

5.4.5.2 Dynamic Source Routing (DSR)

Dynamic Source Routing (DSR) [13] ist ein reaktives Routing-Verfahren und hat gewisse Ähnlichkeiten zu AODV. Erst wenn Daten gesendet werden wollen, sucht dieses Verfahren den Weg zum Ziel. Das Netzwerk wird nicht mit unnötigen Kontrollpaketen belastet. Interessanterweise werden bei diesem Protokoll die Zieladressen der Knoten gerade mit ins Paket gepackt. Netzwerkgeräte, die Daten nur weiterleiten, brauchen keine gültige Routing-Tabelle. Dadurch kann aber der Header des Paketes sehr gross werden.

5.5 Beurteilung anhand von Technologien

Im folgenden Abschnitt wird beschrieben, welche Technologie am besten die definierten Problemzonen aus Kapitel 2.4 abdeckt.

5.5.1 Reaktiv

Anhand der Erfahrungen der theoretischen Lösungsskizze eignet sich ein reaktives Routing-Protokoll am besten. Der Kontrollfluss bei proaktiven Protokollen ist zu hoch. Auch ein individueller Algorithmus könnte diesen Kontrollfluss nicht wirklich verkleinern, da die definierten Problemzonen zu unterschiedlich sind.

Ein weiterer Vorteil des reaktiven Algorithmus ist zudem, dass die Rechenleistung nur gebraucht wird, wenn auch Daten gesendet werden sollen. Somit sinkt der Stromverbrauch, da nicht immer gerechnet werden muss.

5.5.2 Layer 2

Das Routing auf Layer 2 bietet viele Vorteile. Einerseits wird die Komplexität reduziert und andererseits sinkt der Stromverbrauch. Ein weiterer Vorteil für die Zukunft von Layer 2 Routing ist der flexible Protokoll-Einsatz auf Layer 3. Ein Layer 3 Routing-Verfahren bindet sich automatisch an ein Layer 3-Protokoll. Beim Layer 2 Routing können unabhängige Layer 3-Protokolle verwendet werden. Im Moment ist das am meisten verwendete Protokoll noch IPv4, aber auch IPv6 wäre möglich.

5.5.3 Metrik

In der Auswahl des besten Pfades spielt die Metrik eine entscheidende Rolle. Die Minimal Hop Count-Metrik ist für ein Wireless-Mesh-Netzwerk untauglich. Bei der drahtlosen Kommunikation kommt es viel häufiger zu Paketverlusten. Die Metrik sollte darum auch die Anzahl an Paketverlusten beachten. Zudem ist die verfügbare Bandbreite ein weiterer wichtiger Indikator. Eine stabile aber langsame Verbindung könnte den Netzwerkfluss nur unnötig behindern. Eine Mischung aus den Faktoren Paketverlust und verfügbarer Bandbreite wäre die ideale Metrik.

5.5.4 GPS Problematik

Das Routing mit GPS zu lösen wäre durchaus eine interessante Variante. Bei der Implementierung von GPS-Informationen ins Routing gibt es aber ein entscheidendes Problem. Die Knoten müssen die GPS-Informationen untereinander austauschen, um berechnen zu können, welches der beste Pfad ist. Der Austausch dieser Informationen funktioniert aber nicht, da kein funktionierendes Netzwerk existiert. Es braucht daher andere Übertragungsmöglichkeiten um diese GPS Informationen untereinander auszutauschen. Dies GPS Variante ist in diesem Fall wenig praktikabel und daher nicht geeignet für die Vernetzung einer Gondelbahn.

5.6 Überblick Protokolle

In der untenstehenden Tabelle sind die verschiedenen Routing-Verfahren mit ihren Eigenschaften aufgelistet.

Routing-Protokoll	Layer 2 / Layer 3	Proaktive / Reaktiv	Metrik	LS / DV	Besondere Eigenschaften
OLSR	Layer 3	Proaktiv	Minimal Hop Count	Link State	
OLSRv2	Layer 3	Proaktiv	ETX	Link State	
AODV	Layer 3	Reaktiv	Minimal Hop Count	Distance Vector	Führt eine Tabelle pro Knoten
B.A.T.M.A.N.	Layer 3	Proaktiv	TQ (eigene Art von ETX)	Distance Vector	
B.A.T.M.A.N. Adv	Layer 2	Proaktiv	Throughput (eigene Art von ETT)	Distance Vector	
802.11s (HWMP)	Layer 2	Hybrid	Airtime Link Metric	Distance Vector	Als Routing-Protokoll wird RM-AODV verwendet
DSDV	Layer 3	Proaktiv	Minimal Hop Count	Distance Vector	
DSR	Layer 3	Reaktiv	Minimal Hop Count	Distance Vector	Nur Sender getrieben, führt keine eigene Tabelle pro Knoten.

Tabelle 14: Zusammenfassung Mesh-Protokolle

5.7 Entscheidung

Die Beurteilung der Lösungsskizze lässt darauf schliessen, dass ein reaktives Protokoll für das Routing innerhalb des Mesh-Netzwerks am besten geeignet ist. Ein proaktives Protokoll sendet bei einem Topologie-Wechsel eine erhebliche Anzahl Kontrolldaten durch das Netzwerk, was zu Instabilität und Trägheit führt. Im Falle der Gondelbahn würde dies zu Performanceeinbussen führen, was sich z.B. negativ auf eine mögliche Internetanbindung der Gondel auswirken könnte. Daher kommen lediglich AODV, 802.11s (HWMP) sowie DSR in Frage, wobei bei DSR und AODV die Metrik ein Problem darstellt. Diese Protokolle berechnen den besten Pfad auf Grund des Minimal-Hop-Counts. Bei der Gondelbahn könnte die Situation entstehen, dass im fahrenden Betrieb eine Gondel auf der Strecke «übersprungen» wird. Dadurch entsteht womöglich ein Pfad, der auf Grund der niedrigen Übertragungsqualität sehr unzuverlässig ist.

802.11s beziehungsweise HWMP bietet mit der Airtime Link Metrik die Möglichkeit, den besten Pfad aufgrund der Übertragungsqualität des WLAN zu definieren, was zwangsläufig die stabilste Verbindung ist. Ausserdem bietet es durch das hybride Verfahren die optionale Möglichkeit, den reaktiven Modus um einen proaktiven zu ergänzen. Ein weiterer Vorteil von HWMP ist die Layer-2-Implementierung. Das gesamte Routing geschieht anhand der MAC-Adressen. Somit spielt es keine Rolle, welche Technologien im oberen Netzwerk-Layer verwendet werden.

Durch die genannten Spezifikationen ist 802.11s somit die bestmöglichste Variante. Die Entscheidung 802.11s zu verwenden wird insofern noch dadurch bestärkt, dass diese Technologie unterdessen auch in den 802.11 Standard integriert worden ist. Im Hinblick auf eine zukunftsträchtige Lösung ist dies sicher nicht zu ignorieren.

6 802.11s in der Praxis

Im Kapitel 802.11s wurde das Routing-Protokoll HWMP theoretisch beschrieben. In diesem Kapitel wird nun analysiert, wie die verschiedenen Pakete in der Praxis aussehen und wie effektiv Daten über dieses Protokoll übertragen werden können.

6.1 Peering

Damit die verschiedenen Mesh Points überhaupt miteinander kommunizieren können, muss zuerst ein Peering durchgeführt werden. Dazu senden diese kontinuierlich Beacons aus. Diese Beacons sind eine Art Lebenszeichen der Mesh Points und enthalten das Mesh-Profil. Dieses ist dazu da, im Mesh-Netzwerk teilnehmende Mesh Points zu identifizieren. Jedes Beacon mit demselben Mesh-Profil gehört zum selben Netzwerk. Das Profil enthält unter anderem die ID des Netzwerkes, sowie Informationen über verwendetes Protokoll und Metrik.

Um ein Peering durchführen zu können, existieren drei Management Frames.

Management Frame	Beschreibung
Mesh Peering Open	Stellt Antrag an Peering-Aufbau und beinhaltet Informationen zu Fähigkeiten der sendenden Station.
Mesh Peering Confirm	Bestätigt den Aufbau des Peering.
Mesh Peering Close	Beendet ein Peering.

Tabelle 15: Mesh Peering Frames

Nachdem ein Mesh Point ein Beacon eines anderen empfangen hat, der dasselbe Mesh-Profil besitzt, sendet dieser ein «Mesh Peering Open». Dieses Frame enthält Informationen über die sendende Station, wie zum Beispiel die akzeptierten Bandbreiten. Ebenfalls Bestandteil dieses Frames ist das Mesh-Profil. Stimmt das Mesh-Profil mit dem der empfangenden Station überein, so kann diese nun mit einem «Mesh Peering Confirm» antworten, um die Verbindung zu bestätigen.

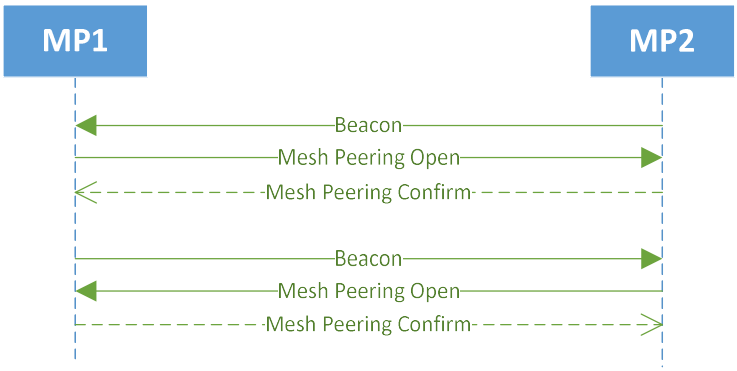


Abbildung 10: Mesh Peering

Diese Kommunikation erfolgt jeweils beidseitig. Erst wenn beide Stationen mit dem jeweiligen Gegenüber ein Mesh Peering Open bzw. Mesh Peering Confirm durchgeführt haben, besteht eine Verbindung. Die Reihenfolge, wer zuerst ein Mesh Peering Open versendet, spielt dabei keine Rolle. Diese

könnten auch verzahnt ineinander ablaufen. Nach einem erfolgreichen Peering befinden sich die Mesh Points in einer Layer 2 Broadcast-Domäne. Möchte man ein Peering wieder auflösen, oder stimmt das Mesh-Profil nicht überein, so kommt das Mesh Peering Close Frame zum Einsatz.

Sind zwei Mesh Points miteinander ein Peering eingegangen, so kann dies unter Linux mit dem Kommandozeilentool iw[2] nachvollzogen werden.

```
$: iw dev $MESH_IFACE station dump
```

```
Station c0:56:27:c8:0a:8b (on mp11)
        inactive time:  30 ms
        rx bytes:       23553372
        rx packets:     73156
        tx bytes:       602336
        tx packets:     6015
        tx retries:     777
        tx failed:      94
        rx drop misc:   25310
        signal:         -39 dBm
        signal avg:     -39 dBm
        Toffset:        10049371 us
        tx bitrate:     54.0 MBit/s
        rx bitrate:     54.0 MBit/s
        expected throughput:    38.85Mbps
        mesh llid:      49387
        mesh plid:      63660
        mesh plink:     ESTAB
        mesh local PS mode:     ACTIVE
        mesh peer PS mode:      ACTIVE
        mesh non-peer PS mode:  ACTIVE
        connected time: 9032 seconds
```

Abbildung 11: Auszug mit relevanten Informationen aus Station Dump

Als Ausgabe erfolgt eine Liste aller verbundenen Mesh Points und deren Informationen. Einige Werte sind dabei besonders interessant hinsichtlich der Lösung des Gondelbahnproblems; sie sind deshalb unten in der Tabelle genauer erläutert.

Parameter	Beschreibung
inactive time	Gibt die Zeit an, wann das letzte Beacon des verbundenen Mesh Points empfangen wurde. Nach Erreichen einer vordefinierten Zeit wird der gesamte Eintrag aus der Liste gelöscht.
signal	Signalstärke.
mesh plink	Gibt an, ob über den verbundenen Mesh Point Daten versendet werden können oder nicht. Der Status ESTAB steht hierbei für «established» und bedeutet, dass ein Peering zwischen den Mesh Points stattgefunden hat und sie miteinander Daten austauschen können.
connected time	Gibt an, wie lange das Peering schon besteht.

Tabelle 16: Station Dump Inhalte

[2] Dokumentation zum Linux Kommandozeilen Werkzeug iw unter folgender Adresse verfügbar: https://wireless.wiki.kernel.org/en/users/documentation/iw

Standardmässig wird nach einem erfolgreichen Peering der Status von «mesh plink» auf «ESTAB» gesetzt, sodass über diese Verbindung kommuniziert werden kann. Möchte man dies unterbinden, so kann der Status manuell auf «BLOCKED» gesetzt werden. Dies geschieht mit folgendem Befehl:

```
$: iw dev $MESH_IFACE station set «MAC ADRESSE» plink_action block
```

Die Mesh Points können sich über die Beacon Frames noch gegenseitig sehen, jedoch wird die Verbindung darüber blockiert. Dies eignet sich hervorragend, um Tests durchführen zu können.

Um die Liste der verbundenen Mesh Points möglichst klein zu halten, kann über den IW Mesh-Parameterwert «mesh_plink_timeout» ausserdem definiert werden, nach welcher Dauer der Inaktivität der Eintrag gelöscht werden soll. Standardmässig liegt der Wert bei 1‘800 Sekunden (30 Minuten). Mit folgendem Befehl kann der Timeout-Wert (in Sekunden) selbst definiert werden:

```
$: iw dev $MESH_IFACE set mesh_param mesh_plink_timeout 30
```

6.2 Daten senden (Ping)

Sobald die Mesh Points durch ein Peering miteinander verbunden sind, können Daten ausgetauscht werden. Zur Regelung des Datenverkehrs kommt hierbei das Routing-Protokoll HWMP zum Einsatz.

Um nun den Aufbau eines Routing-Pfades von HWMP nachvollziehen zu können, wird von einem Mesh Point zu einem anderen ein Ping gesendet. Zwischen diesen zwei Mesh Points befindet sich ausserdem ein dritter, welcher als Zwischen-Hop fungiert.

6.2.1 Ablauf

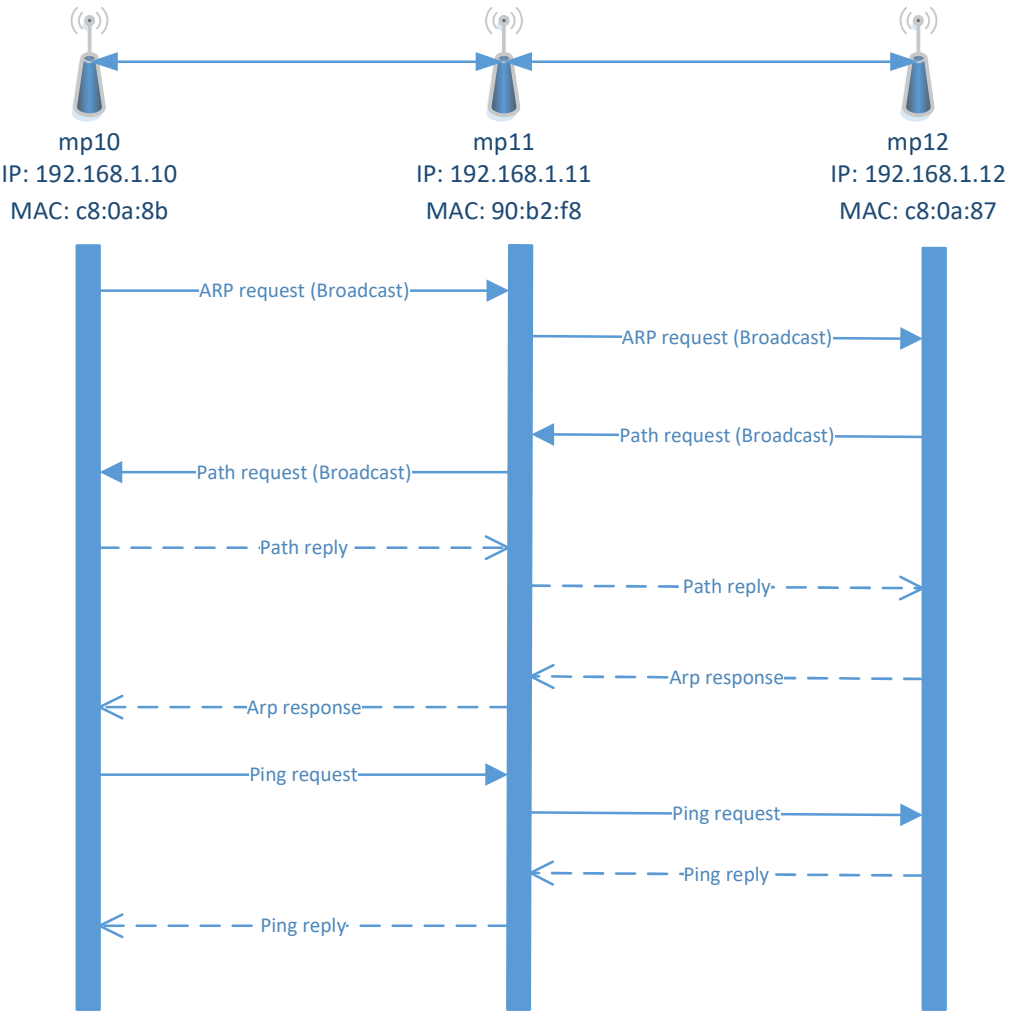


Abbildung 12: Ablauf Ping im Mesh-Netzwerk

Der Mesh Point mp10 sendet einen Ping zum Mesh Point mp12. Da die beiden Mesh Points nicht direkt miteinander verbunden sind, muss der Mesh Point mp11 die Daten weiterleiten. Da der Mesh Point mp10 nicht weiss, welches Gerät die IP-Adresse 192.168.1.12 hat, wird ein ARP Request an alle verbundenen Mesh Points versendet. Der mp11 erhält diese ARP-Nachricht und sendet sie ebenfalls an alle anderen Mesh Points weiter. Der mp12 erhält diesen ARP Request und versucht eine ARP Reply zu senden. Da dies aber eine Unicast-Nachricht ist, muss er zuerst herausfinden, über welchen Pfad er mp10 erreichen kann. Um diesen Pfad nun zu finden, sendet er an alle seine verbundenen Mesh Points einen Broadcast mit einem Path Request aus. Der Path Request wird weitergeleitet bis zum mp10 und dieser antwortet dem mp12 mit einem Path Reply. Nun kann mp12 den ARP Response senden und der Mesh Point mp10 weiss nun die MAC-Adresse vom mp12. Durch den Path Request vom mp12 weiss der mp10 auch gerade den Pfad zum mp12 und kann den Ping erfolgreich übermitteln.

6.2.2 Forwarding Table

Durch das Übermitteln der Path Requests und der Path Replys wird eine Forwarding Table auf den Mesh Points aufgebaut. Betrachtet man dies auf eine logisch abstrakte Weise, kann man sagen, dass vor jedem Mesh Point ein Switch steht. Anhand dieser Forwarding Table wird entschieden, wohin das Paket als nächstes gesendet wird.

Mit folgendem Befehl kann diese Tabelle auf Linux ausgegeben werden:

```
$: iw dev $MESH_IFACE mpath dump
```

Die Ausgabe sieht folgendermassen aus:

```
DEST ADDR          NEXT HOP           IFACE   SN    METRIC   QLEN    EXPTIME    DTIM    DRET    FLAGS
c0:56:27:c8:0a:8b  c0:56:27:c8:0a:8b  mp11    2     1366     0       0          0       0       0x14
```

Was die einzelnen Einträge genau bedeuten, wird in Tabelle 17 erläutert.

Eintrag	Abkürzung	Beschreibung
DEST ADDR	Destination Address	Zieladresse.
NEXT HOP	Next Hop	Nächster Hop auf dem Weg zur Zieladresse.
IFACE	Interface	Gibt das Interface an, über das Pakete weitergeleitet werden.
SN	Sequence Number	Diese wird benötigt, um die Aktualität eines Pfades festzuhalten. Kommt beispielsweise ein PREQ mit einer höheren Sequenznummer beim Knoten an, so wird der Eintrag in der Tabelle aktualisiert.
METRIC	Metric	Die Metrik gibt Auskunft darüber, wie gut bzw. wie schlecht ein Pfad ist. Je niedriger der Wert, desto besser der Pfad.
QLEN	Frame Queue Length	Anzahl zwischengespeicherter Frames für diesen Mesh-Pfad.
EXPTIME	Expiration Time	Definiert die Zeit, nachdem ein Mesh-Pfad seine Gültigkeit verliert. Die Ausgabe erfolgt in Jiffies.
DTIM	Discovery Timeout	Verbleibende Zeit für das Finden eines Pfades (solange dieser Pfad aktiv gesucht wird).

DRET	Discovery Retries	Anzahl der Versuche, um einen Pfad zu finden (solange dieser Pfad aktiv gesucht wird).
FLAGS	Flags	Flags für verschiedene Status: MESH_PATH_ACTIVE = BIT(0) MESH_PATH_RESOLVING = BIT(1) MESH_PATH_DSN_VALID = BIT(2) MESH_PATH_FIXED = BIT(3) MESH_PATH_RESOLVED = BIT(4)

Tabelle 17: Forwarding Table Einträge

6.3 Pfad aktualisieren

Während der Übertragung der Daten wird der Pfad ständig aktuell gehalten. Über den Path Request generiert der Sender ständig Abfragen, über welchen Pfad das Ziel noch erreichbar ist. Im Gegensatz zur initialen Pfadsuche sieht der Path Request beim Aktualisieren leicht anders aus.

6.3.1 Target Only Flag

Kennt der Mesh Point den Pfad zum Ziel noch nicht, wird ein Path Request ausgesendet, bei dem das Target Only Flag nicht gesetzt ist. Dies hat zur Folge, dass ein beliebiger Mesh Point im Netzwerk ein Path Reply senden darf. Vorausgesetzt ist, dass der Mesh Point noch einen gültigen Pfad zum Zielknoten besitzt.

```
▾ HWMP Per-Target Flags: 0x00
      .... ...0 = TO Flag: [TO = 0] Intermediate Nodes May Respond
      .... .0.. = USN Flag: [USN = 0] Target Sequence Number Known at Originator
```

Abbildung 13: Target Only Flag nicht gesetzt

Bei der Pfad Aktualisierung muss der sendende Mesh Point sicher sein, dass der Pfad zum Zielknoten noch Gültigkeit besitzt. Dies wird mit dem Target Only Flag erreicht. Ist dieses Flag gesetzt, darf nur der Zielknoten den Path Reply aussenden. Wäre dieses Flag nicht gesetzt, würde schon der Next Hop vom sendenden Mesh Point den Path Reply zurücksenden.

```
▾ HWMP Per-Target Flags: 0x01
      .... ...1 = TO Flag: [TO = 1] Only Target Will Respond
      .... .0.. = USN Flag: [USN = 0] Target Sequence Number Known at Originator
```

Abbildung 14: Target Only Flag gesetzt.

6.3.2 Aktualisierungsintervall

Die Mesh Parameterwerte «mesh_path_refresh_time» und «mesh_hwmp_active_path_timeout» der Linux Implementation definieren den Aktualisierungsintervall. Der Path Timeout bestimmt, wie lange ein Pfad seine Gültigkeit besitzt. Der aktuelle Wert, wie lange ein Pfad noch gültig ist, ist dem Eintrag «EXPTIME» der Forwarding Table zu entnehmen. Die Path Refresh Time bestimmt, wie viele Millisekunden bevor ein Pfad seine Gültigkeit verliert ein neuer Path Request mit gesetztem Target Only Flag generiert wird. Im Path Request ist der Wert des Path Timeout im Field «HWMP Lifetime» ersichtlich.

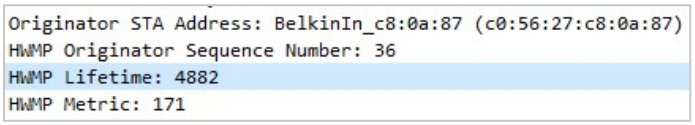


Abbildung 15: HWMP Lifetime Wert auf 4882 gesetzt

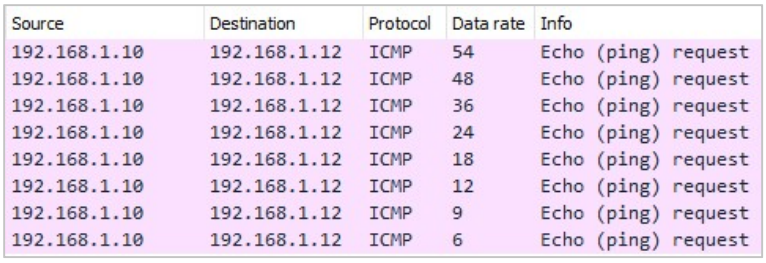
6.4 Ausfall eines Pfades

In der Theorie steht beschrieben, dass ein Path Error ausgelöst wird sobald ein Pfad nicht mehr verfügbar ist. Danach versucht der sendende Mesh Point mit einem Path Request einen neuen Pfad zu finden. Zur Detektierung eines ausgefallenen Pfades steht im WLAN-Standard folgendes geschrieben:

> *«The detection might be triggered by the fact that a mesh STA is unable to forward an MSDU/MMPDU to a next-hop mesh STA» [3, p. 1407]*

Diese Beschreibung lässt einen sehr grossen Spielraum zu, wie ein Ausfall eines Pfades detektiert werden kann. Im Falle der Linux Kernel-Implementation von 802.11s wird dies mit nicht erhaltenen WLAN Acknowledgements erreicht. Die Funktion, die den Path Error auslöst, befindet sich in der Kernel Datei «mesh_hwmp.c» und heisst «mesh_plink_broken» [14]. Diese Funktion wird aufgerufen, falls ein gewisser Schwellwert an nicht erhaltenen WLAN Acknowledgements erreicht wird.

Sendet im WLAN der Sender zum Empfänger ein Paket, wird der Erhalt vom Empfänger mit einem Acknowledgement [15, p. 170] bestätigt. Erhält nun ein Sender auf gesendete Pakete mehrmals kein Acknowledgement, wird dies erfasst und automatisch ein Path Error ausgelöst. Zuerst versucht der Sender den Empfänger aber noch mit tieferen Datenraten zu erreichen. Wenn ein Mesh Point einen anderen Mesh Point bisher immer mit einer Datenrate von 54 Mbit/s erreicht hatte, versucht er das erste Paket auch mit 54 Mbit/s zu senden. Erreicht er die Gegenstation nicht, wird automatisch die Datenrate reduziert. Dieser Vorgang wurde im Wireshark aufgenommen und ist als Screenshot auf Abbildung 16 ersichtlich.



Source	Destination	Protocol	Data rate	Info
192.168.1.10	192.168.1.12	ICMP	54	Echo (ping) request
192.168.1.10	192.168.1.12	ICMP	48	Echo (ping) request
192.168.1.10	192.168.1.12	ICMP	36	Echo (ping) request
192.168.1.10	192.168.1.12	ICMP	24	Echo (ping) request
192.168.1.10	192.168.1.12	ICMP	18	Echo (ping) request
192.168.1.10	192.168.1.12	ICMP	12	Echo (ping) request
192.168.1.10	192.168.1.12	ICMP	9	Echo (ping) request
192.168.1.10	192.168.1.12	ICMP	6	Echo (ping) request

Abbildung 16: WLAN-Datenratenreduzierung

Die Abbildung 17 beschreibt, wie das WLAN Acknowledgement immer als Bestätigung gesendet wird. Dabei sendet der Mesh Point mp10 an den Mesh Point mp12 Daten. Jedes Paket das vom Mesh Point mp10 an den Mesh Point mp12 geht, muss nun von den Empfängern bestätigt werden. Das bedeutet, erhält der Mesh Point mp11 ein Paket vom Mesh Point mp12, so muss er diesen Erhalt mit einem Acknowledgement bestätigen. Der Mesh Point mp11 sendet danach das Paket weiter und der Mesh Point mp12 bestätigt den Erhalt dieses Paketes mit einem Acknowledgement an den Mesh Point mp11.

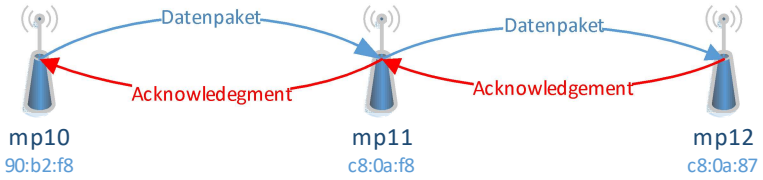


Abbildung 17: WLAN Acknowledgement-Bestätigung

Dieses Acknowledgement wird aber vom Empfänger nur generiert, wenn es ein Unicast-Verkehr ist. Bei Broadcast- oder Multicast-Verkehr wird kein Acknowledgement gesendet, weswegen dort niemals ein Path Error zustande kommen kann. Die Abbildung 18 zeigt die WLAN Acknowledgements in einem Wireshark Trace.

Source	Destination	Protocol	Info
192.168.1.10	192.168.1.12	ICMP	Echo (ping) request id=0x518a
	BelkinIn_c8:0a:8b (c0:56:27:c8:0a:8b) (RA)	802.11	Acknowledgement, Flags=........C
192.168.1.10	192.168.1.12	ICMP	Echo (ping) request id=0x518a
	BelkinIn_90:b2:f8 (94:10:3e:90:b2:f8) (RA)	802.11	Acknowledgement, Flags=........C

Abbildung 18: Wireshark Trace WLAN Acknowledgement

6.5 802.11s Frames

Im 802.11-Standard sind diverse Frame-Arten definiert. Die Mesh Frames sind als Action Frames in der Kategorie Mesh zugewiesen. Solche Wireless LAN Management Frames übernehmen die Verwaltung des Mesh-Netzwerks.

6.5.1 Path Request

Der Path Request wird in der Theorie als Route Request Message (RREQ) bezeichnet. Diese Nachricht wird von einem Mesh Point ausgesendet, wenn er die MAC-Adresse des Empfängers kennt, aber den Pfad zu diesem nicht.

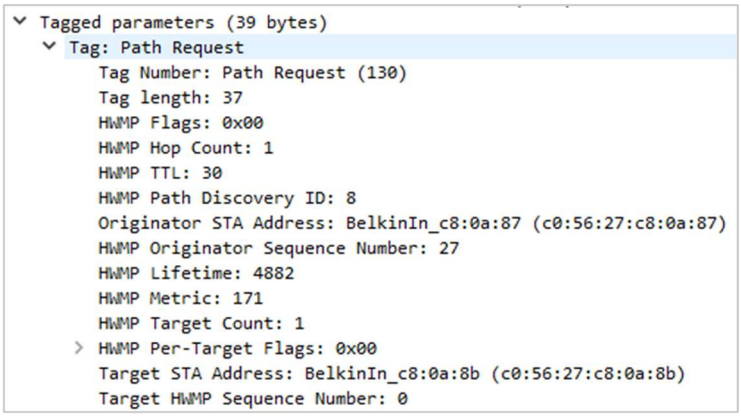


Abbildung 19: Wireshark Trace Path Request Frame

Eine Erklärung zu den verschiedenen Elementen gibt es in der Tabelle 18.

Feld	Beschreibung
Tag Number	Beschreibt, welcher Elementtyp als nächstes im Frame folgt. Die Element ID für den PREQ ist 130 und für den PREP 131.
Tag Length	Wie viele Bit der Path Request umfasst.
HWMP Flags	Flags, die zum Beispiel beim Einsatz des proaktiven HWMP Algorithmus verwendet werden.
HWMP Hop Count	Über wie viele Knoten dieser Path Request schon weitergeleitet wurde.
HWMP TTL	Über wie viele Knoten dieser Path Request noch weitergeleitet werden kann.
HWMP Path Discovery ID	Eindeutige ID, die vom Originator für diesen Path Request festgelegt wurde.
Originator STA Address	MAC-Adresse des Originators.
HWMP Orignator Sequence Number	Sequenz-Nummer des Originators. Damit können die Knoten feststellen, wie aktuell ein Path Request ist.
HWMP Lifetime	Wie lange der Pfad noch Gültigkeit besitzt.
HWMP Metric	Aktueller Metrik Wert des Path Requests.
HWMP Target Count	Wie viele Zielknoten in diesem Path Request gesucht werden. In einem einzigen Path Request kann nach bis zu 20 Zielknoten gleichzeitig gesucht werden.
HWMP Per Target Flags	Spezifisches Target-Attribut pro Zielknoten.
Target STA Address	MAC-Adresse des Zielknotens.
Target HWMP Sequence Number	Sequenz-Nummer der Zielknotenabfrage.

Tabelle 18: Path Request Frame Felder

6.5.2 Path Reply

In der HWMP Theorie wird der Path Reply als Route Reply Message (RREP) bezeichnet. Sobald ein Mesh Point einen Path Request erhält, der für ihn gedacht ist, sendet er einen Path Reply zurück. Wurde im Path Request das Target Only Flag nicht gesetzt, kann auch ein Mesh Point einen Path Reply zurücksenden, der einen gültigen Pfad zum Ziel kennt.

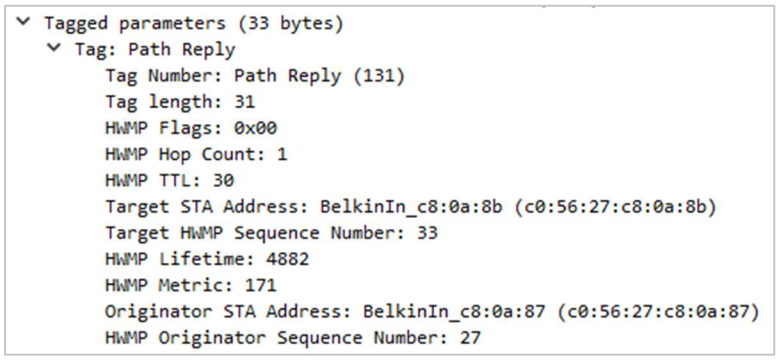


Abbildung 20: Wireshark Trace Path Reply Frame

Beim Path Reply fallen die HWMP Per Target Flags, der HWMP Target Count und die HWMP Path Discovery ID weg. Ansonsten werden die gleichen Felder wie beim Path Request verwendet.

6.5.3 Path Error

Die Auslösung eines Path Errors wird im Kapitel 6.4 genauer erläutert. Im Reason Code steht jeweils beschrieben, warum der Path Error generiert wurde. Im Falle des Wireshark Traces in Abbildung 21 konnte der Sender der Error Message die Target STA mit nicht mehr erreichen.

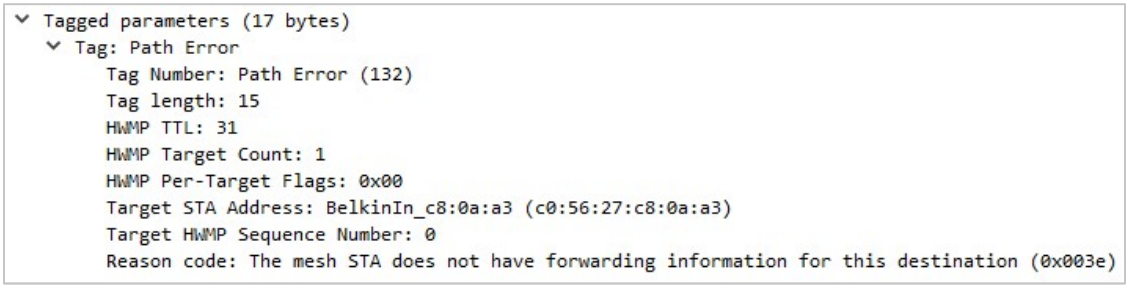


Abbildung 21: Wireshark Trace Path Error Frame

6.5.4 Beacon Frame

Das Beacon Frame wird kontinuierlich von einem Mesh Point ausgesendet, um anderen Mesh Points mit Informationen über diesen zu versorgen. Für das Mesh-Netzwerk interessant sind vor allem die letzten beiden Tags «Mesh ID» und «Mesh Configuration», welche Informationen zum Mesh-Profil beinhalten.

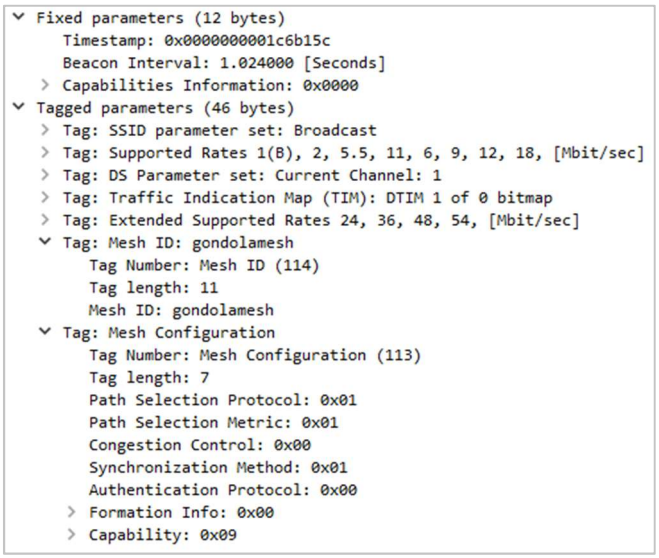


Abbildung 22: Wireshark Trace Beacon Frame

Das Mesh-Profil besteht aus zwei Feldern:

Feld	Beschreibung
Mesh ID	Identifizierung des Mesh-Netzwerkes mit einem Namen.
Mesh Configuration	Enthält diverse weitere Werte, welche die Fähigkeiten des lokalen Mesh Points beschreiben. Zum Beispiel die zu verwendende Metrik oder das eingesetzte Routing-Protokoll.

Tabelle 19: Inhalt Mesh-Profil

6.5.5 Mesh Peering Open

Das Mesh Peering Open Frame enthält ebenfalls das Mesh-Profil.

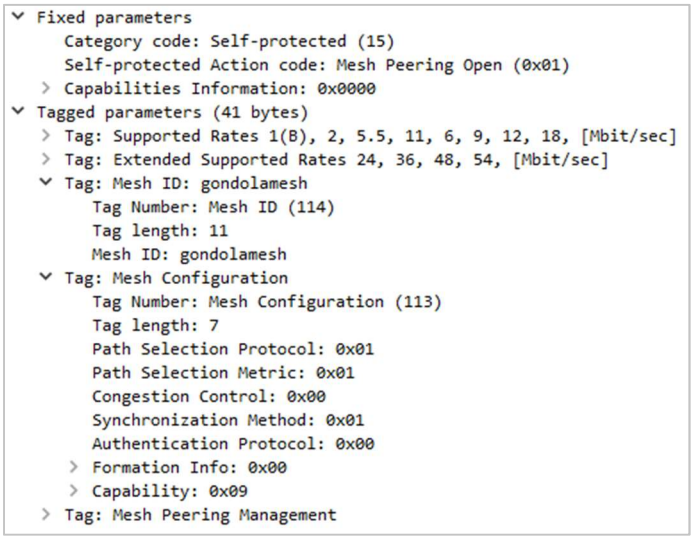


Abbildung 23: Wireshark Trace Mesh Peering Open Frame

6.5.6 Mesh Peering Confirm

Auch das Mesh Peering Confirm Frame enthält das Mesh-Profil.

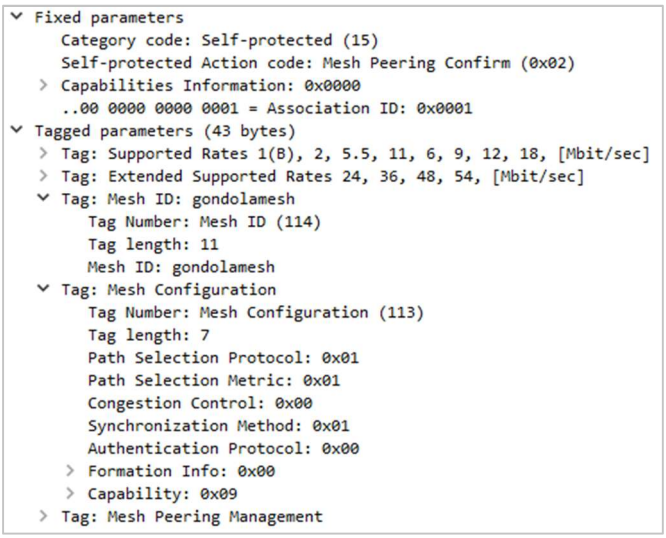


Abbildung 24: Wireshark Trace Mesh Peering Confirm Frame

6.5.7 Mesh Peering Close

Das Mesh Peering Close Frame enthält lediglich die Mesh ID. Vielmehr wird auch nicht benötigt, da für einen Verbindungsabbruch keine zusätzlichen Informationen benötigt werden.

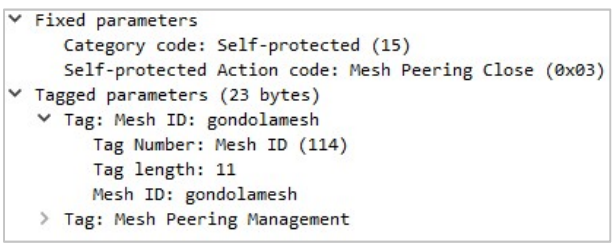


Abbildung 25: Wireshark Trace Mesh Peering Close Frame

6.6 Security

Die beschriebene Art, wie ein Peering aufgebaut wird, bietet keinerlei Sicherheit. Um Teilnehmer in einem Mesh-Netzwerk zu werden, braucht man lediglich dasselbe Mesh-Profil wie die anderen Mesh Points im Netzwerk. Dieses kann durch Sniffen im WLAN problemlos abgehört und selbst verwendet werden. Um diesem Sicherheitsaspekt entgegen zu wirken, definiert 802.11s eine zweite Peering-Möglichkeit, das sogenannte Authenticated Mesh Peering Exchange (AMPE) [16]. Als Authentifizierungsmechanismus kommt dabei das Protokoll Simultaneous Authentication of Equals (SAE) zum Einsatz. Das Prinzip von SAE basiert darauf, dass auf den Mesh Points, die eine gesicherte Verbindung untereinander aufbauen wollen, ein gemeinsames Passwort hinterlegt ist.

7 Gondelbahn mit 802.11s

802.11s bildet eine solide Grundlage, um das Gondelbahnproblem zu lösen. Es hat jedoch auch seine Grenzen und es bedarf womöglich einer eigenen Implementierung, um das Problem effektiv zu lösen. Daher soll in diesem Kapitel mit Hilfe von Tests herausgefunden werden, worin die Grenzen bei 802.11s liegen und wie diese im Zusammenhang mit dem Gondelbahnproblem stehen. Zur Durchführung dieser Tests wird dabei ein Testlabor aufgebaut.

7.1 Aufbau eines 802.11s Testlabors

Für den Aufbau des Testlabors werden Hardware-Komponenten verwendet, die bereits vorhanden sind. Daher steht für den Aufbau folgende Hardware zur Verfügung:

Anzahl	Hardware	Beschreibung
5	Raspberry Pi 2 Model B	ARM Cortex-A7 (4x 900MHz) 1 GB RAM 10/100 Mbit Ethernet
5	Linksys AE3000	USB WLAN Adapter mit Chipset Ralink RT3573 (Unterstützung für Mesh-Netzwerke)
5	TL-WN722N	USB WLAN Adapter mit Chipset Atheros AR9271L (Unterstützung für Mesh-Netzwerke)
2	AirPcap NX	USB Wireless Sniffer für WLAN Protokollanalyse.

Tabelle 20: Hardware für Testlabor

Mithilfe der Hardwarekomponenten wird ein Wireless Mesh Netzwerk auf Basis von 802.11s aufgebaut. Um dies zu realisieren und auch Performance-Tests durchführen zu können, wird ausserdem folgende Software auf den Raspberry Pis installiert:

Software	Version	Beschreibung
Archlinux	4.4.27-1-ARCH	Betriebssystem.
IPerf	2.0.9	Messen von TCP- und UDP-Datenströmen im Netzwerk.
linphone	3.10.2	Softphone zum Testen von VoIP.
mplayer	1.3.0	Ermöglicht Streaming von Musikdiensten.
vlc	2.2.4	Mediaplayer zum Senden von Streams.
wget	1.18	Ermöglicht Download von Files über HTTP.

Tabelle 21: Software für Testlabor

Um die Management- und Control-Frames von WLAN analysieren zu können, die bei der Verwendung von 802.11s zum Einsatz kommen, wird auf einem Notebook Wireshark installiert. Zur Erkennung dieser speziellen Frames wird ausserdem der AirPcap NX Dongle benötigt.

7.1.1 Netzwerkschema

Das Labor wird anhand von folgendem Netzwerkschema aufgebaut:

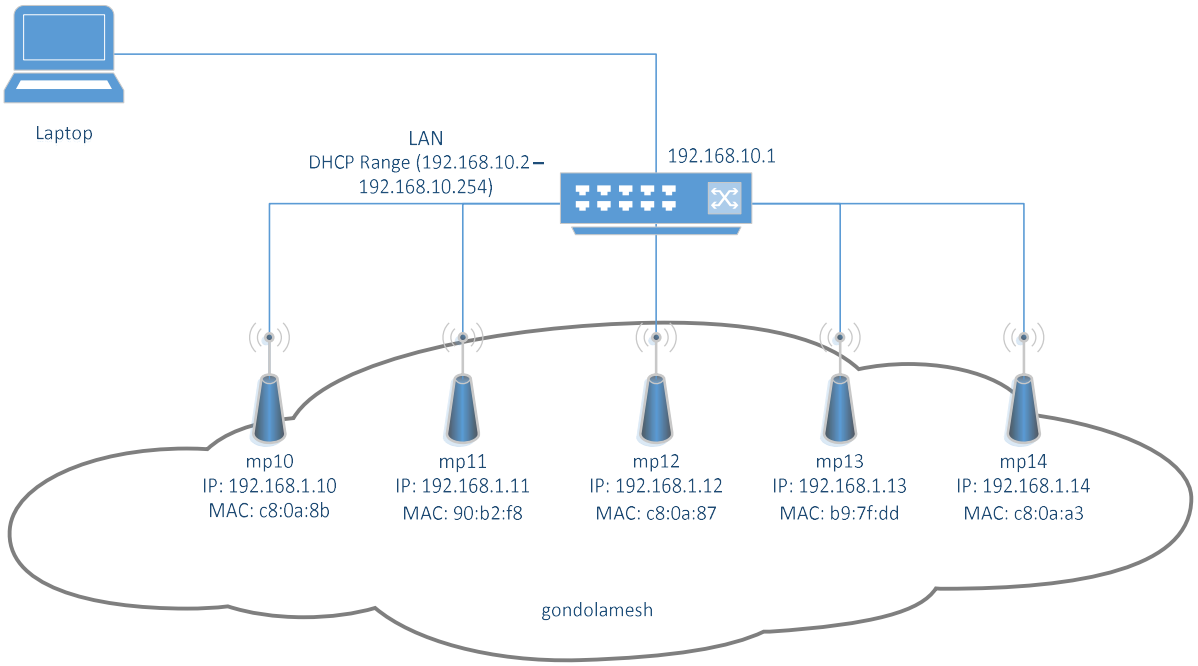


Abbildung 26: Testlabor Netzwerkschema

Die Ethernet-Schnittstellen der Raspberry Pis erhalten über einen DHCP-Server IP-Adressen aus dem Adressraum 192.168.10.0/24. Über die dadurch vergebenen Adressen können die Raspberry Pis mittels SSH konfiguriert werden. Die Schnittstelle wurde bewusst auf DHCP konfiguriert, um einen einfachen Wechsel an ein bestehendes Netzwerk mit Internetzugang zu ermöglichen, falls zusätzliche Installationspakete aus dem Internet heruntergeladen werden müssen.

Damit innerhalb des Mesh-Netzwerkes einfacher Tests durchgeführt werden können, wurden den WLAN-Schnittstellen der Raspberry Pi fixe IP-Adressen und Hostnames vergeben.

Während der Testdurchführung müssen die Raspberry Pis nicht dauernd mit dem LAN verbunden sein. Diese Schnittstelle dient lediglich der Konfiguration. Sobald die Raspberry Pis fertig konfiguriert sind, kann diese Verbindung unterbrochen und der Raspberry Pi im freien Raum betrieben werden.

7.1.2 Aufbau des Mesh

Da 802.11s seit 2012 bereits im Linux Kernel implementiert ist, muss keine zusätzliche Software auf den Raspberry Pis installiert werden, um ein Mesh-Netzwerk zu betreiben. Zur Inbetriebnahme reichen wenige Linux Befehle. Damit die Befehle nicht bei jedem Neustart eines Raspberry Pis manuell ausgeführt werden müssen, wird ein Startup-Skript verwendet, welches die Befehle nacheinander ausführt.

```
#!/bin/bash

ipSuffix=${HOSTNAME:2}

iw dev wlan0 interface add $HOSTNAME type mp

ifconfig $HOSTNAME 192.168.1.$ipSuffix

iw dev $HOSTNAME mesh join gondolamesh
```

In einem ersten Schritt wird der WLAN-Schnittstelle eine neue virtuelle Schnittstelle hinzugefügt, die als Mesh Point dienen soll. Danach wird dieser eine fixe IP-Adresse zugewiesen. In einem letzten Schritt wird mittels des Befehls «mesh join gondolamesh» der Schnittstelle mitgeteilt, dass sie fortan Teilnehmer im Mesh-Netzwerk mit der ID «gondolamesh» ist.

Wird dieses Skript auf allen Raspberry Pis ausgeführt, verbinden sich diese automatisch zu einem Mesh-Netzwerk.

7.1.3 Konfiguration eines Mesh Point Portal

Während des Aufbaus des Testlabors wurde auch ein Mesh Point Portal in Betrieb genommen. Ein Mesh Point Portal ist ein normaler Mesh Point, der zusätzlich die Funktion bietet, das Mesh-Netzwerk mit einem normalen LAN zu verknüpfen. Er dient dabei als Netzwerkbrücke zwischen diesen zwei Netzwerken.

Um ein solches Mesh Point Portal zu konfigurieren wird folgende zusätzliche Konfiguration auf einem Mesh Point benötigt:

```
#!/bin/bash

ip_suffix=${HOSTNAME:3}

iw dev wlan0 interface add $HOSTNAME type mp

brctl addbr br0

brctl stp br0 off

brctl addif br0 eth0

brctl addif br0 $HOSTNAME

ifconfig $HOSTNAME down

ifconfig eth0 down

ifconfig $HOSTNAME 0.0.0.0 up

ifconfig eth0 0.0.0.0 up
```

```
ifconfig br0 192.168.1.$ip_suffix

iw $HOSTNAME mesh join gondolamesh
```

Hierbei wird eine Netzwerkbrücke «br0» unter Linux angelegt, welche die LAN Schnittstelle «eth0» mit der Mesh Schnittstelle «$HOSTNAME» verbindet. Befindet sich im verknüpften LAN ausserdem ein Gateway ins Internet, so kann dieser auf den anderen Mesh Points hinzugefügt werden, sodass diese ebenfalls Zugang ins Internet haben.

```
$: route add default gw 192.168.1.1
```

7.2 Distanzmessung

In einem ersten Test soll nun festgestellt werden, wie weit die Mesh Points voneinander entfernt sein dürfen, um noch sinnvoll Daten übertragen zu können. In der Anforderungsanalyse wurde definiert, dass über mindestens 50m noch eine stabile Verbindung möglich sein muss.

7.2.1 Testaufbau

Zwei Mesh Points mp13 und mp14 sind miteinander verbunden.

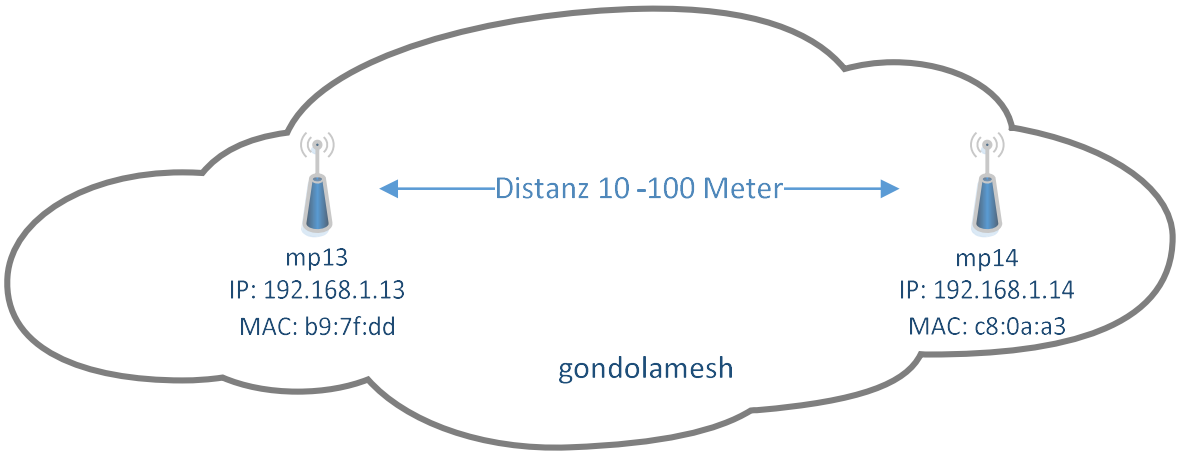


Abbildung 27: Testaufbau Distanzmessung

Der Mesh Point mp13 entfernt sich nun in Zehn-Meter-Schritten vom Mesh Point mp14. Bei jedem Schritt werden die folgenden Werte gemessen.

	Grösse	Bemerkung
Bandbreite TCP	Mbit/s	Iperf Messung.
Bandbreite UDP	Mbit/s	Iperf Messung.
Durchschnittliche Antwortzeit	Millisekunden	Durchschnitt von zehn Ping Nachrichten.
Signalstärke	dbm	Angabe vom Linux Kommandozeilen-werkzeug «iw».
Metrik	-	Angabe vom Linux Kommandozeilen-werkzeug «iw».

Tabelle 22: Testkriterien Distanzmessung

Der Test wird in einer sehr ländlichen Umgebung mit wenigen Störquellen durchgeführt. In den WLAN-Adaptern sind Rundstrahlantennen eingebaut. 802.11n ist nicht aktiviert.

7.2.2 TCP und UDP Messung

In der nachfolgenden Grafik ist zu erkennen, dass insbesondere beim Schritt von 60 auf 70 Meter ein grosser Unterschied bei der Bandbreite entsteht. Sowohl die stabilere TCP-Messung wie auch die UDP-Messung zeigen dieses Verhalten. Mit einer Bandbreite von 8.7 Mbit/s (TCP) und 11.5 Mbit/s (UDP) bei 60 Metern ist noch problemlos eine Datenübertragung möglich. Nach 100 Metern ist die Verbindung nur noch ganz schlecht und es kann keine stabile Verbindung zwischen den Mesh Points mehr aufgebaut werden.

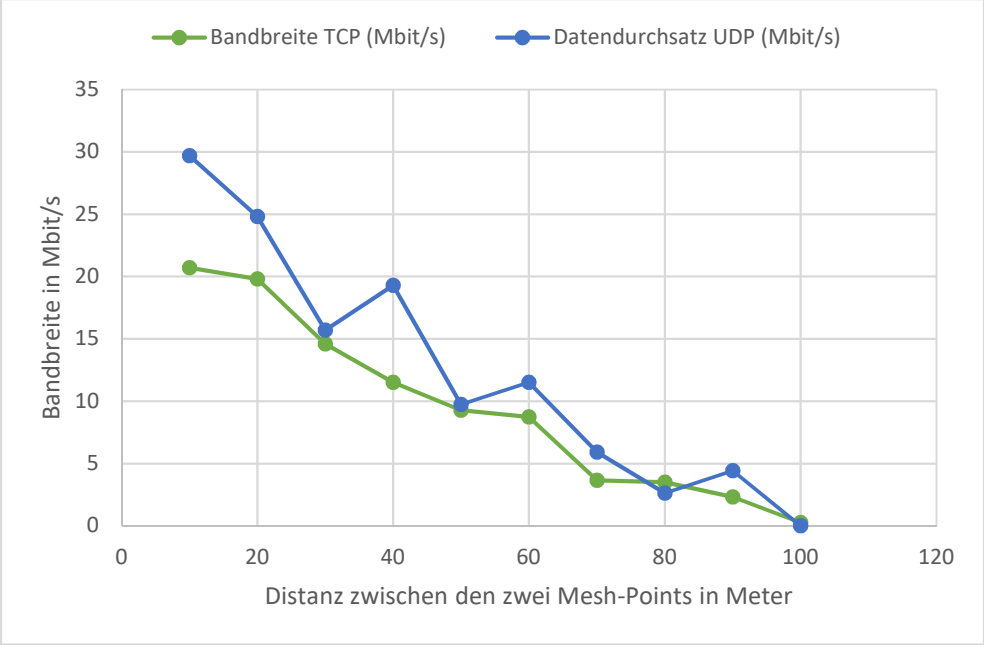


Abbildung 28: Messergebnis TCP / UDP

7.2.3 Antwortzeit (Ping)

Die durchschnittliche Antwortzeit nimmt mit der grösseren Distanz zu. Die Antwortzeiten liegen aber immer in einem akzeptablen Bereich. Auch in dieser Grafik steigt die Antwortzeit sprunghaft zwischen 60 und 70 Metern an.

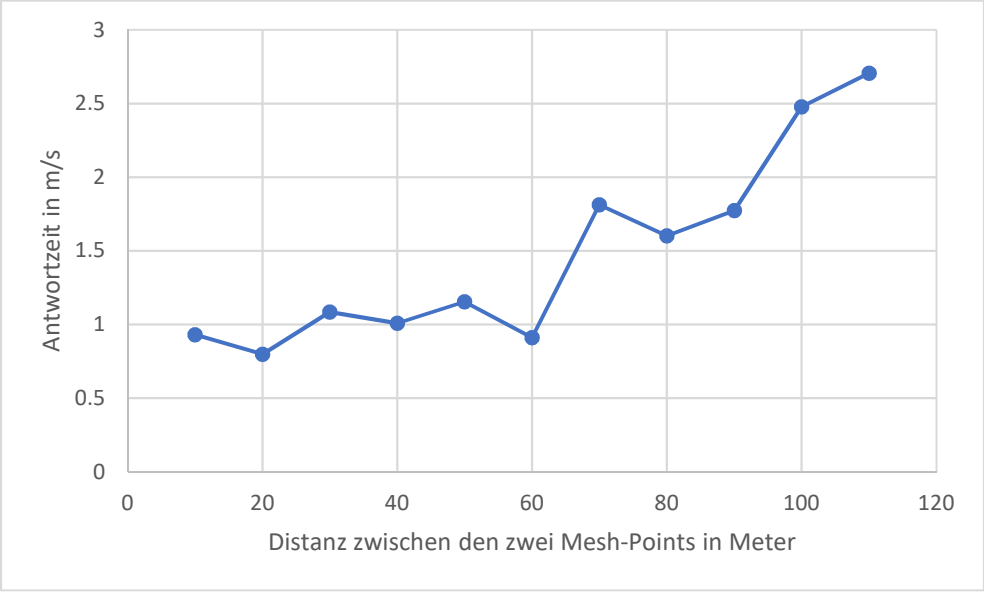
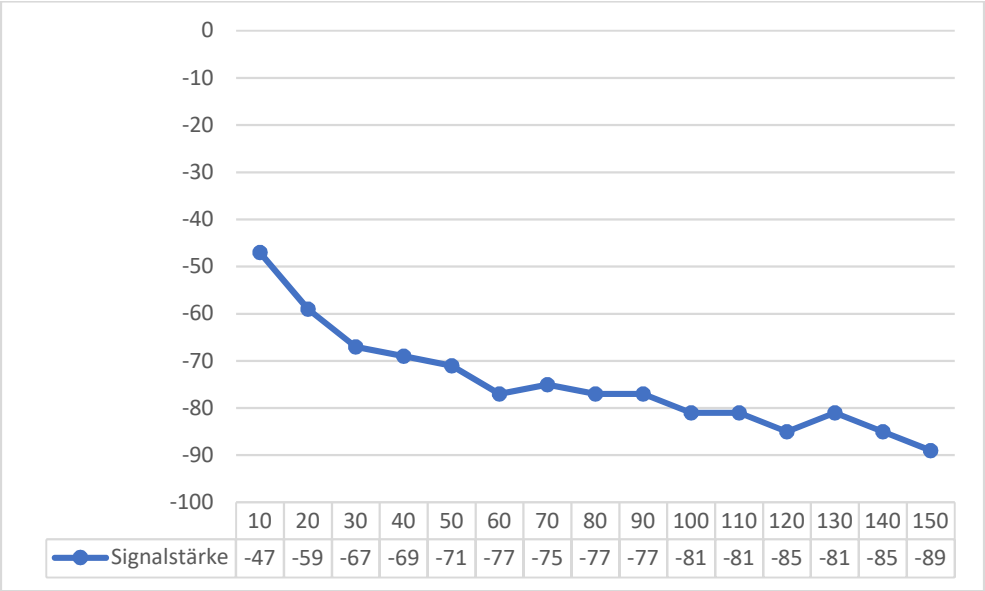


Abbildung 29: Messergebnis Antwortzeit

7.2.4 Signalstärke

Die beiden Mesh Points sehen sich über eine sehr grosse Distanz. Dies hat den Grund, dass die Mesh Points die Beacons von anderen Stationen noch sehr lange empfangen können. Die Beacons wurden im Test mit einer Datenrate von 1 Mbit/s ausgesendet. Dies hat zur Folge, dass die Reichweite eines Beacons sehr gross ist. Sobald jedoch die Signalstärke unter -80dbm sinkt, ist keine stabile Datenübertragung mehr möglich.

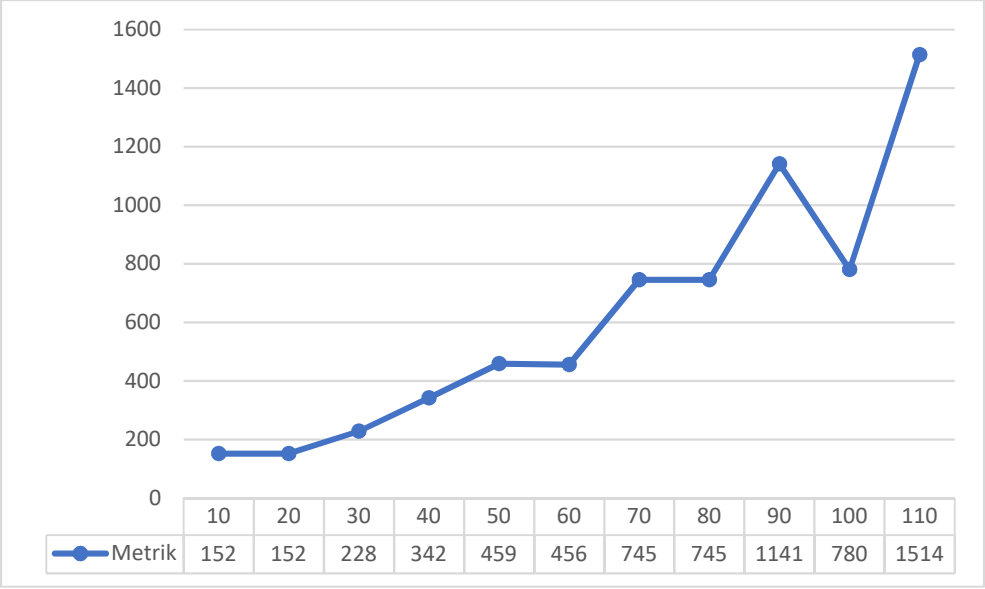


	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
Signalstärke	-47	-59	-67	-69	-71	-77	-75	-77	-77	-81	-81	-85	-81	-85	-89

Abbildung 30: Messergebnis Signalstärke

7.2.5 Metrik

Auch beim Metrik-Verlauf ist ein erhöhter Anstieg zwischen Meter 60 und 70 ersichtlich. Danach wird die Metrik immer weniger aussagekräftiger.



	10	20	30	40	50	60	70	80	90	100	110
Metrik	152	152	228	342	459	456	745	745	1141	780	1514

Abbildung 31: Messergebnis Metrik

7.2.6 Fazit

Das Fazit der verschiedenen Messungen ist, dass die Gondeln nicht mehr als 60 Meter auseinander sein sollten. Eine Verbindung über grössere Distanzen ist noch möglich aber die Stabilität könnte darunter leiden. Im Test wurden Rundstrahlantennen verwendet, mit Richtstrahlantennen könnte die maximale Distanz sicher noch vergrössert werden.

Werden die Signalstärke und der Datendurchsatz TCP direkt miteinander verglichen, ergibt sich folgendes Diagramm. Dieses ist Distanz-unabhängig und zeigt auf, dass der Datendurchsatz bis Signalstärke -80 dBm noch gut ist. Somit kann ausgesagt werden, dass zwischen zwei Mesh Points mindestens eine Signalstärke von -80 dbm herrschen muss, damit ein akzeptabler Datendurchsatz erreicht wird.

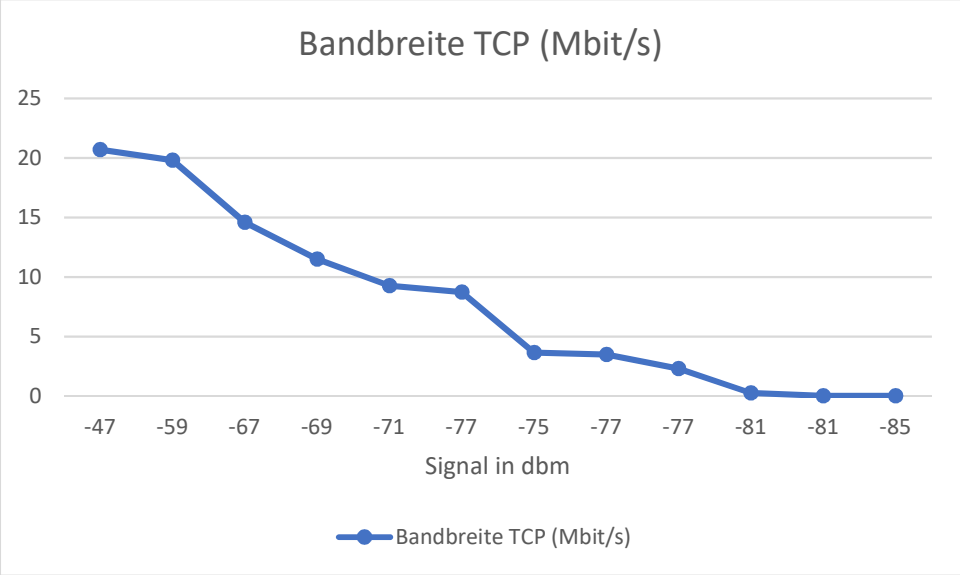


Abbildung 32: Abnahme der Bandbreite bei schlechter werdenden Verbindung

7.3 Performance

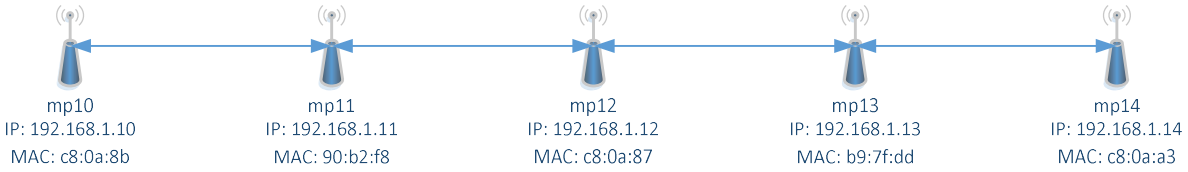
Damit Sprachsignale und Daten ohne Verzögerungen und Störungen übertragen werden können, muss im Mesh-Netzwerk zwischen zwei Mesh Points eine gewisse Bandbreite vorhanden sein. Daher sollten folgende Werte im ganzen Mesh-Netzwerk zur Verfügung stehen:

	Wert	Bemerkung
Bandbreite	Minimal 1 Mbit/s	Für eine erfolgreiche Sprachübertragung sind mindestens 128 kbit/s nötig. Da jedoch auch eine bidirektionale Verbindung benötigt wird und allenfalls auch andere Medien übertragen werden sollen, wird von einer minimalen Bandbreite von 1 Mbit/s ausgegangen.
Antwortzeit	Maximal 300 Millisekunde	Falls die Gondeln mit einer Gegensprechfunktion ausgestattet werden sollen, darf das Netzwerk nur eine gewisse maximale Antwortzeit haben. Eine Verzögerung von 300ms ist in diesem Fall noch zumutbar.

Tabelle 23: Performance Anforderungen an Gondelbahn

7.3.1 Performance über mehrere Hops

Sobald die Daten über mehrere Hops gesendet werden, wird der Datendurchsatz immer kleiner. Wieviel dies aber effektiv ist, wird mit dem folgenden Test herausgefunden.



Damit der Test realisiert werden kann, muss den einzelnen Mesh Points mitgeteilt werden, zwischen welchen von ihnen Daten ausgetauscht werden können. Da die Signalstärke von WLAN recht unberechenbar ist und somit Verbindungen zu Stande kommen können, welche nicht gewünscht werden, müssen einzelne Verbindungen zwischen Mesh Points explizit geblockt werden. Dies kann in Linux mit Hilfe des Befehls «plink_action block» realisiert werden, welcher im Kapitel 6.1 bereits genauer beschrieben ist.

Um nun die Bandbreite zu messen, werden mit Iperf TCP-Verbindungen zwischen den einzelnen Mesh Points aufgebaut. Das Messergebnis sieht folgendermassen aus:

Direkte Verbindung:

```
iperf -c 192.168.1.11

[  3]  0.0-10.0 sec  20.2 MBytes  16.9 Mbits/sec
```

1 Hop dazwischen:

```
iperf -c 192.168.1.12

[  3]  0.0-10.0 sec  11.4 MBytes  9.54 Mbits/sec
```

2 Hops dazwischen:

```
iperf -c 192.168.1.13

[  3]  0.0-10.0 sec  5.88 MBytes  4.92 Mbits/sec
```

3 Hops dazwischen:

```
iperf -c 192.168.1.14

[  3]  0.0-10.0 sec  3.18 MBytes  2.66 Mbits/sec
```

7.3.2 Fazit

Generell kann gesagt werden, dass die Bandbreite mit jedem Hop fast um die Hälfte reduziert wird. Dies erkennt man am folgenden Diagramm. Da sich in diesem Test aber alle Mesh Points gesehen hatten und nicht über eine grössere Distanz getrennt waren, belegten alle Mesh Points auch zur gleichen Zeit den gleichen WLAN-Kanal.

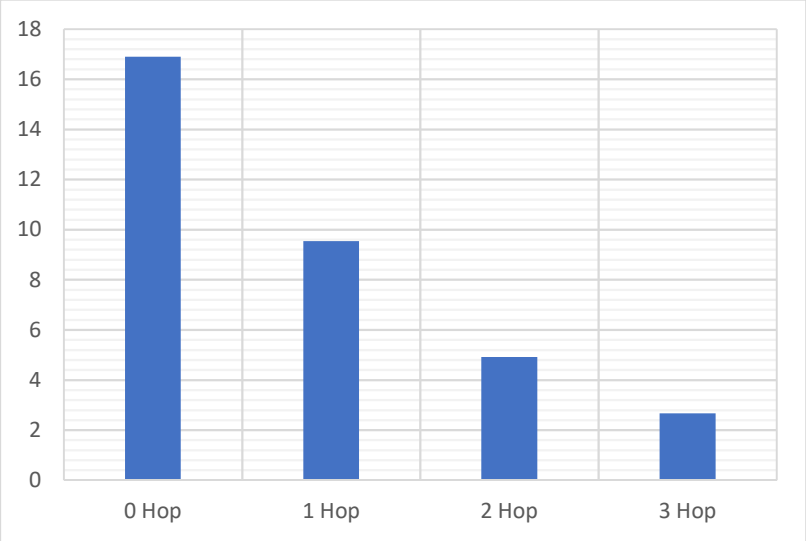


Abbildung 33: Abnahme der Bandbreite nach Hops

7.4 Dynamik

Das wesentliche Problem des Mesh-Netzwerkes der Gondelbahn ist die hohe Dynamik. Wie das Routing-Protokoll HWMP mit dieser Dynamik umgeht, wird in diesem Kapitel erklärt. Da das System nur bedingt als Ganzes getestet werden kann, wird es dabei in verschiedene Szenarien unterteilt.

Nicht jedes Szenario lässt sich exakt im Testlabor nachstellen, weshalb vereinzelt die Szenarien theoretisch mit HWMP beschrieben werden. Dies betrifft vor allem die ersten beiden Tests «Kreuzen in voller Fahrt» und «Fährt in Station».

7.4.1 Szenario Kreuzen in voller Fahrt

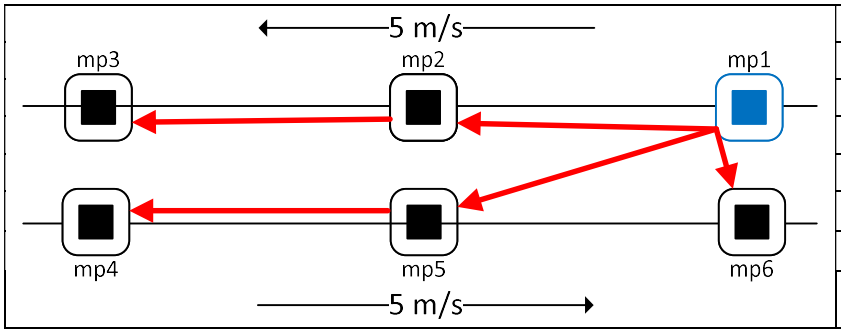
Die Gondel mit dem Mesh Point mp1 fährt mit der vollen Geschwindigkeit von der Berg- zur Talstation. Auf der Fahrt kreuzt sie dabei mehrere Gondeln.

7.4.1.1 Erwartung

Der Mesh Point mp1 in der Gondel verbindet sich mit allen anderen Knoten in seinem Umkreis bzw. führt ein Peering mit diesen durch. Werden nun Daten im Netzwerk gesendet, wird die Forwarding Table aufgebaut. Je nach Situation verändert sich die Forwarding Table.

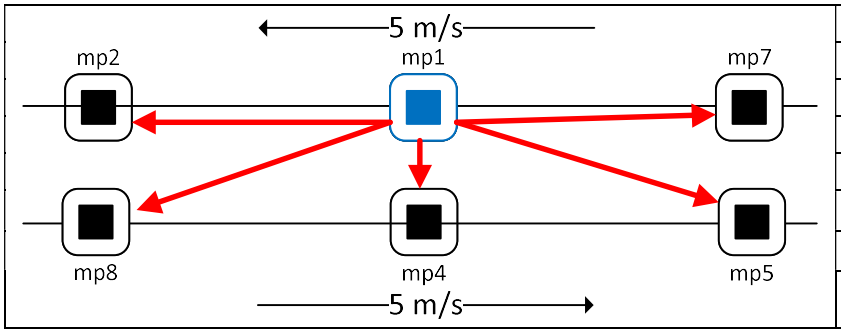
Die nachfolgenden Grafiken stellen den optimalen Fall aus der Sicht von mp1 dar. Das heisst, es wird davon ausgegangen, dass z.B. mp1 nicht direkt mit mp3 verbunden ist, da sich diese nicht sehen. Mesh Point mp1 hat lediglich zu seinen direkten Nachbarn eine Verbindung.

Mesh Point 1 kreuzt sich mit dem Mesh Point 6:



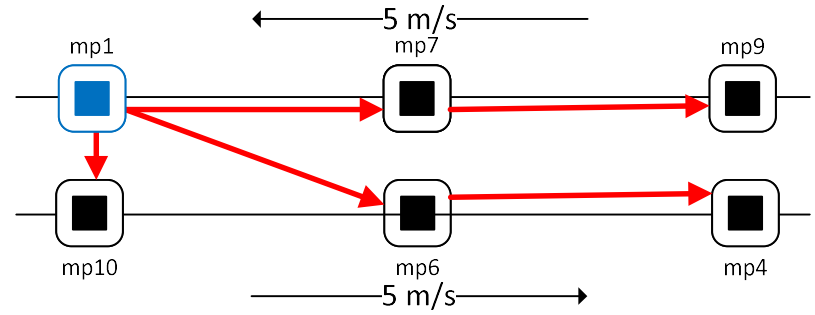
Forwarding Table mp1	
Destination	**Next Hop**
mp2	mp2
mp3	mp2
mp5	mp5
mp4	mp5
mp6	mp6
[weitere Einträge]	

Mesh Point 1 kreuzt sich mit dem Mesh Point 4:



Forwarding Table mp1	
Destination	**Next Hop**
mp2	mp2
mp7	mp7
mp5	mp5
mp4	mp4
mp8	mp8
[weitere Einträge]	

Mesh-Point 1 kreuzt sich mit dem Mesh-Point 10:



Forwarding Table mp1	
Destination	**Next Hop**
mp7	mp7
mp9	mp7
mp10	mp10
mp6	mp6
mp4	mp6
[weitere Einträge]	

7.4.1.2 Schlussfolgerung

Wie das Szenario aufzeigt, verbindet sich mp1 durch Verwendung von 802.11s immer mit seinen direkten Nachbarn, also auch denjenigen, welche auf der Gegenseite nur kurz sichtbar sind. Wird ein neuer Pfad gesucht und besteht ein aktives Peering zwischen zwei gegenüberliegenden Mesh Points, so wird der Path Request auch an diesen gesendet. Da die Pfadsuche nur wenige Millisekunden dauert, könnte ein gültiger Pfad über einen gegenüberliegenden Mesh Point aufgebaut werden. Der gegenüberliegende Mesh Point ist aber nur kurz der beste Pfad. Er bleibt aber solange gültig, bis der Timer in der Forwarding Table ausläuft.

Die Gültigkeit des Pfades kann über den Parameter «mesh_hwmp_active_path_timeout» eingestellt werden. Ist dieser Wert sehr klein, wäre ein Pfad über den gegenüberliegenden Knoten vernachlässigbar. Dies würde aber zur Folge haben, dass das Protokoll ständig versucht, den Pfad zu aktualisieren. Durch einen kleinen Wert wird das Netzwerk aber sehr instabil, da andauernd Pfade gewechselt werden und der Datenverkehr dadurch gestört wird. Die Anzahl der Management Frames würde sich zusätzlich drastisch erhöhen. Es wäre sinnvoller, die Verbindung zum gegenüberliegenden Knoten zu blockieren, damit die Path Requests diese gar nicht erst erreichen. Es soll nur möglich sein, mit den jeweiligen Nachbarn in gleicher Fahrtrichtung zu kommunizieren. Dies kann nicht standardmässig konfiguriert werden und muss mit einer eigenen Implementierung gelöst werden.

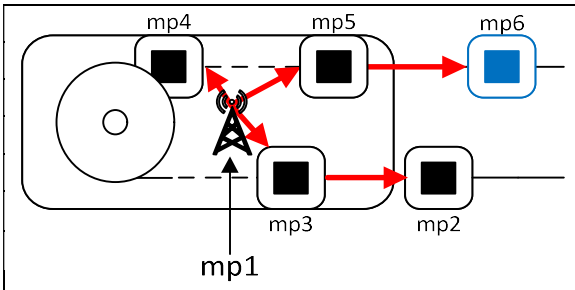
7.4.2 Szenario fährt in Station

Die Gondel fährt mit voller Geschwindigkeit in die Station hinein. Damit die Passagiere austeigen können, wird sie vom Seil ausgekoppelt und fährt langsamer durch die Station. Durch das Verlangsamen sind nun mehrere Gondeln näher zusammen. Sind alle Passagiere eingestiegen, koppelt die Gondel wieder beim Seil ein und fährt mit voller Geschwindigkeit aus der Station.

7.4.2.1 Erwartung

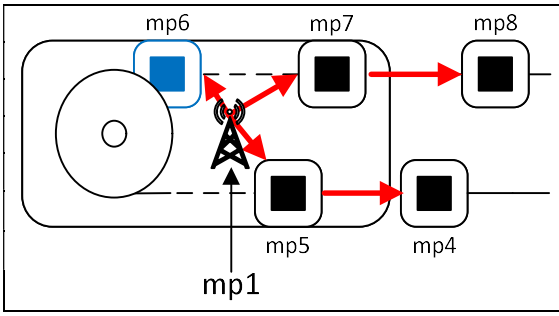
Sobald die Gondel in die Station einfährt, verbindet sich der Mesh Point mp6 in der Gondel mit dem fixen Mesh Point mp1 der Station. Das bedeutet, in der Forwarding Table ist der Destination Hop und der Next Hop der gleiche Mesh Point. Sobald die Gondel die Station wieder verlässt, ist der Mesh Point mp6 in der Gondel nicht mehr direkt verbunden. Die nachfolgenden Grafiken sollen diesen Vorgang verdeutlichen.

Knoten nähert sich der Station:



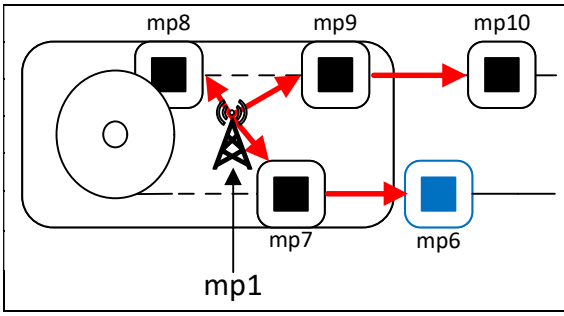
Forwarding Table mp1	
Destination	**Next Hop**
mp5	mp5
mp4	mp4
mp3	mp3
mp6	mp5
mp2	mp3
[weitere Einträge]	

Knoten befindet sich in der Station:



Forwarding Table mp1	
Destination	**Next Hop**
mp5	mp5
mp4	mp5
mp6	mp6
mp7	mp7
mp8	mp7
[weitere Einträge]	

Knoten entfernt sich aus der Station:



Forwarding Table mp1	
Destination	**Next Hop**
mp6	mp7
mp7	mp7
mp8	mp8
mp9	mp9
mp10	mp9
[weitere Einträge]	

7.4.2.2 Schlussfolgerung

Es kann gut sein, dass dadurch, das sich mehrere Mesh Points gleichzeitig in der Station befinden, der beste Pfad zum fixen Mesh Point nicht direkt, sondern über einen zusätzlichen Hop führt. Dies ist dadurch gegeben, das HWMP den besten Pfad anhand der Metrik bestimmt. Diese Konstellation entsteht beispielsweise dann, wenn die Ausbreitung des WLAN-Signals zum Empfänger derart schlecht ist, dass der Pfad mit der besten Metrik über einen zusätzlichen Hop führt. Im Falle des Gondelbahn Problems stellt dies jedoch kein erhebliches Hindernis dar, lediglich die Bandbreite würde reduziert werden. Es macht darum auch keinen Sinn, immer direkt mit dem fixen Mesh Point zu verbinden, da dies nicht unbedingt die beste Verbindung ist.

7.4.3 Szenario Zielknoten verlässt das Netzwerk

Bei diesem Szenario sind drei Mesh Points miteinander verbunden. Mesh Point 1 sieht Mesh Point 3 nur indirekt über Mesh Point 2. Nach einer gewissen Zeit wird Mesh Point 3 aus dem Netzwerk entfernt. Während des Entfernens werden Daten in Form von einem Ping übertragen.

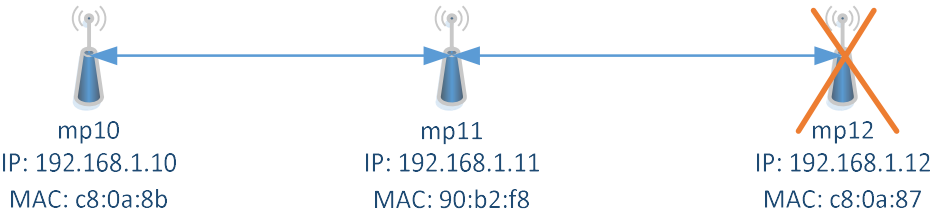


Abbildung 34: Topologie-Szenario Zielknoten verlässt das Netzwerk

Damit dieses Szenario getestet werden kann, dürfen die Mesh Points mp10 und mp12 nicht direkt miteinander verbunden sein. Da dies durch die Ausbreitung der WLAN-Signale schwierig zu realisieren ist, wird die Verbindung zwischen diesen, wie bereits in anderen Tests, beidseitig geblockt.

7.4.3.1 Konfiguration

Konfiguration mp10:

```
iw dev mp10 station set c0:56:27:c8:0a:87 plink_action block
```

Konfiguration mp12:

```
iw dev mp12 station set c0:56:27:c8:0a:8b plink_action block
```

Als Beispiel und gleichzeitig zur Verifikation der Konfiguration werden die Forwarding Tables jedes Mesh Points aufgelistet. Lediglich der Mesh Point mp11 sollte direkt mit beiden Mesh Points verbunden sein.

Forwarding Table mp10:

DEST ADDR	NEXT HOP	IFACE	SN	METRIC	QLEN	EXPTIME	DTIM	DRET	FLAGS
94:10:3e:90:b2:f8	94:10:3e:90:b2:f8	mp10	0	152	0	0	0	0	0x10
c0:56:27:c8:0a:87	94:10:3e:90:b2:f8	mp10	28	323	0	0	0	0	0x14

Forwarding Table mp11:

DEST ADDR	NEXT HOP	IFACE	SN	METRIC	QLEN	EXPTIME	DTIM	DRET	FLAGS
c0:56:27:c8:0a:8b	c0:56:27:c8:0a:8b	mp11	34	171	0	0	0	0	0x10
c0:56:27:c8:0a:87	c0:56:27:c8:0a:87	mp11	28	171	0	0	0	0	0x14

Forwarding Table mp12:

DEST ADDR	NEXT HOP	IFACE	SN	METRIC	QLEN	EXPTIME	DTIM	DRET	FLAGS
94:10:3e:90:b2:f8	94:10:3e:90:b2:f8	mp12	0	152	0	0	0	0	0x10
c0:56:27:c8:0a:8b	94:10:3e:90:b2:f8	mp12	34	323	0	0	100	0	0x14

7.4.3.2 Erwartung

Sobald der Mesh Point mp12 abgeschaltet wird, gehen von ihm keine Beacon Frames mehr aus. Der Mesh Point mp11 bemerkt dies und die Peering-Informationen werden nicht mehr aktualisiert. In der Neighbor Table ist dies erkennbar durch das Ansteigen der «inactive time» und dem Gleichbleiben aller anderen Werte. Der Mesh Point mp11 sendet dem Mesh Point mp10 eine Meldung, dass der Mesh Point mp12 nicht mehr über ihn erreichbar ist. Der Mesh Point mp10 reagiert mit einem neuen Path Request (Broadcast). Mesh Point mp11 empfängt diesen Request und macht ebenfalls einen Path Request (Broadcast), um einen neuen Pfad zu finden. Da immer noch kontinuierlich Ping-Anfragen vom Mesh Point mp10 gesendet werden, wird dieser Vorgang mit dem jedem Ping-Versuch wiederholt.

7.4.3.3 Verifikation

Die Erwartungen wurden in diesem Szenario nicht ganz erfüllt. Die Peering-Informationen auf dem Mesh Point mp11 werden richtigerweise nicht mehr aktualisiert und die «inactive time» steigt an, jedoch sendet der Mesh Point mp11 erst nach ca. 15 Sekunden eine Meldung, dass dieser nicht mehr erreichbar ist.

Beim Analysieren mit Wireshark wurde folgendes Verhalten festgestellt.

1	BelkinIn_90:b2:f8	Broadcast		802.11	112 Beacon frame, SN=547
	BelkinIn_c8:0a:8b	Broadcast		802.11	112 Beacon frame, SN=637
2	192.168.1.10	192.168.1.12		ICMP	160 Echo (ping) request
		BelkinIn_c8:0a:8b (c0:56:27:c8:0a:8b) (RA)		802.11	40 Acknowledgement, Flag
3	192.168.1.10	192.168.1.12		ICMP	160 Echo (ping) request
	192.168.1.10	192.168.1.12		ICMP	160 Echo (ping) request
	192.168.1.10	192.168.1.12		ICMP	160 Echo (ping) request

Abbildung 35: Wireshark Trace Ping schlägt fehl

1. Mesh Point mp10 und Mesh Point mp12 senden ihre Beacon Frames aus. Das Beacon Frame vom Mesh Point mp12 fehlt.
2. Mesh Point mp10 startet einen Ping request. Der Ping Request wird vom Mesh Point mp11 empfangen und der Erhalt mit einem Acknowledgement bestätigt.
3. Mesh Point mp11 versucht nun den Ping Request weiter an den Mesh Point mp12 weiterzuleiten, dies jedoch erfolglos. Dies passiert, weil der Mesh Point mp11 den Mesh Point mp12 in seiner Forwarding Table eingetragen hat. Das Peering ist ebenfalls noch offen, nur die Inaktivitätszeit erhöht sich ständig. Dies wäre der Indikator dafür, dass der Mesh Point mp12 nicht mehr aktiv im Netzwerk ist.

Das Verhalten der verzögerten Ausfallmeldung lässt sich dadurch erklären, dass Ping nicht innert kürzester Zeit genügend Pakete generiert, um einen Schwellwert an nicht bestätigten Paketen zu erreichen. Der Path Error wird erst ausgelöst, wenn eine gewisse Anzahl an Paketen nicht bestätigt wurde. Dieser Vorgang wurde bereits im Kapitel 6.4 beschrieben.

Nachdem der Schwellwert erreicht wurde, wird vom mp11 eine Path Error-Meldung generiert. Der Mesh Point mp10 empfängt diese Meldung und sendet sie weiter. Da der Mesh Point mp10 immer noch versucht Daten zu senden, löst er einen neuen Path Request aus. Die Path Request-Anfragen werden solange durchgeführt, bis ein neuer Pfad gefunden wurde oder die Datenübertragung abgebrochen wird.

Die automatische Pfadaktualisierung ist eine zweite Variante um festzustellen, ob ein Pfad noch existiert. Sobald der Pfad in der Forwarding Table nicht mehr gültig ist, generiert der Mesh Point mp10

einen neuen Path Request mit dem Target Only Flag gesetzt. Diese Anfrage wird an den mp11 weitergeleitet. Da von diesem kein Path Reply zurückkommt, versucht der mp10 den mp12 mit einem Path Request ohne Target Only zu erreichen. Der Mesh Point mp11 meint, noch einen gültigen Pfad zum mp12 zu besitzen und sendet nun selber den Path Reply zum mp10 zurück. Gleichzeitig generiert der mp11 einen Path Request Broadcast mit dem Target Only Flag gesetzt. Der Mesh Point will überprüfen, ob der Pfad zum mp12 noch stimmt. Dieser ist aber auch nicht erfolgreich. Danach versucht der mp10 immer wieder über einen Path Request eine Verbindung zum mp12 aufzubauen. Dieser Versuch geht solange, bis der Verbindungsversuch von der Applikation abgebrochen wird.

In dieser zweiten Variante gibt es keinen Path Error, durch fehlgeschlagene Path Requests können die Mesh Points aber feststellen, dass ein Mesh Point nicht mehr erreichbar ist.

7.4.4 Szenario Übertragungsknoten verlässt das Netzwerk

In diesem Szenario sendet der Mesh Point mp10 einen Ping an den Mesh Point mp14. In der Abbildung 36 ist ersichtlich, dass fünf Mesh Points in einem Netzwerk miteinander verbunden sind. Jedoch sehen sich nicht alle Mesh Points direkt. Will der Mesh Point mp10 Daten an den Mesh Point mp14 senden, gibt es zwei Wege. Entweder die Daten gehen über den Mesh Point mp12 oder den Mesh Point mp13. Während dem Senden von Daten soll automatisch der Pfad gewechselt werden, wenn ein Mesh Point ausfällt.

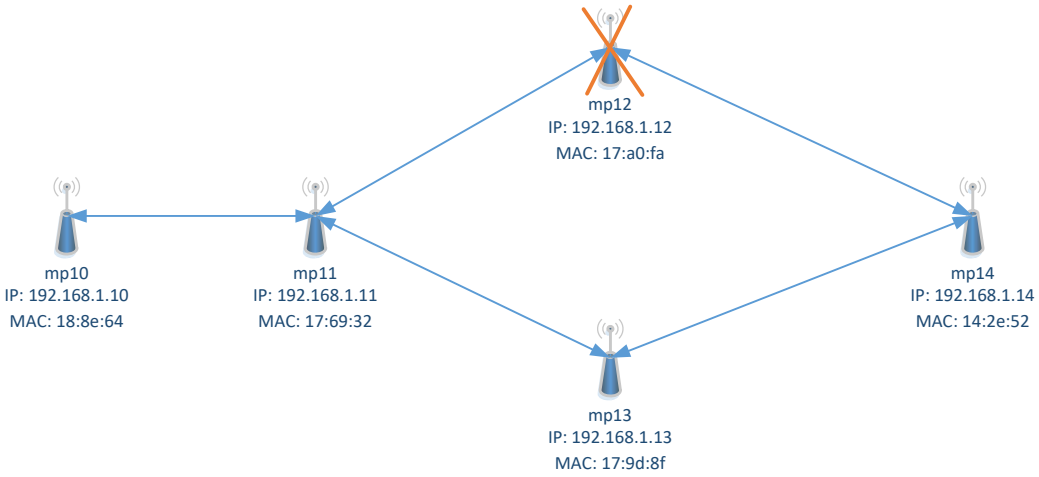


Abbildung 36: Topologie Szenario Übertragungsknoten verlässt das Netzwerk

7.4.4.1 Konfiguration

Damit diese Topologie richtig aufgebaut wird, werden einzelne Verbindungen zwischen Mesh Points blockiert. Dazu dient folgende Konfiguration:

Konfiguration mp10:

```
iw dev mp10 station set f8:1a:67:17:a0:fa plink_action block

iw dev mp10 station set f8:1a:67:17:9d:8f plink_action block

iw dev mp10 station set 98:de:d0:14:2e:52 plink_action block
```

Konfiguration mp11:

```
iw dev mp11 station set 98:de:d0:14:2e:52 plink_action block
```

Konfiguration mp12:

```
iw dev mp12 station set f8:1a:67:17:9d:8f plink_action block

iw dev mp12 station set f8:1a:67:18:8e:64 plink_action block
```

Konfiguration mp13:

```
iw dev mp13 station set f8:1a:67:17:a0:fa plink_action block

iw dev mp13 station set f8:1a:67:18:8e:64 plink_action block
```

Konfiguration mp14:

```
iw dev mp14 station set f8:1a:67:18:8e:64 plink_action block

iw dev mp14 station set f8:1a:67:17:69:32 plink_action block
```

7.4.4.2 Erwartung

Sobald der Mesh Point mp12 ausfällt, muss ein neuer Pfad gefunden werden. Dafür sendet der Mesh Point mp11 einen Path Error an seine Nachbarknoten. Der Mesh Point mp10 empfängt diese Meldung und erstellt einen neuen Path Request. Dieser Path Request wird von allen Mesh Points weitergeleitet, bis der Ziel-Mesh Point gefunden wurde. Der Ziel-Mesh Point sendet wie gewohnt einen Path Reply zurück. Danach ist der Pfad zum mp14 wieder funktionsfähig und Daten können gesendet werden. Der Ausfall eines Übermittlungsknotens sollte darum die Zeit nur kurz verzögern.

7.4.4.3 Verifikation

Auch dieses Verhalten konnte mit Wireshark nur bedingt nachvollzogen werden. Wie schon beim Szenario «Zielknoten verlässt das Netzwerk» sendet der mp11 erst nach unbestimmter Zeit einen Path Error. Verliert der Pfad in dieser Zeit seine Gültigkeit, löst der Mesh Point mp12 automatisch einen neuen Path Request mit dem Target Only Flag aus. Mit diesem Path Request wird gerade die alternative Verbindung gefunden ohne je einen Path Error erhalten zu haben. Nach dem erfolgreichen Erhalt des Path Replys können die Daten wieder erfolgreich übermittelt werden.

7.4.4.4 Fazit

Die Path Error Meldung wird bei einem Ausfall eines Links nicht unmittelbar danach generiert. Der Mesh Point erkennt zu langsam, ob eine Verbindung noch aktiv ist oder nicht. Durch das ständige Aktualisieren rückt der Sinn der Path Error-Meldung in den Hintergrund. Ist auf dem Mesh Point ein kleiner HWMP Lifetime-Wert eingestellt, übernimmt die automatische Pfadaktualisierung den Path Failover. Eine Verbindung wird somit höchstens so lange unterbrochen, wie der HWMP Lifetime-Wert eingestellt ist. Danach weiss das Netzwerk den alternativen Pfad und sendet die Daten über diesen.

Dieses Verhalten wurde jedoch vor allem beim Ping festgestellt. Bei anderen Applikationen die mehr Netzwerkverkehr generierten, wurde der Path Error viel schneller ausgelöst.

7.5 Energieeffizienz

In diesem Kapitel wird beschrieben, wieviel Energie der Raspberry Pi verbraucht. Dies ist wichtig für die Dimensionierung des Akkus.

7.5.1 Verbrauchsangaben

Der Raspberry Pi 2 Model B braucht gemäss dieser Webseite[3] im schlafenden Zustand etwa 200 mA und bei Volllast 800 mA. Durch einen angeschlossenen USB WLAN-Adapter braucht es zusätzliche Energie. Wie in dieser Quelle[4] beschrieben braucht der Raspberry Pi im normalen Zustand etwa 650 mA.

7.5.2 Akku Test

Mit einem Akku Test soll nun herausgefunden werden, wie lange ein Raspberry Pi betrieben werden kann. Geht man davon aus, dass eine Gondelbahn durchschnittlich acht Stunden am Tag betrieben wird, so sollte der Akku mindestens für diese Zeit durchhalten.

Für den Test wurde an einen Raspberry Pi ein Akku mit einer Leistung von 2800 mAh angeschlossen. Der Raspberry Pi wurde über diese Zeit hinweg unter relativ kleiner Last betrieben. Es wurden Netzwerkverkehr generiert und Topologie-Änderungen durchgeführt.

Die Messung ergab, dass mit diesem Akku der Raspberry Pi dreieinhalb Stunden lang betrieben werden kann. Hochgerechnet auf acht Stunden werden somit mindestens 6‘400 mAh benötigt. Da der Raspberry Pi jedoch nicht unter Volllast getestet wurde, ist eine Akkukapazität von über 8‘000 mAh sicherlich besser geeignet.

7.5.3 Schlussfolgerung

Im Kapitel Verbrauchsangaben wurde angenommen, dass der Raspberry Pi etwa 650 mA verbraucht. Durch die folgende Berechnung und die Werte aus dem Kapitel Akku-Test kann nun die Annahme verifiziert werden.

$$\frac{2800mAh * \frac{3.7V}{5V}}{3.5} = 592\,mA$$

Der effektive Verbrauch war während dem Akku-Test etwa 600 mA, was die ursprüngliche Annahme von etwa 650 mA bestätigen würde. Somit kann ausgesagt werden, dass der Raspberry Pi im normalen Mesh-Betrieb etwa 600-650 mA verbraucht. Die Stromversorgung der Gondel sollte aber etwa für 800 mA ausgelegt sein, da zusätzliche Services die CPU-Last erhöhen könnten.

[3] https://www.raspberrypi.org/help/faqs/#powerReqs

[4] https://learn.adafruit.com/introducing-the-raspberry-pi-2-model-b?view=all

7.6 Services

In den vorherigen Kapiteln wurde beschrieben, wie das Mesh-Netzwerk gemäss Theorie reagieren sollte und wie die Reaktion in der Realität aussieht. Dies jedoch nur mit dem primitiven Ping-Kontrollmechanismus. Applikationen senden die Daten aber normalerweise mit unterschiedlichen Mechanismen aus. Wie das Netzwerk mit diesen Daten umgeht, wird im nächsten Abschnitt untersucht.

7.6.1 Gegensprechanalage

Damit im Notfall Passagiere in einer Gondel jemanden in der Station kontaktieren können, braucht es eine Gegensprechanlage. Da das ganze Netzwerk schon auf IP basiert, eignet sich das Kommunikationsprotokoll SIP [17] perfekt dafür.

7.6.1.1 Mechanismus

Über SIP wird zuerst die Verbindung zwischen den Teilnehmern aufgebaut. Danach findet die Kommunikation nur noch über eine RTP-Verbindung [18] statt. Im Fall des Testnetzwerkes hat jeder Mesh Point eine statische IP-Adresse. Die SIP-Verbindung kann somit zwischen den Mesh Points direkt aufgebaut werden. Die anschliessende RTP-Verbindung ist dabei immer Peer-to-Peer.

7.6.1.2 Paketanalyse

Sobald die Verbindung per SIP aufgebaut ist, tauschen die beiden Teilnehmer des Gesprächs nur noch RTP Unicast-Nachrichten miteinander aus. Diese Pakete werden per UDP versendet und haben etwa einen Payload von 90 – 110 Bytes. Da als Empfängeradresse eine Unicast MAC-Adresse eingetragen ist, müssen alle Pakete mit dem WLAN Acknowledgement bestätigt werden. Kann ein Empfänger nicht erreicht werden, versucht der Sender den Empfänger mit einer niedrigen Datenrate zu erreichen.

```
192.168.1.1       192.168.1.14                              UDP     7078→7078 Len=97
                  BelkinIn_90:b2:f8 (94:10:3e:90:b2:f8) (RA)  802.11  Acknowledgement, Flags=........C
```

Abbildung 37: Wireshark Trace UDP Paket mit Acknowledgement

7.6.1.3 Praktischer Test

Für die praktischen Tests wurde linphone als VoIP Client verwendet. Über die Kommandozeilen-basierte Schnittstelle können sehr einfach Telefongespräche aufgebaut werden.

Durch die Feststellung, dass alle Pakete mit einem WLAN Acknowledgement bestätigt werden müssen und sehr viele Pakete durchs Netz gesendet werden, müsste ein Path Error sehr schnell ausgelöst werden. Mit einem Test wurde dies versucht zu bestätigen.

Im Kapitel 7.4.4 wurde beschrieben, was passiert, wenn während eines Pings ein Übertragungsknoten das Netzwerk verlässt. Der gleiche Test wurde nun auch noch durchgeführt mit einer aktiven Telefonverbindung. Die Mesh Points mp10 und mp14 sind dabei aktiv miteinander verbunden. Sobald nun der Mesh Point mp12 ausfällt, erhalten die beiden Mesh Points mp11 und mp14 keine WLAN Acknowledgements mehr. Ein Path Error wird ausgelöst und es wird sofort nach einem neuen Pfad gesucht. Der Failover auf den zweiten Pfad dauerte weniger als 0.3 Sekunden, und das Gespräch zwischen den beiden Stationen kann normal weiter geführt werden.

7.6.1.4 Fazit

Durch die kleinen aber vielen UDP-Pakete wird ein Path Error sehr schnell ausgelöst und ein alternativer Pfad dadurch in kurzer Zeit gefunden. Mit SIP könnte somit eine solche Gegensprechanalage realisiert werden.

7.6.2 Multicast

Die Idee ist, die Lautsprecherdurchsagen über Multicast zu verteilen. Dadurch müsste die Quelle die Pakete mit den Sprachdurchsagen nur einmal aussenden. Alle Mesh Points in den Gondeln könnten diese Pakete empfangen und verarbeiten.

7.6.2.1 Mechanismus

Der Mesh Point in der Station sendet die Audiosignale aus und jeder Mesh Point sendet diese weiter. Die Daten werden dabei über UDP versendet. Das bedeutet, dass der Sender nicht kontrolliert, ob der Empfänger Pakete erhalten hat. Der Mesh Point in der Station weiss somit nicht, ob alle Mesh Points im Netzwerk die Pakete erhalten haben. Empfängt ein Mesh Point ein Multicast-Signal, wird es automatisch weitergeleitet; vorausgesetzt ist ein aktives Peering zwischen den beiden Knoten.

Erhält ein Mesh Point ein Paket, das er schon weitergeleitet hat, wird es nicht mehr weitergeleitet. Somit wird sichergestellt, dass Multicast-Pakete im Netzwerk nicht dupliziert werden. Dafür ist in jedem versendeten Paket im Mesh Control Field eine Sequenznummer hinterlegt. [3, p. 971]

7.6.2.2 Paketanalyse

Zusätzlich zur verbindungslosen UDP-Verbindung werden die Pakete vom Empfänger auch nicht mit dem WLAN Acknowledgement bestätigt. Gerade diese WLAN Acknowledgements wären aber nötig um feststellen zu können, ob die Pakete erfolgreich weitergeleitet wurden. Der Mesh Point empfängt die Pakete nur und sendet sie an Mesh Points in seiner Umgebung weiter.

Im Kapitel 6.4 wurde beschrieben, wie die Pakete bei Unicast mit einem Acknowledgement bestätigt werden. Als Beispiel wird in der Abbildung 38 erklärt, dass diese fehlenden Acknowledgements nun bei Multicast zu einem Problem werden. Jeder Mesh Point sieht nur gerade seine beiden Nachbarn und hat mit ihnen ein Peering aufgebaut. Der Mesh Point mp10 sendet dabei einen Multicast-Strom ins Netzwerk. Da der Erhalt der Pakete nun nicht mit einem Acknowledgement bestätigt wird, kann es sein, dass Multicast-Pakete im Netzwerk verloren gehen. In der Grafik ist zu erkennen, dass der Mesh Point mp15 nur gerade die Multicast Pakete 1 und 4 erhält. Alle anderen Multicast-Pakete gehen auf dem Weg verloren.

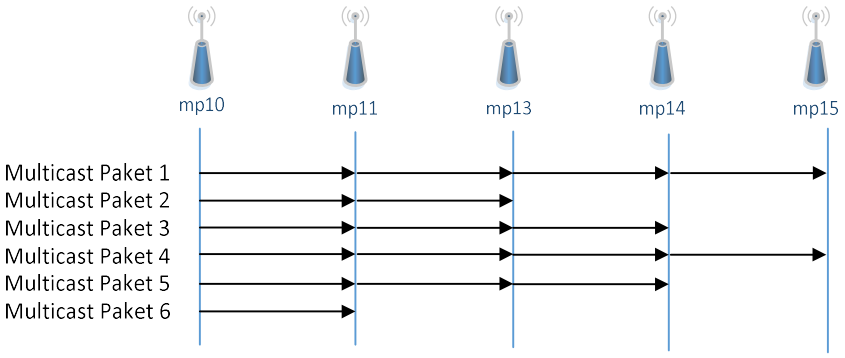


Abbildung 38: Verteilung der Multicast Pakete im Mesh-Netzwerk

Als Receiver- und Destination-Adresse ist dabei die MAC-Adresse 01:00:5e:01:01:21 eingetragen. Die ersten 23 Bit der MAC-Adresse sind immer gleich und zeigen dem Mesh Point auf, dass gerade Multicast-Pakete über Layer2 übertragen werden. Dadurch weiss der Mesh Point, dass er diese Pakete an alle seine Nachbarn weiterleiten sollte. Im Gegensatz dazu sind die letzten 25 Bit der MAC-Adresse generiert. Diese sind aber nicht zufällig, sondern werden von der IP-Adresse abgeleitet. Dies ist nötig,

damit auf Layer 2 die Mesh Points den Datenverkehr entsprechend filtern können. Hört eine Anwendung auf einem Mesh-Point auf eine bestimmte IP-Adresse, kann anhand der Destination MAC-Adresse herausgefunden werden, ob das Paket an den Network Layer weitergereicht werden soll.

7.6.2.3 Praktischer Test

Im praktischen Test wurde eine Audiodatei per Multicast an die Mesh Points verteilt. Als Sender- und Empfängerprogramm wurde dabei VLC verwendet. Der Test wurde zuerst in einem Umfeld ausgeführt mit vielen Störungseinflüssen. Die Durchsagen waren aber nicht verständlich, da sie ständig unterbrochen wurden. Da die gesendeten Pakete nicht mit einem WLAN Acknowledgement bestätigt werden, können Pakete unbemerkt verloren gehen. Wenn der Empfänger nun die Nachricht aus den Paketen wiederherstellen will, fehlen einzelne Pakete. Diese fehlenden Blöcke werden mit Nullen aufgefüllt und es ist Stocken in der Durchsage wahrnehmbar.

Der Grund für den Verlust der Pakete kann verschiedene Gründe haben. Dass Störungen auf dem WLAN-Kanal einen grossen Einfluss haben können, wurde in einem zweiten Test bewiesen. Dieser erfolgte an einem Ort mit sehr wenigen Störungen. Die Durchsage war deutlich besser erkennbar, aber auch bei diesem Test gab es hin und wieder Unterbrüche.

Eine Messung hat gezeigt, wie hoch etwa die Paketverluste sind. Das Mesh Point Portal mpp1 sendete dabei über 20 Sekunden einen Multicast-Strom aus. Der Mesh Point mp10 empfängt diese Multicast-Pakete und sendet diese an alle seine Mesh Points in der Umgebung weiter. Der Mesh Point mp11 erhält dadurch diese Pakete und sendet sie wieder weiter. Dies bildet die Situation ab, wenn mehrere Gondeln an einem Seil hängen, untereinander aber nur den direkten Nachbarn sehen.

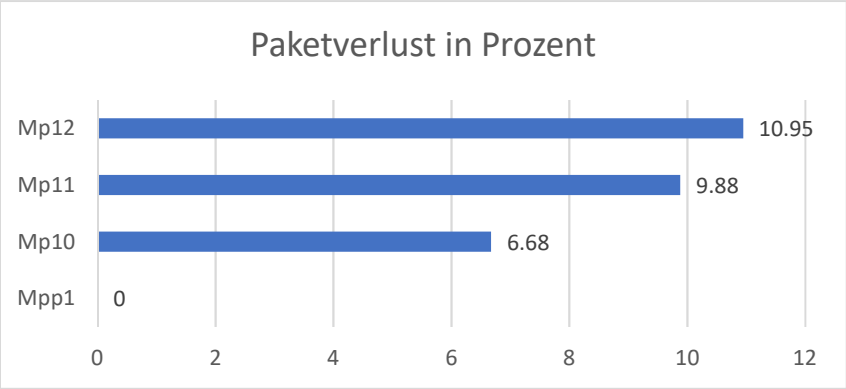


Abbildung 39: Paketverlust über mehrere Hops

Die Multicast-Datenrate war auf 6 Mbit/s eingestellt. Wenn sich zwei Mesh Points miteinander verbinden, versuchen sie über die höchstmögliche Datenrate Daten auszutauschen. Sobald eine Station jedoch keine Acknowledgements mehr versendet, wird die Datenrate reduziert. Dies funktioniert jedoch bei Multicast nicht. Darum wird eine Standard-Multicast-Datenrate verwendet. Alle Mesh Points, die diese Datenrate noch empfangen können, werden die Pakete auch weiterleiten.

7.6.2.4 Fazit

Gerade im Notfall ist es wichtig, dass Sprachdurchsagen vollständig und ohne Fehler beim Empfänger ankommen. Multicast über das Mesh-Netzwerk wird unterstützt und wäre in der Theorie möglich. Jedoch ist die Verbindung nicht gesichert und externe Störeinflüsse können die Übertragung empfindlich stören. In den Tests wurden jeweils Audiosignale per MP3 versendet. Ein Codec mit einer besseren Fehlerkorrektur könnte den Paketverlust besser überspielen. Gehen aber über längere Zeit mehrere Pakete verloren, kommt es auch mit einem besseren Codec zu Unterbrechungen im Audio Stream.

7.6.3 Audioübermittlung

Da sich Multicast im Falle des Gondelbahnproblems nur bedingt eignet, um Audiodaten zu übermitteln, gibt es nur die Alternative, Daten per Broadcast oder Unicast an die einzelnen Mesh Points zu übermitteln. Broadcast hat hierbei dasselbe Problem im WLAN wie Multicast - dass Pakete vom Empfänger nicht mit einem Acknowledgement bestätigt werden. Die Daten müssen daher besser per Unicast übermittelt werden. Um dies zu realisieren, eignet sich das Realtime Streaming Protokoll RTSP [19].

7.6.3.1 Mechanismus

RTSP ist, ähnlich wie SIP, für den Aufbau beziehungsweise die Kontrolle einer Verbindung zuständig. Die Clients bauen mittels RTSP eine Verbindung zu einem Quellserver auf, welcher einen Audiostream anbietet. Sobald die Verbindung zwischen Client und Server aufgebaut ist, werden Daten mittels UDP ausgetauscht.

7.6.3.2 Paketanalyse

Die Paketanalyse zeigt auf, dass wie bereits in Kapitel 7.6.1 beschrieben, die Unicast UDP Pakete mit einem Acknowledgement bestätigt werden. Die Payload variiert dabei zwischen 740 und 850 Bytes.

192.168.1.1	192.168.1.14	UDP	746 54
	BelkinIn_30:c3:a7 (14:91:82:30:c3:a7) (RA)	802.11	40 54
192.168.1.1	192.168.1.16	UDP	746 54
	BelkinIn_30:c3:a7 (14:91:82:30:c3:a7) (RA)	802.11	40 54
192.168.1.1	192.168.1.16	UDP	746 54
	BelkinIn_c8:0a:a3 (c0:56:27:c8:0a:a3) (RA)	802.11	40 54

Abbildung 40: Wireshark Trace RTSP UDP Stream

7.6.3.3 Praktischer Test

Im praktischen Test wurde auf einem Mesh Point Portal mit VLC ein RTSP Stream erstellt, welcher von zwei weiteren Mesh Points empfangen wurde. Ein Mesh Point diente dabei als Zwischen-Hop zum anderen. Dieses Szenario ist auch im obigen Wireshark Trace ersichtlich, wo die Quelle die beiden Mesh Points bedient.

Der Audiostream konnte dabei von beiden Mesh Points empfangen und abgespielt werden. Bemerkbar machten sich jedoch äusserliche Störeinflüsse. Während es in einer «ruhigeren» Umgebung mit wenig externen Störungen keinerlei Probleme bei der Übertragung gab, so sind bei einer stark gestörten Umgebung viele Paketverluste zu vermelden. Eine stark gestörte Umgebung ist hierbei zum Beispiel die Hochschule, bei der mehrere WLAN Signale auf demselben Kanal arbeiten. Diese Störungen führten zu einem enormen Stocken beziehungsweise Unverständnis des Audiosignals.

Ein weiterer Test hat gezeigt wie viele Pakete nochmals gesendet werden müssen, bis sie beim Empfänger ankommen. Der Test wurde in einer Umgebung mit mässigen Störungseinflüssen durchgeführt. Die Datenrate musste von 26 Pakten auf 48 Mbit/s reduziert werden, davon wurden 15 Pakete vom Empfänger bestätigt. Die restlichen 11 Pakete versuchten es mit einer Datenrate von 36 Mbit/s. Bei einem Paket muss die Datenrate bis auf 18 Mbit/s reduziert werden, bis es vom Empfänger bestätigt werden konnte.

Versuch	Gesendete Pakete	Datenrate
1	758	54
2	26	48
3	11	36
4	3	24
5	1	18

Tabelle 24: Messergebnis WLAN Retries

7.6.3.4 Fazit

Mittels RTSP können unkompliziert Audiodaten an die verschiedenen Mesh Points übermittelt werden. Durch die Verwendung von UDP ist jedoch keine garantierte Übermittlung gewährleistet, was im Falle von Paketverlusten auch Datenverlust bedeutet.

Als Alternative könnte man auf einen HTTP Stream ausweichen, der die Audiodaten mittels TCP vom Server zum Client übermittelt. Somit wäre gewährleistet, dass jeder Mesh Point die Audiosignale auch korrekt empfängt, jedoch würde die Bandbreite darunter leiden. Der Vorteil von TCP liegt aber darin, dass der Stream sicher vollständig übertragen wird.

8 Implementierung

In diesem Kapitel wird beschrieben, wie das Mesh-Netzwerk anhand einer reellen Gondelbahn umgesetzt wird. Mit den Erkenntnissen aus den vorhergehenden Kapiteln wurde zudem ein Skript zur Verbesserung des Mesh-Netzwerkes geschrieben.

8.1 Reale Testumgebung

Um feststellen zu können, ob die erarbeitete Lösung auch tatsächlich in einem realen System funktioniert, braucht es eine Testumgebung, die möglichst nahe an die Realität herankommt. Bevor eine ganze Gondelbahn mit der vorgeschlagenen Lösung ausgetestet wird, muss die Lösung in einem kleineren Modell intensiv getestet werden.

8.1.1 WLAN-Signalstärke

Im Kapitel Distanzmessung wurde herausgefunden, dass das WLAN-Signal über 100 Meter strahlen kann. Dies ist ideal für eine reelle Gondelbahn, aber in einem Modell bringt diese grosse Distanz Problem mit sich. Um die Ausbreitung der WLAN-Signale zu minimieren, braucht es eine Lösung, wie die Signale gedämpft werden können. Eine Variante wäre, die externen WLAN-Antennen mit Dämpfungsgliedern abzuschwächen. Da aber nicht alle WLAN Adapter externe Antennen haben, braucht es eine Antennen-unabhängige Lösung, um die Ausbreitung der Signale einschränken zu können. Eine weitere Möglichkeit wäre, das Signal auf Systemebene abzuschwächen. Dies funktioniert jedoch nicht bei jedem WLAN Adapter. Unter Linux kann mit dem Befehl «iw $MESH_IFACE set txpower fixed 100» die Sendeleistung auf 1 dBm reduziert werden. Standardmässig wird mit 20 dBm gesendet (volle Leistung). In unmittelbarer Nähe ist die Reduzierung nicht sonderlich wahrnehmbar, ein Mesh Point empfängt das ausgesendet Signal nur um etwa 10 dBm weniger. Diese Abschwächung ist aber im kleinen Modell noch zu wenig, weshalb eine andere Lösung gesucht werden musste.

Da WLAN-Strahlen empfindlich gegenüber Aluminium sind, wird eine bessere Abschwächung der Signale durch das Umschliessen der Antenne in einer Blechdose erreicht.

8.1.2 Testumgebungsvarianten

Für die Systemtests wurden diverse Testumgebungsvarianten ausprobiert. Physikalische Einflüsse spielen bei der Testumgebung eine erhebliche Rolle. Aus diesem Grund wurde eine Variante in der Luft und eine auf dem Boden ausgearbeitet. Mit beiden Varianten wurden Versuche durchgeführt.

8.1.2.1 Lego Eisenbahn

Mit einer Lego Eisenbahn kann die Dynamik im Netzwerk gut getestet werden. Die Raspberry Pis werden dabei in die Blechdosen gelegt. Die Blechdosen simulieren sozusagen die Gondeln und werden auf Wagen auf der ganzen Strecke verteilt. Die Wagen werden mit einem Faden untereinander verbunden. Bewegt sich nun der vorderste Wagen, zieht er alle anderen Wagen mit. Diese Variante hat aber gewisse Nachteile in der Grössenlimitierung. Sowohl der Abstand zwischen den Gondeln wie auch die Menge an Mesh Points sind beschränkt.

8.1.2.2 Seilbahn

Als Variante in der Luft wurde eine Seilbahn evaluiert. Diese Variante hat im ersten Moment am meisten Ähnlichkeit zur Realität. Schwierigkeiten ergeben sich hier aber primär aufgrund der Schwerkraft. Je mehr Gondeln am Seil hängen, desto grösser werden die Kräfte auf das Seil und die Umlaufstationen. Dafür sind grössere Distanzen zwischen den Gondeln möglich.

8.1.3 Logische Topologie

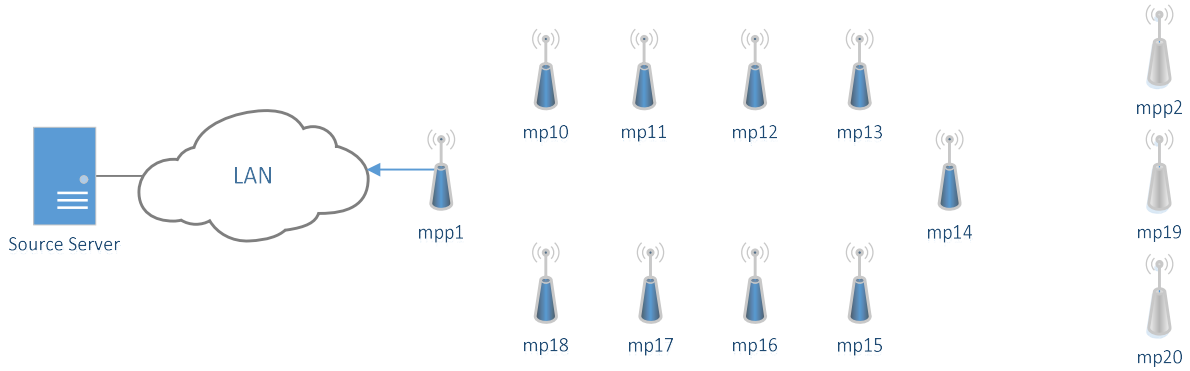


Abbildung 41: Testumgebung Topologie-Übersicht

Die Topologie wird so abgebildet, dass sie einer reellen Gondelbahn nahekommt. Auf der linken Seite befindet sich die Talstation mit dem Zugang ins LAN, wo zum Beispiel ein Streaming Server stehen kann. In der Mitte befindet sich das Mesh-Netzwerk, welches durch das Mesh Point Portal mit dem LAN verbunden ist. Die weissen Geräte in der Topologie-Übersicht sind nicht aktive Mesh Points, welche lediglich fürs Testing zusätzlich eingebaut werden können.

8.1.4 Verwendete Hardware

Anzahl	Hardware	Beschreibung
4	Raspberry Pi 2 Model B	ARM Cortex-A7 (4x 900MHz) 1 GB RAM 10/100 Mbit Ethernet
8	Raspberry Pi 3	ARM Cortex-A53 (4x 1200MHz) 1 GB RAM 10/100 Mbit Ethernet
12	Xtorm FS101 (Akku)	5000 mAh
12	Linksys AE3000	USB WLAN Adapter mit Chipsatz RT3573 (Unterstützung für Mesh Netzwerke).

Tabelle 25: Verwendete Hardware in reeller Testumgebung

8.2 Verbesserungsskript

Im Kapitel 7.4.1 Szenario Kreuzen in voller Fahrt wurde das Problem beschrieben, dass sich kreuzende Gondeln ebenfalls miteinander verbinden können und somit über diesen Pfad ebenfalls ein Datenfluss möglich ist. Dies ist jedoch nicht nötig für das gesamte System, und darum werden mit Hilfe eines Skripts die Verbindungen zwischen den Mesh Points kontrolliert.

8.2.1 Grundidee

Ein Mesh Point verbindet sich nur mit einem anderen Mesh Point, wenn er sein WLAN Beacon-Signal eine gewisse Zeit empfangen hat. Beim Starten sind somit zuerst alle Verbindungen blockiert. Empfangen beide Mesh Points über einen längeren Zeitraum das Signal des anderen, wird die Blockierung aufgehoben, und die Mesh Points können ein Peering miteinander aufbauen. Sobald aber diese Mesh Points kein WLAN-Signal mehr gegenseitig austauschen, wird die Verbindung wieder blockiert. Diese Situation wird eintreten, wenn sich die Gondelbahn nach einem längeren Unterbruch wieder in Bewegung setzt.

8.2.2 Algorithmus

Jede Sekunde liest das Skript die aktuelle Peering Liste aus. In der Ausgabe kann anhand der «inactive time» festgestellt werden, von welchem Mesh Point das letzte Mal ein aktives Signal empfangen wurde. Da die Mesh Points jede Sekunde ein Beacon aussenden, sollte die «inactive time» nicht grösser als eine Sekunde sein, wenn sich ein anderer Mesh Point in der Umgebung befindet. Jede Sekunde wird nun die «inactive time» abgefragt und damit ein Zähler erhöht, wenn der Mesh Point noch sichtbar ist. Erreicht der Zählerstand einen Wert höher als zehn, wird die Blockierung aufgehoben und der Mesh Point kann sich mit der Gegenstation verbinden. Kommt nun ein Mesh Point ausser Reichweite, reduziert sich dieser Zähler wieder. Ist dieser Wert nun kleiner als zehn, wird die Verbindung wieder blockiert. Es gibt sowohl einen minimalen wie auch einen maximalen Zählerwert. Damit wird sichergestellt, dass die Zähler nicht zu gross werden und ein Wechsel zwischen Blockieren und Öffnen der Verbindung nicht zu lange dauert. Gerade wenn die Gondelbahn über längere Zeit steht, würde der Zähler zu gross werden und ein Reduzieren würde sehr lange dauern.

Je nach Geschwindigkeit der Bahn und Abstand zwischen den Gondeln müssen diese Werte bei einer konkreten Implementation nochmals individuell angepasst werden.

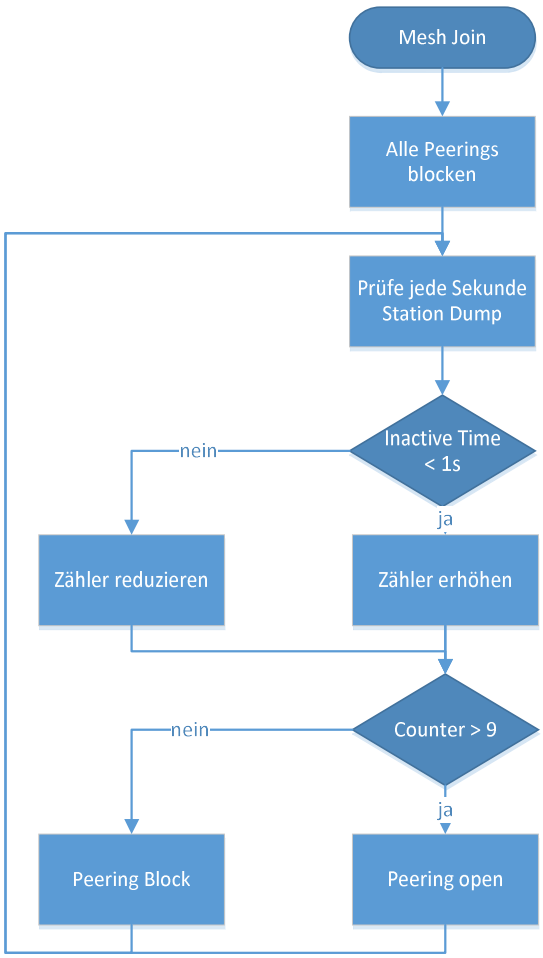


Abbildung 42: Blocking Algorithmus

8.2.3 Spezialfall Mesh Point Portal

Sobald die Gondel in die Station einfährt, soll sie sich mit dem Mesh Point Portal verbinden können. Der Verbindungskontrollmechanismus sollte darum das Mesh Point Portal nicht blockieren. Im Skript kann definiert werden, welches die Mesh Point Portals sind, damit diese standardmässig nicht blockiert sind.

8.2.4 Realisation

Das Skript ist in der Skriptsprache Bash realisiert. Diese kann auf allen Raspberry Pis direkt ausgeführt werden, und es braucht keine zusätzlich installierten Programme und Bibliotheken.

Um den aktuellen Zählerstand in Verbindung mit der MAC-Adresse abspeichern zu können, wird ein Associative Array verwendet. Die MAC-Adresse ist dabei der Schlüssel und der aktuelle Zählerstand der Wert. Nach jeder Abfrage wird somit dieser Wert erhöht oder reduziert. Als Standardwert ist 0 beim Zähler eingetragen.

8.3 Konfiguration

Die Raspberry Pis sind so konfiguriert, dass sie beim Starten automatisch mit dem Mesh-Netzwerk verbinden. Das Mesh-Netzwerk wird auf Kanal 1 aufgebaut. Damit niedrige Datenraten verhindert werden können und die Beacon Signale nicht zu weit strahlen, werden Bitraten unter 6 Mbit/s nicht verwendet. Nach dem Start wird automatisch das Verbesserungsskript geladen, um die Verbindungen zu blockieren.

8.4 Systemtest

In einem globalen Systemtest wurde versucht zu beweisen, dass alle Komponenten miteinander zusammenarbeiten, um ein stabiles Mesh-Netzwerk zu gewährleisten. In den bisherigen kleinen Einzeltests wurde nur untersucht, wie das Mesh-Netzwerk in gewissen Situationen reagiert, um die Schwachpunkte herauszufinden. In den folgenden Systemtests wurde das System als Ganzes betrachtet.

Da aber das Kreuzen der Gondeln nur bedingt einen Einfluss auf die hohe Dynamik im Netzwerk hat, wurde dies wiederum einzeln getestet. Dies ermöglichte auch eine bessere, qualifizierte Aussage bei den Tests mit hoher Dynamik. WLAN-Signale können bekanntlich nur bedingt beeinflusst werden und würden diesen Test nur unnötig stören.

8.4.1 Kreuzen

Das Kreuzen der Seilbahn wurde getestet um zu beweisen, dass nur mit Mesh Points eine Verbindung eingegangen wird, von denen konstant ein Signal empfangen wird. Für diese Funktion kam das in Kapitel 8.2 beschriebene Verbesserungsskript zum Einsatz.

8.4.1.1 Testaufbau

Da für diesen Test die Länge der Lego Eisenbahn nicht ausreichte, wurde die Seilbahn verwendet. Alle Raspberry Pis sind dabei in Blechdosen verpackt und am Seil aufgehängt.

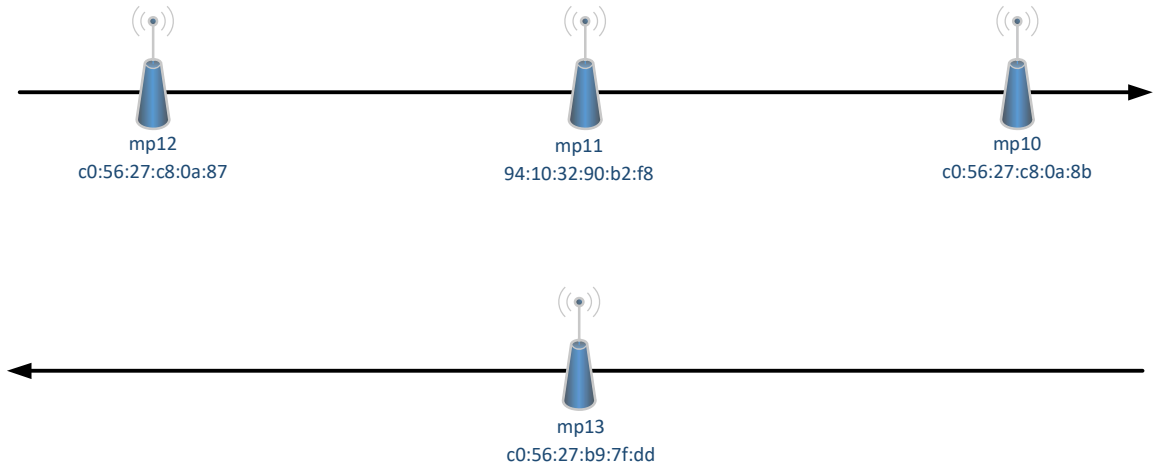


Abbildung 43: Kreuzenden Gondeln Seilbahn

8.4.1.2 Durchführung

Wie in der Abbildung 43 ersichtlich, kreuzt nun der Mesh Point mp13 nacheinander die Mesh Points mp10, mp11 und mp12. Die Geschwindigkeit ist dabei so hoch, dass der Mesh Point mp13 die kreuzenden Mesh Points nicht länger als neun Sekunden sehen sollte. Daten werden in diesem Test keine versendet, da der Fokus auf dem Blockierungsmechanismus liegt. Über eine blockierte Verbindung können logischerweise keine Daten ausgetauscht werden.

8.4.1.3 Resultat

Das Logfile auf dem Mesh Point mp13 beweist, dass die anderen Mesh Points nur kurz gesehen werden, aber keine Verbindung mit ihnen aufgebaut wird. Dieser Vorgang ist auf Abbildung 44 gut ersichtlich.

Der Mesh Point mp13 sieht den Mesh Point mp12 für eine gewisse Zeit. In dieser Zeit erhöht sich der Zählerwert, sichtbar in der Spalte Zählerwert. Fährt der Mesh Point mp12 wieder aus der Reichweite von mp10 (nach der roten Linie), erhöht sich die «inactive time» und der Zählerwert wird reduziert.

Sekunde	MAC-Adresse	Inactive Time	Zählerwert	Signalstärke in dB	Status
13	c0:56:27:c8:0a:87	153310	0	-77	BLOCKED
14	c0:56:27:c8:0a:87	70	1	-79	BLOCKED
15	c0:56:27:c8:0a:87	420	2	-77	BLOCKED
16	c0:56:27:c8:0a:87	770	3	-83	BLOCKED
17	c0:56:27:c8:0a:87	110	4	-73	BLOCKED
18	c0:56:27:c8:0a:87	460	5	-79	BLOCKED
19	c0:56:27:c8:0a:87	810	6	-77	BLOCKED
20	c0:56:27:c8:0a:87	100	7	-75	BLOCKED
21	c0:56:27:c8:0a:87	1480	6	-75	BLOCKED
22	c0:56:27:c8:0a:87	2860	5	-81	BLOCKED
23	c0:56:27:c8:0a:87	4230	4	-81	BLOCKED
24	c0:56:27:c8:0a:87	5600	3	-81	BLOCKED
25	c0:56:27:c8:0a:87	6940	2	-81	BLOCKED
26	c0:56:27:c8:0a:87	8330	1	-81	BLOCKED
27	c0:56:27:c8:0a:87	9710	0	-81	BLOCKED

Abbildung 44: Verbesserungsskript Blocking Log

8.4.1.4 Erkenntnis

Das entwickelte Verbesserungsskript funktioniert. Es wird nur mit Mesh Points eine Verbindung aufgebaut, die über längere Zeit gesehen werden.

8.4.2 Stationsdurchfahrt

Der Systemtest Stationsdurchfahrt soll aufzeigen, wie das Mesh-Netzwerk mit der Dynamik umgeht, falls ein Mesh Point in die Station einfährt, sich mit dem fixen Stationsknoten verbindet und die Station anschliessend wieder verlässt. Getestet wird unter anderem eine aktive VoIP-Verbindung, wie auch ein Audiostream auf TCP- und UDP-Basis. Bei TCP kommt hierbei ein HTTP Stream zum Einsatz und bei UDP ein RTSP Stream.

8.4.2.1 Testaufbau

Um den Test durchführen zu können, wurde auf die Lego Eisenbahn zurückgegriffen und mit den Schienen ein Oval aufgebaut.

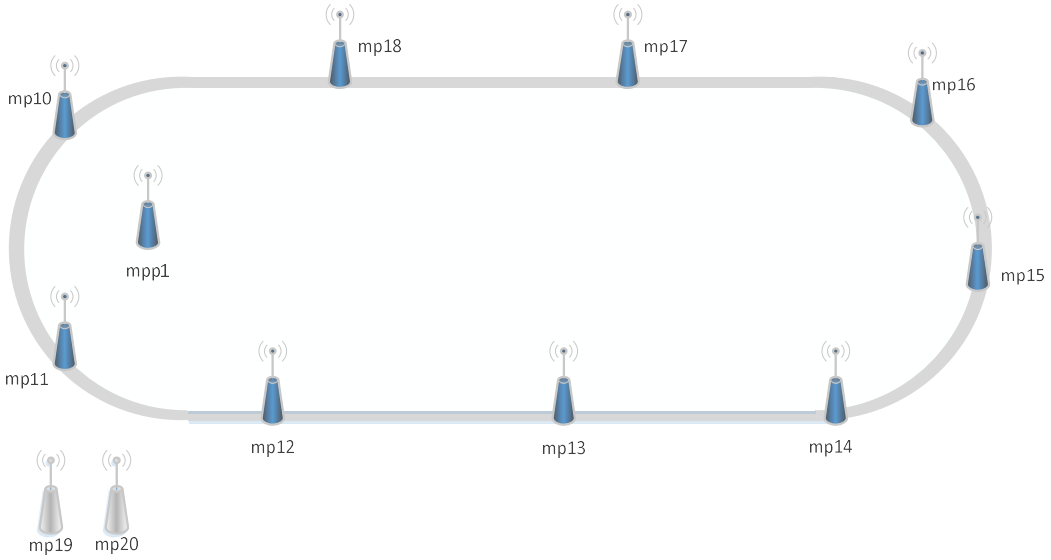


Abbildung 45: Testaufbau Stationsdurchfahrt Legoeisenbahn

Da in diesem verkleinerten System die dynamische Verbindung aufgrund der unberechenbaren WLAN-Signale nicht zuverlässig beeinflusst werden kann, wurden einzelne Verbindungen zwischen den Mesh Points blockiert. Jeder Mesh Point auf der Bahn (mp10 – mp18) darf nur jeweils mit seinen direkten Nachbarn kommunizieren. Alle anderen Verbindungen untereinander wurden manuell blockiert. Dies entspricht etwa den reellen Umständen, wie sie bei der Gondelbahn vorkommt.

mpp1 dient dabei als zentraler Zugangspunkt, mit dem sich jeder einzelne Mesh Point verbinden darf. Dieser wird durch eine Blechdose abgeschirmt, sodass er sich nicht direkt mit allen Mesh Points auf der Bahn verbinden kann.

mp19 und mp20 sind für die Konfigurationen zuständig und sind über LAN mit Laptops verbunden. mp19 kann dabei nur mit mpp1 direkt kommunizieren und mp20 nur mit mp11. Somit werden unnötige SSH-Verbindungen innerhalb des beweglichen Systems unterbunden.

8.4.2.2 Durchführung

Durchgeführt werden insgesamt drei Tests. Als erstes wird auf mpp1 ein RTSP Stream generiert, welcher mittels UDP eine Sprachnachricht an mp11 schickt. An mp11 ist ein Lautsprecher angeschlossen, um die Sprachnachricht auszugeben. Der Befehl, um den RTSP Stream zu starten, sieht folgendermassen aus:

```
alarm@mpp1: cvlc -vvv teststream.mp3 –sout
'#rtp{dst=192.168.1.1,port=1234,sdp=rtsp://192.168.1.1:8080/stream.sdp}'
```

Auf mp11 wird der Stream folgendermassen empfangen:

```
alarm@mp11: cvlc -vvv rtsp://192.168.1.1:8080/stream.sdp
```

Nachdem der Stream gestartet wurde und die Audioausgabe erfolgt, dreht die Lego Eisenbahn in mässigem Tempo eine Runde. Zeitgleich wird auf dem mpp1 jede Sekunde die Forwarding Table ausgelesen um analysieren zu können, über welche Hops mp11 den Stream über die Zeit hinweg empfangen hat.

Derselbe Testvorgang wird auch mit einem HTTP Stream wiederholt, um zu analysieren, wie sich eine TCP-Verbindung auf die Dynamik auswirkt. Für das Starten des HTTP Streams wurden folgende Befehle verwendet:

```
alarm@mpp1: cvlc -vvv input_stream --sout
'#standard{access=http,mux=ogg,dst=192.168.1.1:8080}'
```

Und zum empfangen:

```
alarm@mp11: cvlc -vvv http://192.168.1.1:8080
```

Zu guter Letzt wurde die VoIP-Verbindung getestet. Da durch die Abschirmung von mpp1 durch eine Blechdose kein Headset angeschlossen werden konnte, wurde die VoIP-Verbindung vom mp19 zu mp11 aufgebaut. Zu diesem Zweck wurde auf mp11 der Lautsprecher durch ein USB Headset ersetzt.

```
alarm@mp19: linphone
alarm@mp19: linphone > call sip:192.168.1.11
```

```
alarm@mp11: linphone
alarm@mp11: linphone > answer
```

8.4.2.3 Resultate

Die VoIP-Verbindung funktionierte über die gesamte Zeit hinweg relativ gut. Zwischendurch war das Sprachsignal teilweise kurz unterbrochen, jedoch konnte man nach kurzer Zeit wieder miteinander sprechen.

Der RTSP Audiostream funktionierte grösstenteils. Zwischendurch stockte der Stream und war nicht mehr gut verständlich.

Der HTTP Audiostream hatte am meisten Mühe. Zu Beginn war kein Problem auszumachen, jedoch brach der Stream nach ca. ¾ Runden ab.

8.4.2.4 Erkenntnisse

Die Auswertung der Forwarding Table ergab, dass während den Übertragungen überdurchschnittlich oft der Routing-Pfad gewechselt hat, und das, obwohl sich der Zug nicht wirklich bewegte. Dies zeigt zwar auf, dass das System mit der Dynamik umzugehen weiss, aber entspricht nicht ganz der Realität. Zurückzuführen ist dies auf die zu klein gewählte Testumgebung. Obwohl der mpp1 durch eine Blechdose abgeschirmt wurde, kamen teils Verbindungen zu weit entfernten Mesh Points zu Stande, da die Metrik zu diesen offenbar für kurze Zeit besser war, als zu den Mesh Points in seiner unmittelbaren Umgebung. Die UDP-Verbindungen von VoIP und RTSP kamen mit der Situation der ständig wechselnden Pfade besser zu Recht als eine TCP-Verbindung. Dies weist darauf hin, dass TCP in einem hoch dynamischen Netzwerk gewisse Probleme verursacht. Dieses Phänomen machte sich erst in den Implementationstest bemerkbar. Das sich TCP und WLAN nicht optimal ergänzen wird bereits in dieser Literatur [15, pp. 411-413] beschrieben.

Um die Dynamik der Stationsdurchfahrt in einem womöglich besseren Umfeld zu testen, wird der Systemtest in Kapitel 8.4.3 in abgeänderter Form nochmals durchgeführt.

8.4.3 Erweiterte Stationsdurchfahrt

Da Aufgrund der vielen unnötigen Pfadwechsel in der Eisenbahn-Testumgebung keine aussagekräftigen Tests und Analysen gemacht werden konnten, musste auf eine alternative Möglichkeit ausgewichen werden. Bei diesem Systemtest werden nun dieselben Tests noch einmal durchgeführt, jedoch bewegt sich nicht immer das gesamte System. Es befindet sich lediglich jeweils ein Mesh Point in Bewegung, der die Einfahrt bzw. Ausfahrt der Station simuliert.

8.4.3.1 Testaufbau

Für diesen Testaufbau wurde dieselbe Konfiguration wie auch bei der Lego Eisenbahn verwendet. Jeder Mesh Point kann nur jeweils mit seinen Nachbarn sowie dem Mesh Point-Portal kommunizieren.

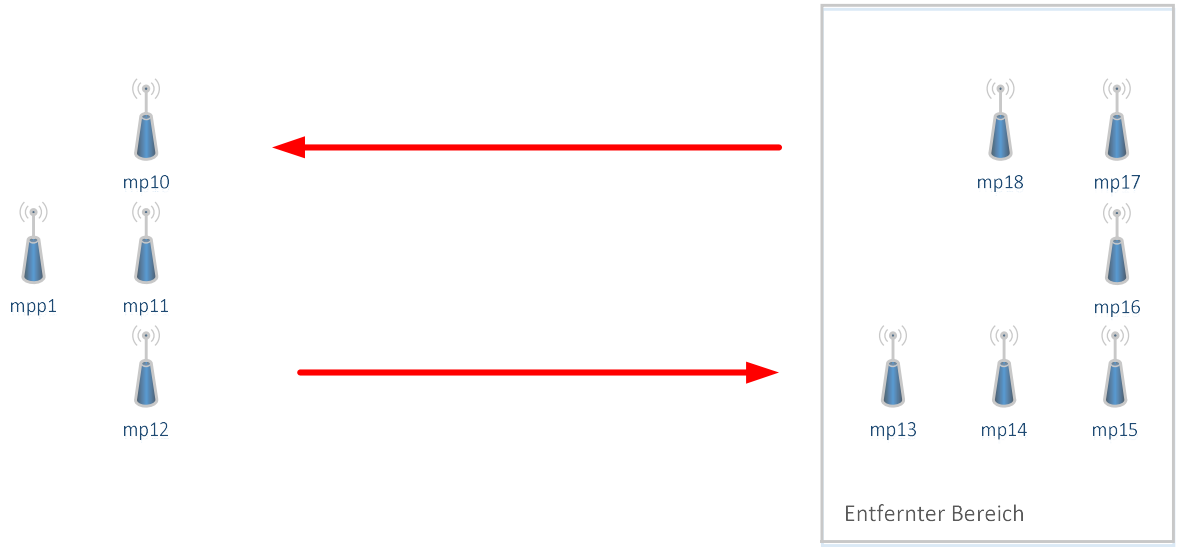


Abbildung 46: Testaufbau Stationsdurchfahrt Einfach

Auf der einen Seite befindet sich in einer Blechdose das Mesh Point Portal, um die WLAN-Signale zu dämpfen. In unmittelbarer Nähe befinden sich drei weitere Mesh Points. Diese stellen die Situation dar, dass mehrere Gondeln sich in der Durchfahrt befinden können. Auf der anderen Seite befinden sich in einem abgesonderten Bereich die restlichen Mesh Points. Durch die Entfernung und die Abschirmung des Mesh Point Portals kann keine direkte Verbindung zu diesem entstehen.

8.4.3.2 Durchführung

Es werden dieselben Tests wie bereits bei der Lego Eisenbahn durchgeführt. Auf dem Mesh Point-Portal wird jeweils ein Stream gestartet und auf mp11 ausgegeben. Danach wird jeweils von der einen Seite ein Mesh Point entfernt und in den abgesonderten Bereich getragen. Aus diesem Bereich wird wiederum ein Mesh Point mitgenommen und in die «Station» getragen. Dies geschieht solange, bis alle Mesh Points wieder an ihrem ursprünglichen Ort sind.

8.4.3.3 Resultate

Zu Beginn ist mp11 direkt mit mpp1 verbunden und kann so den Stream direkt empfangen. Die Ausfahrt von mp12 ändert daran nichts. Auch die anschliessende Einfahrt von mp13 ändert an der Situation nichts. Wird nun mp11 jedoch aus der Station entfernt, so verliert er nach einer gewissen Zeit die Verbindung zum mpp1 und muss sich einen neuen Pfad suchen. Nach der Suche verläuft der Pfad nun

über mp10, der sich noch innerhalb der Station befindet und eine aktive Verbindung zu mpp1 besitzt. Als Resultat läuft die Verbindung zwischen mp11 und mpp1 nicht mehr direkt, sondern über den Zwischen-Hop mp10. Zu sehen ist dieser Pfadwechsel in der sekündlich geloggten Forwarding Table in Abbildung 47.

DESTADDR	NEXTHOP	IFACE	SN	METRIC	QLEN	EXPTIME	DTIM	DRET	FLAGS
14:91:82:30:c3:a7	14:91:82:30:c3:a7	mp11	1010	171	0	1380	100	0	0x15
14:91:82:30:c3:a7	14:91:82:30:c3:a7	mp11	1012	171	0	4450	100	0	0x15
14:91:82:30:c3:a7	14:91:82:30:c3:a7	mp11	1012	171	0	3410	100	0	0x15
14:91:82:30:c3:a7	c0:56:27:c8:0a:8b	mp11	1014	380	0	4150	100	0	0x15
14:91:82:30:c3:a7	c0:56:27:c8:0a:8b	mp11	1014	380	0	3110	100	0	0x15
14:91:82:30:c3:a7	c0:56:27:c8:0a:8b	mp11	1015	380	0	4410	100	0	0x15

Abbildung 47: Pfadwechsel bei Stationsausfahrt

Sobald sich mp11 im entfernten Bereich befindet, wird - je nachdem, über welchen Pfad die Metrik gerade besser ist - entschieden, worüber die Verbindung läuft. Im Test hat dies dazu geführt, dass zwischendurch der längere Pfad als besserer ausgewählt wurde, da die Metrik zwischendurch darüber einfach besser war.

Nähert sich nach einer gewissen Zeit mp11 wieder der Station, so verbindet sich dieser wieder direkt mit dem mpp1. Zu sehen ist dies in Abbildung 48.

DESTADDR	NEXTHOP	IFACE	SN	METRIC	QLEN	EXPTIME	DTIM	DRET	FLAGS
14:91:82:30:c3:a7	c0:56:27:c8:0a:87	mp11	1075	323	0	4000	100	0	0x15
14:91:82:30:c3:a7	c0:56:27:c8:0a:87	mp11	1075	323	0	2970	100	0	0x15
14:91:82:30:c3:a7	c0:56:27:c8:0a:87	mp11	1075	323	0	1930	100	0	0x15
14:91:82:30:c3:a7	14:91:82:30:c3:a7	mp11	1076	171	0	4920	100	0	0x15
14:91:82:30:c3:a7	14:91:82:30:c3:a7	mp11	1076	171	0	3880	100	0	0x15
14:91:82:30:c3:a7	14:91:82:30:c3:a7	mp11	1078	171	0	4930	100	0	0x15

Abbildung 48: Pfadwechsel bei Stationseinfahrt

Die Qualität der verschiedenen Audiostreams war über die gesamte Zeit hinweg relativ gut. Beim RTSP Stream gab es zwischendurch einige Paketverluste, was zu kleinen Stockungen im Stream führte. Der HTTP Stream hatte mit den dauernden Pfadwechseln auf Grund der Metriken im entfernten Bereich wesentlich mehr Probleme. Es kam dazu, dass der Stream zwischendurch für eine Zeit von ca. drei bis vier Sekunden unterbrochen wurde.

Die VoIP-Verbindung funktionierte wie auch schon beim Test mit der Eisenbahn zuverlässig. Auch in diesem Test hörte man zwischendurch ein «Knacksen», aber es war zu jeder Zeit möglich, verständlich mit dem Gegenüber zu sprechen.

8.4.3.4 Erkenntnisse

Der Test zeigte auf, dass die Dynamik mit der Stations-Ein- bzw. -Ausfahrt keine Probleme verursacht. Der Pfad wurde bei der Ausfahrt aktualisiert, als keine direkte Verbindung zwischen Mesh Point und Mesh Point-Portal mehr möglich war. Bei der Einfahrt wurde die direkte Verbindung wiederaufgenommen. Die VoIP-Verbindung sowie der RTSP Stream funktionierten zuverlässig, einzig der HTTP Stream war zwischendurch nicht befriedigend. Der Unterbruch von drei bis vier Sekunden konnte jedoch in einem weiteren Test mit erhöhter Buffer-Kapazität im Mediaplayer des Clients überbrückt werden.

9 Ergebnisdiskussion & Ausblick

Im folgenden Kapitel werden die wichtigsten Erkenntnisse nochmals zusammengefasst und ein Fazit gezogen.

9.1 Anpassungen Realität

Im Kapitel 8.4 wurde die entstandene Lösung in einem kleineren Modell getestet. Soll diese Lösung in einer realen Gondelbahn zum Einsatz kommen, müssen jedoch einzelne Konfigurationen den Gegebenheiten angepasst werden.

9.1.1 Time To Live

Im Path Request Frame hat es ein «Time To Live»-Feld. Dieser Wert gibt an, wie lange das Paket im Netzwerk versendet wird, bis es ein Mesh Point automatisch verwirft. Jeder Mesh Point, der diesen Path Request erhält, reduziert diesen Wert um eins. Standardmässig ist der Wert 32. Sind nun aber mehr wie 32 Mesh Points im Netzwerk, kann ein Path Request allenfalls nicht alle Mesh Points erreichen. Sind mehr als 32 Mesh Points im Netz, empfiehlt sich dieser Wert mit folgendem Befehl zu erhöhen:

```
$: iw dev <devname> set mesh_param mesh_ttl <Wert>
```

9.1.2 Antennen

Da im Modell die Distanzen zwischen den Mesh Points relativ klein waren, wurden WLAN-Adapter mit Rundstrahlantennen verwendet. Bei einer realen Gondelbahn verbinden sich die Gondeln mehrheitlich immer in dieselbe Richtung. Um bei grösseren Distanzen eine bessere Verbindung zu gewährleisten, macht es deshalb Sinn, gerichtete Antennen zu verwenden.

9.1.3 WLAN-Datenraten & Frequenz

Für die Durchführung von Tests und die Überprüfung auf korrekte Funktionalität des Mesh-Netzwerks wurden die High Throughput-Datenraten deaktiviert. Dies ermöglichte eine genauere Analyse der einzelnen Verbindungen in Wireshark. In einer produktiven Umgebung sollten die maximal möglichen Datenraten aktiviert werden, um die höchstmögliche Bandbreite erreichen zu können. Ausserdem wäre ein Kanalwechsel prüfenswert, um allfällige Störungen minimieren zu können.

9.2 Ergebnisdiskussion

Die Implementationstests haben gezeigt, dass die ausgewählte Technologie beziehungsweise die eigene Implementierung im Grundsatz gut funktioniert. Mesh-Netzwerke können mit dem Protokoll 802.11s relativ einfach aufgebaut werden. Der reaktive Pfadfindungsmechanismus arbeitet sehr schnell und belastet das Netzwerk nicht unnötig mit Kontrollnachrichten. Werden nur einzelne und kleine Datenpakete im Netzwerk versendet, funktioniert dies über das Mesh-Netzwerk ohne weitere Probleme. Bei grösseren und kontinuierlicheren Datenströmen kommt das Netzwerk aber vermehrt an seine Grenzen. Hier ist es wichtig, dass die verwendeten Applikationen auch in der Lage sind, mit den Eigenheiten eines Mesh-Netzwerkes umgehen zu können. Insbesondere sind hier auch die Eigenschaften von WLAN zu beachten.

9.2.1 802.11s

Die Auswahl vom 802.11s Protokoll war sicher richtig. Der Test «Übertragungsknoten verlässt das Netzwerk» war im ersten Moment unbefriedigend. Weiter Untersuchungen zeigten jedoch, dass ein Path Error erst ausgelöst wird, wenn eine gewisse Anzahl von WLAN Acknowledgements ausbleibt.

Mit dieser Erkenntnis war ein weiteres kritisches Gegenargument entkräftet. Zumal in der bereits erwähnten Literatur [9] beschrieben wird, dass andere Mesh-Protokolle wesentlich schlechter sind.

9.2.2 Gondelbahnszenarien

Beim Szenario «Kreuzen in voller Fahrt» wurde festgestellt, dass die Gondeln auch Pfade über nur kurzfristig sichtbare Gondeln aufbauen. Dies kann in einem normalen dynamischen Netzwerk durchaus gewünscht sein und ist keineswegs ein Fehler in der Implementierung. Jedoch ist dies im Gondelbahn Szenario nicht nötig. Ein entwickeltes Skript löst dieses Problem. Die Mesh Points verbinden sich nur noch mit Mesh Points, welche über längere Zeit gesehen wurden. Somit kann der unnötige Pfadaufbau verhindert werden und im ganzen Netzwerk wird mehr Stabilität erreicht.

Im zweiten Szenario «Gondel fährt in Station» wurde intensiv getestet, was geschieht, wenn eine Gondel die Station durchfährt und sich das Routing dementsprechend ändert. Dieses Szenario wurde im Modell getestet und die Ergebnisse zeigten auf, dass das Mesh-Netzwerk in diesem Fall richtig reagiert und Daten weiterhin erfolgreich versendet werden können.

9.2.3 Sprachdurchsagen

Die Idee, Sprachdurchsagen per Multicast an die verschiedenen Gondeln zu übertragen, ist durch die Gegebenheiten im WLAN-Mesh-Netzwerk nur schlecht möglich. Es kann per Multicast nicht garantiert werden, dass alle Mesh Points im Netzwerk erreicht werden. Zusätzlich können Interferenzen die WLAN-Verbindung empfindlich stören. Die Erkenntnis, dass WLAN Acknowledgements nur bei Unicast-Verbindungen ausgelöst werden, macht eine Multicast-Verbindung unsicher.

Die Sprachdurchsagen per RTP Stream zu versenden, führte zu mehr Erfolg. Die Sprachdurchsagen waren deutlich besser hörbar. Dadurch, dass jedes versendete Paket mit einem WLAN Acknowledgement bestätigt werden sollte und bei allfälligen Übertragungsfehlern Retransmissions durchgeführt werden, gehen weniger Pakete verloren, als bei Multicast.

Eine Garantie, dass sämtliche Pakete beim Mesh Point ankommen, ist nur mittels TCP möglich. Während in einem statischen Mesh-Netzwerk bei TCP keine grösseren Probleme auftreten, verursacht eine solche gesicherte Verbindung bei den ständigen Topologie-Wechseln in der Talstation mehrheitlich Probleme.

9.2.4 Gegensprechanlage

Die Tests der Gegensprechanlage mittels Voice over IP verliefen sehr positiv. Hat ein Passagier in einer Gondel ein Problem, kann er über die Gegensprechanlage mit der Station Kontakt aufnehmen. Die Verbindungsunterbrüche während den Pfadwechseln waren sehr klein und in einem normalen Gespräch fast nicht hörbar.

9.3 Ausblick

Eine Gondelbahn könnte mit der evaluierten Technologie vernetzt werden. Nebst Konfigurationsanpassungen gibt es zudem noch weitere Änderungen und Funktionen die denkbar wären um ein solches Gondelbahn Mesh-Netzwerk zu realisieren.

9.3.1 Wartung & Betrieb

Sind Änderungen am Verbesserungsskript oder an der Mesh-Konfiguration nötig, muss in der aktuellen Situation jeder Raspberry Pi einzeln konfiguriert werden. Dies ist nicht sehr wartungsfreundlich und kleine Änderungen dauern in der Implementierung verhältnismässig lange. Wird zum Beispiel im aktuellen Fall das Mesh Point Portal ersetzt, muss auf allen Raspberry Pi das Verbesserungsskript angepasst werden.

Ein zentrales Systemkonfigurationswerkzeug könnte hier helfen die Raspberry Pi bei Bedarf automatisch neu zu konfigurieren. Sollten die Änderungen im laufenden Betrieb umgesetzt werden, wäre auch ein Controller denkbar. Mit einem solchen Controller könnten sogar dynamische Änderungen eingespielt werden, um das Netzwerk zu beeinflussen. Gerade bei speziellen Wetterverhältnissen könnten sich solche Änderungen positiv auf das Netzwerk auswirken.

9.3.2 Monitoring

Bisher wurden die Verbindungsinformationen einzeln auf jedem Raspberry Pi ausgewertet. Mithilfe von Logs wurde überprüft, ob die Verbindungen und Forwarding Tables korrekt waren. In einer produktiven Umgebung wäre aber eine Überwachung aller Mesh Point wünschenswert. Die Mesh Points könnten ihre Verbindungsinformationen an eine zentrale Instanz senden, die diese Informationen aufbereitet und in einer visualisierten Topologie darstellt. Diese Statusinformationen sollte aber nicht zu häufig ausgetauscht werden, da ansonsten der Vorteil der reaktiven Pfadfindung verloren geht. Somit könnte auch überprüft werden, ob noch alle Mesh Points korrekt funktionieren.

Monitoring und die bereits erwähnte Controllerfunktionalität könnten sich zudem optimal ergänzen. Eine zentrale Managementfunktionalität gibt es für 802.11s basierte Mesh-Netzwerke derzeit noch nicht.

9.3.3 Proaktives Routing

Bisher wurde im Dokument nur wenig über die proaktiven Möglichkeiten von HWMP beschrieben. In einem kleinen Mesh-Netzwerk mit wenigen Mesh Points macht die Verwendung der proaktiven Funktionen nur wenig Sinn. Bei einem Mesh-Netzwerk mit vielen Mesh Points ist diese Funktion sicher prüfenswert.

9.3.4 Audiostream

Während der Übertragung kam es hin und wieder zu kleineren Unterbrüchen und Störungen. Mit einem besseren Codec können sicher kleinere Störungen überspielt werden.

9.3.5 Multicast

Damit Multicast im Mesh-Netzwerk auch zuverlässig funktioniert, müsste die Stromverteilung mit einer gewissen Intelligenz ergänzt werden. Eine Möglichkeit wäre, den Stream mehrfach auszusenden, damit die Wahrscheinlichkeit einer erfolgreichen Übertragung steigt. Eine zusätzliche Software müsste dann aber die mehrfach erhaltenen Pakete filtern, damit einzelne Audioteile nicht wiederholt abgespielt werden.

Blockiert ein Mesh Point eine Verbindung zu einem anderen Mesh Point, werden auch keine Multicast-Pakete darüber versendet. Das entwickelte Skript verhindert somit eine bessere Ausbreitung der Multicast-Signale im Netzwerk. Mit einem zusätzlichen Programm könnte dieses Skript ausgeschaltet und kurzfristig alle Verbindungen wieder erlaubt werden.

10 Glossar

AMPE	Authenticated Mesh Peering Exchange *Peering Management Methode, die im Zusammenhang mit SAE benötigt wird, um gesicherte Peerings zu ermöglichen.*
ALM	Airtime Link Metric *Metrik für die Bestimmung eines besten Routing-Pfades anhand der Qualität des WLAN-Signals.*
AODV	Ad-Hoc On Demand Distance Vector *Reaktives Routingverfahren in einem mobilen Ad-Hoc-Netzwerk.*
Beacon	*Management-Frame im WLAN, das eine Station kontinuierlich aussendet um Informationen über sich preiszugeben.*
HWMP	Hybrid Wireless Mesh Protocol *Routing-Protokoll, welches standardmässig in 802.11s implementiert ist.*
Jiffies	*Globale Variable, die angibt wie viele Ticks seit dem Systemstart stattgefunden haben.*
OGM	Originator Message *Kontroll-Pakete welche bei B.A.T.M.A.N. für die Pfadfindung benötigt werden.*
RTP	Real-Time Transport Protocol *Protokoll zur Übertragung von Multimedia-Datenströmen im IP-Netzwerk.*
RTSP	Real-Time Streaming Protocol *Netzwerkprotokoll, mit dem die Übertragung von audiovisuellen Inhalten im IP-Netzwerk gesteuert werden kann*
SAE	Simultaneous Authentication of Equals *Passwort-basiertes Authentifizierungs-Protokoll, das zum Aufbau von gesicherten Peerings im Mesh Netzwerk benötigt wird.*
SIP	Session Initiation Protocol *Netzwerkprotokoll zur Steuerung einer Kommunikationssitzung zwischen mehreren Teilnehmern.*
Transcoding	*Umwandeln einer Audio- oder Video-Datei in ein anderes Format.*
TU	Time Unit *Zeiteinheit in Linux, abgestimmt auf CPU-Clocktime, 1 TU entspricht 1024 Mikrosekunden.*

11 Literaturverzeichnis

[1] X. Hong, K. Xu und M. Gerla, «Scalable routing protocols for mobile ad hoc networks,» *IEEE Network,* Nr. 16, p. 11–21, 2002.

[2] L. Zhao und A. Y. Al-Dubai, «Routing Metrics for Wireless Mesh Networks: a survey,» [Online]. Available: http://www.napier.ac.uk/~/media/worktribe/output-293859/csieliangzhaopdf. [Zugriff am 17 Dezember 2016].

[3] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,* Piscataway, NJ, USA: IEEE Standard Association, 2012.

[4] T. Clausen und P. Jacquet, «Optimized Link State Routing Protocol (OLSR),» Oktober 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3626.txt. [Zugriff am 17 Dezember 2016].

[5] T. Clausen, C. Dearlove, U. Herberg und P. Jacquet, «The Optimized Link State Routing Protocol Version 2,» April 2014. [Online]. Available: https://www.ietf.org/rfc/rfc7181.txt. [Zugriff am 17 Dezember 2016].

[6] S. R, Belding-Royer, E. M, Perkins und C. E, «Ad hoc On-Demand Distance Vector (AODV) Routing,» Juli 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3561.txt. [Zugriff am 14 Dezember 2016].

[7] «B.A.T.M.A.N. protocol concept,» Open Mesh, [Online]. Available: https://www.open-mesh.org/projects/open-mesh/wiki/BATMANConcept. [Zugriff am 17 Dezember 2016].

[8] «B.A.T.M.A.N. advanced,» Open Mesh, [Online]. Available: https://www.open-mesh.org/projects/batman-adv/wiki/Wiki. [Zugriff am 17 Dezember 2016].

[9] R. G. Garroppo, S. Giordano und L. Tavanti, «Experimental evaluation of two open source solutions for wireless mesh routing at layer two,» in *IEEE 5th International Symposium on Wireless Pervasive Computing 2010*, 2010, pp. 232-237.

[10] M. Bahr, «Proposed Routing for IEEE 802.11s WLAN Mesh Networks,» [Online]. Available: http://dev.laptop.org/~cscott/802.11s/80211s-a5-bahr.pdf. [Zugriff am 17 Dezember 2016].

[11] M. Singh, L. Song-Gon und L. HoonJae, «Non-root-based hybrid wireless mesh protocol for wireless mesh networks,» *International Journal of Smart Home,* p. 71–83, 2013.

[12] C. E. Perkins und P. Bhagwat, «John Hopkins Whiting School of Engineering: Highly Dynamic Destination-Sequenced Distance-Vector Routing,» 1994. [Online]. Available: http://www.cs.jhu.edu/~cs647/class-papers/Routing/p234-perkins.pdf. [Zugriff am 19 Dezember 2016].

[13] D. Johnson, Y. Hu und D. Maltz, «The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4,» Februar 2007. [Online]. Available: https://tools.ietf.org/rfc/rfc4728.txt. [Zugriff am 19 Dezember 2016].

[14] L. C. Cobo, «Github Linux Kernel Source Code,» [Online]. Available: https://github.com/torvalds/linux/blob/master/net/mac80211/mesh_hwmp.c. [Zugriff am 15 12 2016].

[15] J. Rech, Wireless LANs - 802.11-WLAN-Technologie und praktische Umsetzung im Detail (3. Aufl.), Heise, 2008.

[16] J. Henry, «cwnp,» November 2011. [Online]. Available: https://www.cwnp.com/uploads/802-11s_mesh_networking_v1-0.pdf. [Zugriff am 17 Dezember 2016].

[17] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley und E. Schooler, «SIP: Session Initiation Protocol,» Juni 2002. [Online]. Available: https://www.ietf.org/rfc/rfc3261.txt. [Zugriff am 17 Dezember 2016].

[18] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobson, «RTP: A Transport Protocol for Real-Time Applications,» Juli 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3550.txt. [Zugriff am 17 Dezember 2016].

[19] H. Schulzrinne, A. Rao und R. Lanphier, «Real Time Streaming Protocol (RTSP),» April 1998. [Online]. Available: https://www.ietf.org/rfc/rfc2326.txt. [Zugriff am 17 Dezember 2016].

12 Abbildungsverzeichnis

13 Tabellenverzeichnis

14 Formelverzeichnis

II Anhänge

Anhang I: Persönliche Berichte
Anhang II: Projektplan
Anhang III: Handbuch Systemumgebung
Anhang IV: Testplan
Anhang V: Eigenständigkeitserklärung
Anhang VI: Urheber- und Nutzungsrechte