# Analyse von DHT-basierten Routingverfahren in drahtlosen Maschennetzwerken


**Humboldt-Universität zu Berlin**
**Mathematisch-Naturwissenschaftliche Fakultät II**
**Institut für Informatik**

Studienarbeit

Christoph Bauer

05.07.2012

Betreuer: Robert Sombrutzki

# Zusammenfassung

Die Wahl eines Routingverfahrens ist maßgeblich für die Leistungsfähigkeit von Kommunikationssystemen. In drahtlosen Maschennetzwerken sind die Anforderungen besonders hoch, denn dort müssen die Verfahren Skalierbarkeit, Mobilität und Robustheit garantieren können. Viele Routingalgorithmen sind speziell für verdrahtete, ortsgebundene Netzwerke wie das Internet entwickelt worden und können entweder gar nicht oder nur mit großem Leistungsverlust in drahtlosen Systemen verwendet werden. Bereits entwickelte Routingprotokolle, die auf einer verteilten Hashtabelle basieren, erfüllen viele der Eigenschaften, die ein Maschennetzwerk benötigt und sind deswegen ein großer Bestandteil aktueller Forschungen.
In dieser Arbeit wird eines dieser Protokolle, BRN-DART, genauer betrachtet und auf Optimierungsmöglichkeiten bezüglich der Wegewahl untersucht. Es werden drei Verbesserungsstrategien vorgestellt und die durch sie erzeugten Routen in einer Simulation mit optimalen Routen verglichen.
Es zeigt sich, dass jede Strategie eine Annäherung an das Optimum bezüglich *Hop Count*-Metrik und ETX-Metrik bewirkt. Insbesondere ließen sich beide Metriken bei über 50% der Routen um mehr als 50% in ihren Werten reduzieren.

# Inhaltsverzeichnis

# 1 Einleitung

Drahtlose Maschennetzwerke finden aufgrund ihrer besonderen Eigenschaften immer häufiger Anwendung. Diese sind:

- **Selbstorganisation:** Das gesamte (Overlay-)Netz wird automatisch aufgrund bestimmter Regeln aufgebaut.
- **Skalierbarkeit:** Es dürfen (theoretisch) beliebig viele Knoten zum Netzwerk hinzukommen oder es verlassen.
- **Zuverlässigkeit:** Netzwerkdienste funktionieren auch nach Beitreten oder Verlassen von Knoten.

Dank dieser Eigenschaften ist es möglich Daten, Ressourcen oder Dienste mithilfe sogenannter DHT-Protokolle auf alle Knoten des Netzwerks zu verteilen. Die DHT-Protokolle verwenden dazu Hashtabellen um neue Daten speichern und gespeicherte Daten suchen zu können. Mögliche Einsatzbereiche dafür finden sich bei Katastrophenfrühwarnsystemen, Sensornetzwerken, verteilten Datenbanken, Tauschbörsen oder gemeinsamer Nutzung eines geteilten Internetzugangs in infrastrukturell benachteiligten Gebieten.
Damit sowohl die Suche als auch die Speicherung effizient ausgeführt werden kann, erzeugen viele DHT-Protokolle aus den Netzwerkknoten eine eigene virtuelle Netzwerkstruktur. Sie kann - bei geeigneter Erzeugung - genutzt werden um Pakete gezielt durch das Netzwerk zu schicken und das Fluten des Netzwerks zu verhindern. Dabei sind, um dies zu gewährleisten, nur wenig (häufig $O(log\ n)$) Routinginformationen pro Knoten nötig.
Die virtuelle Struktur wird völlig unabhängig von der wirklichen/physikalischen Struktur des Netzwerks erzeugt und hat deswegen meist nichts mit ihr gemeinsam. Darum ist es nötig das in der virtuellen Struktur stattfindende Routing auf die physikalische Netzwerkstruktur zu übertragen. Dies kann auf zwei Arten getan werden: Einige DHT-Protokolle benötigen ein zusätzliches Routingprotokoll, welches das Routing im physikalischen Netzwerk übernimmt. Andere betreiben das Routing dort selbst auf Basis ihrer DHT. Eines dieser DHT-basierten Routingprotokolle liefert das DHT-Protokoll DART und soll in dieser Arbeit genauer betrachtet werden.

Diese Arbeit ist wie folgt aufgebaut. In Kapitel 2 werden drahtlose Maschennetzwerke und ihre Verbindung zu DHTs erläutert und das Konzept von Routingverfahren erklärt. Danach wird in Kapitel 3 und 4 das DHT-basierte Routingverfahren DART vorgestellt und eine nach diesem Konzept implementierte Version beschrieben. Zu der Implementierung wurden mögliche Optimierungen bezüglich kürzerer Wege bei der Weiterleitung untersucht. Die Ergebnisse werden in Kapitel 5 mit dem Standardverfahren verglichen und es wird gezeigt, dass deutliche Verbesserungen erzielt werden konnten. Schließlich wird in dieser Arbeit ein Ausblick zu weiteren Optimierungsmöglichkeiten gegeben.

# 2 Grundlagen

## 2.1 Drahtlose Maschennetzwerke

Drahtlose Maschennetzwerke sind zufällig strukturierte Netze, deren Teilnehmer durch Funkverbindungen miteinander kommunizieren. Sie bestehen aus verschiedenen Arten von Knoten: Maschenknoten (*Mesh Nodes*)und Klienten (*Client Stations*)(Abb. 1). Maschenknoten bilden das Netzwerk und bestimmen somit dessen Topologie. Einige von ihnen dienen den Klienten als Access Points[1], andere stellen diverse Dienste wie z.B. Verbindungen zum Internet zur Verfügung. Damit auch Teilnehmer, die keinen direkten Zugang zu einem der Dienste haben, diesen trotzdem nutzen können, leiten benachbarte Maschenknoten die Pakete für sie weiter (*Multi Hop Routing*).
Klienten nehmen als Endgeräte die Dienste des Maschennetzwerks in Anspruch, sie bieten selbst keine an. Klassische Klienten sind Smartphones, Laptops und Tablets.

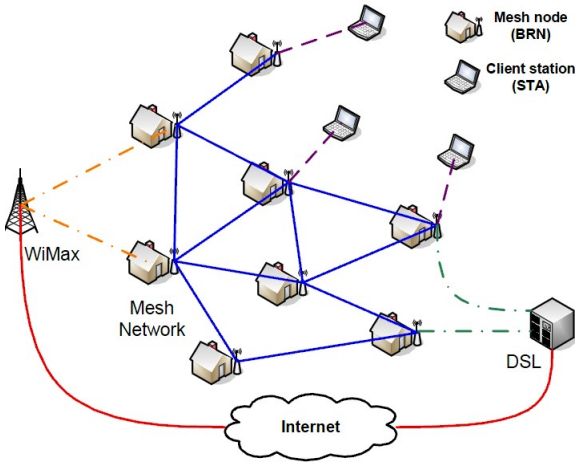



Abbildung 1: Beispiel eines drahtlosen Maschennetzwerks

Drahtlose Maschennetzwerke sind dezentralisiert, d.h. sie besitzen keinen zentralen Server, der alle Dienste bereitstellt und die Organisation des Netzes übernimmt. Dienste werden stattdessen auf alle Maschenknoten gleichermaßen verteilt.
Eine weitere Eigenschaft drahtloser Maschennetzwerke ist die Selbstorganisation ([1]). Jeder Maschenknoten kann selbstständig neue Nachbarknoten entdecken und sie in das Netzwerk integrieren. Ebenso können Ausfälle oder Austritte der Nachbarknoten erkannt werden. Das ist wichtig, da sich Teilnehmer bewegen können und das Netzwerk sich den Standortänderungen anpassen können muss. Durch die Bewegungen ändert sich auch die Topologie des Netzwerkes. Zu einem drahtlosen Maschennetzwerk kann also keine klare Aussage über die Struktur getroffen werden. Es ist lediglich bekannt, dass von einem beliebigen Knoten jeder andere Knoten erreicht werden kann. Dabei existieren zwischen den Knoten normalerweise mehrere Wege, die benutzt werden können. So kann einer-

[1]Ein Access Point ist ein Zugangsknoten zum Netzwerk. Er verbindet Geräte drahtlos miteinander. Dabei vermeidet er Datenkollisionen und überbrückt Unterschiede zwischen verschiedenen Übertragungsmedien.

seits eine bessere Verteilung des Netzwerkverkehrs erzielt werden, andererseits können ausgefallene Wege durch Alternativrouten[1] ersetzt werden.

## 2.2 Peer-To-Peer-Netze

Anhand der Aufgabenverteilung ihrer Knoten werden Netzwerke nach zwei Modellen unterschieden. Im *Client-Server*-Modell übernehmen Knoten entweder die Rolle eines *Servers* oder die eines *Clients*. *Server* organisieren das Netzwerk und stellen Dienste oder Ressourcen zur Verfügung, *Clients* können die Dienste anfordern. Im *Peer-To-Peer*-Modell (P2P) ist diese Rollenverteilung aufgehoben. Alle Knoten werden als gleichberechtigt angesehen, denn sie stellen gleichzeitig *Server* und *Client* dar.
Der Vorteil von *Peer-To-Peer*-Systemen ist, dass sie Daten, Ressourcen oder Anwendungen auf ihre Knoten verteilen können. Dadurch ist ein Dienst über alle Maschenknoten gleichzeitig nutzbar. Damit ein *Client* in solch einem System einen bestimmten Teil des Dienstes wiederfindet, muss ein Suchverfahren für das Netzwerk definiert sein. Dieses richtet sich nach der Strukturierung des Netzwerks. Es gibt zwei Möglichkeiten ein P2P-Netz aufzubauen:

- **unstrukturiert** - das Netz ändert sich nichtdeterministisch beim Hinzufügen der neuen Knoten oder Daten
- **strukturiert** - die Netzstruktur ändert sich nach einigen bestimmten Regeln.

Unstrukturierte P2P-Systeme besitzen beliebige Verbindungen zwischen den Knoten, sodass eine vermaschte Topologie entsteht. Um in einem solchen Netz Datenobjekte zu finden muss die Suchanfrage durch das gesamte Netz geflutet werden. Das erzeugt aber einen enormen Datenverkehr (*Broadcast Storm Problem*[2]), der in strukturierten P2P-Netzen nicht vorkommt. Dort werden Daten und Knoten nach bestimmten Regeln einander zugeordnet. Dadurch kann eine Suchanfrage gezielt zum zuständigen Knoten weitergeleitet werden. Auch in unstrukturierten Netzen kann dieses Prinzip genutzt werden. Dazu bildet man ein strukturiertes *Peer-To-Peer*-System als virtuelles Overlay-Netzwerk auf das unstrukturierte physikalische Netzwerk ab (siehe Abb. 2) und kann dort die Suche durchführen. Einen möglichen Ansatz solcher Suchverfahren bieten *Skip*-Listen bzw. *Skip*-Graphen [3], einen anderer Ansatz sind DHT-Protokolle.

## 2.3 Distributed Hash Tables

Eine Hashtabelle ist eine Datenstruktur, die Paare aus Schlüssel und Wert enthält. Sie ermöglicht eine sehr schnelle Suche nach Daten zu einem bestimmten Schlüssel. *Peer-To-Peer*-Systeme nutzen diese Effizienz für die verteilte Datenspeicherung aus und verteilen Daten mithilfe einer Hashtabelle auf alle Knoten. Das führt zur Entlastung der Knoten, die als alleinige Speichermedien durch ständige Anfragen überlastet wären.

[1] Eine Route ist eine Abfolge von Direktverbindungen (auch *Links* oder *Hops* genannt), die benutzt wird, um Nachrichten von einem Knoten zu einem anderen durch das Netzwerk zu übertragen.

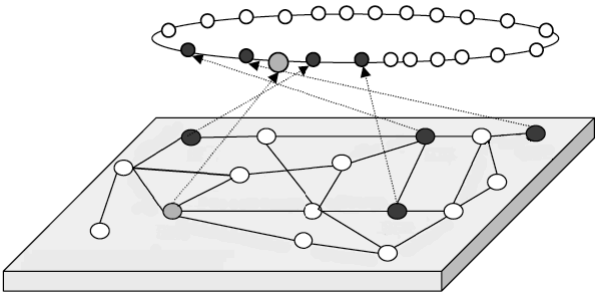


Abbildung 2: *Peer-To-Peer*-Overlaynetzwerk als Ringtopologie

Das dafür benutzte Prinzip der verteilten Hashtabelle (engl. ***D**istributed **H**ash **T**able*) ([13],[16],[17],[18]) ist folgendes.
Alle Datenobjekte erhalten durch eine global bekannte Hashfunktion[1] Schlüssel aus einem linearen Wertebereich. Der entstehende Schlüsselraum wird auf alle Netzwerkknoten verteilt. Jeder Knoten ist dann für einen Schlüsselbereich und dessen zugeordnete Datenobjekte zuständig. Anwendungen können nun über eine einheitliche Schnittstelle die verteilte Hashtabelle benutzen: Mit der Funktion $publish(key, value)$ wird ein neues Datum *value* an den für den Schlüssel *key* zuständigen Knoten vermittelt. Mit $lookup(key) \rightarrow value$ werden in dem für den Schlüssel zuständigen Knoten die gewünschten Daten gesucht und der Anwendung übergeben (siehe Abb. 3). Durch das gleichbleibende Interface können die DHTs einfach ausgetauscht werden.

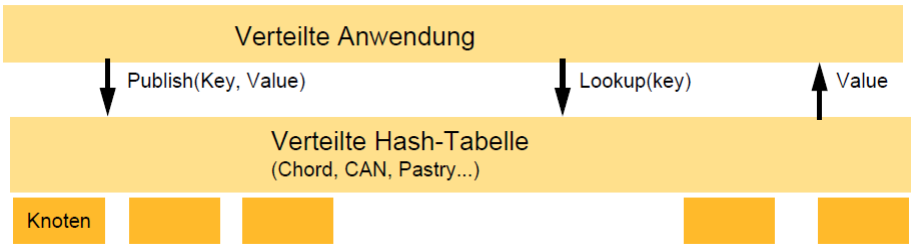
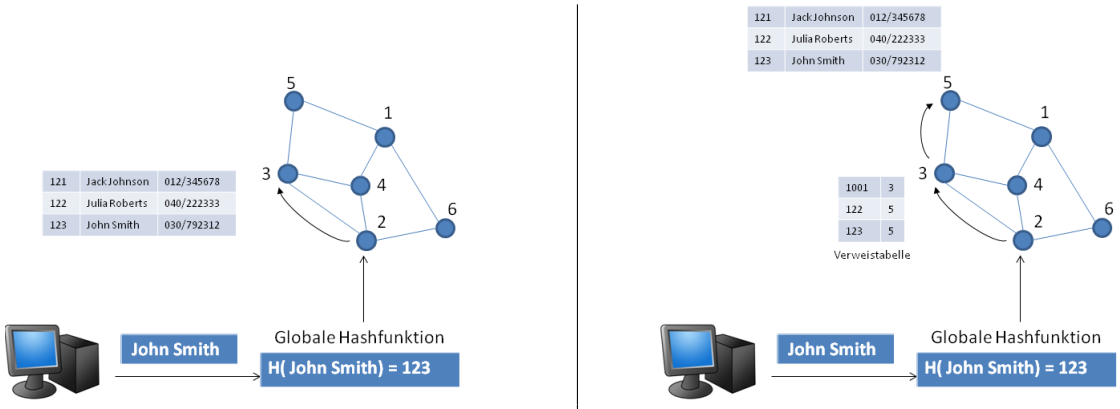



Abbildung 3: Schnittstelle verteilter Hashtabellen

Beide Funktionen benötigen eine Suchvorschrift um den passenden Knoten zu einem Schlüssel gezielt zu finden. Diese wird in einem der DHT zugehörigen Protokoll beschrieben und orientiert sich an der Netzwerkstruktur, die von der DHT vorausgesetzt wird. So ist Chord ([4]) beispielsweise eine DHT, die eine ringförmige Anordnung benutzt, Pastry ([**?**]) ist eine DHT, die eine Baumtopologie benutzt. Oftmals liegt die benötigte

[1] Hashfunktionen sind math. Abbildungen, die aus einer oft sehr großen Wertemenge eine Ausgabe aus einer kleineren Wertemenge erzeugt, dem sogenannten Hashwert. Beispiel: hash(i) = i mod N , wobei N eine Primzahl ist.

Struktur dem Netzwerk nicht zugrunde, weswegen die DHT in ein passendes virtuelles *Peer-To-Peer*-Netz integriert wird.
Für die Implementierung der Suche muss die Art der Speicherung berücksichtigt werden, denn hier gibt es zwei Möglichkeiten (Abb. 4):




Direktes Speichern | Indirektes Speichern

Abbildung 4: Verwendung einer DHT als Telefonbuch

Beim direkten Speichern speichert der zuständige Knoten ein (Schlüssel,Wert)-Paar direkt. Eine Suchanfrage kann von ihm sofort beantwortet werden. Diese Variante wird nur bei Daten $<$ 1kByte benutzt, da die Effizienz der DHT sonst nachlässt. Hintergrund ist die Neuaufteilung der Daten bei Verlassen oder Hinzukommen eines Knotens. Die Neuaufteilung erfordert einen Datenaustausch zwischen den Knoten und belastet das Netzwerk mit zunehmender Datengröße. Beim indirekten Speichern liefert der zuständige Knoten lediglich einen Verweis auf den Knoten, der das Datum speichert. Das hat den Vorteil, dass bei Neuverteilung der Bereiche die Daten selbst nicht ausgetauscht werden müssen, lediglich die Verweise werden durch das Netzwerk geschickt. Außerdem können Knoten, die aufgrund ihres zuständigen Schlüsselraums sehr viele Daten speichern müssen, sehr einfach entlastet werden, indem sie Teile ihrer Daten anderen Knoten zuteilen und nur die Verweise dorthin speichern müssen. Dies wäre beispielsweise für Telefonbücher nützlich, da ein für den Buchstaben $'S'$ zuständiger Knoten mehr Daten speichern müsste als ein Knoten, der für Namen zuständig ist, die mit $'Q'$ beginnen. Nachteil ist, dass Anfragen oft länger benötigen, da erst der Verweis und dann der Speicherort gesucht werden muss.
Um den Datenaustausch bei Verlassen und Beitreten von Knoten minimal zu halten werden für DHTs zusätzlich konsistente Hashfunktionen gewählt. Kommt ein neuer Knoten hinzu oder verlässt ein Knoten das Netzwerk, dann wird im Gegensatz zu einer inkonsistenten Hashfunktion der Schlüsselraum nicht komplett neu an alle vorhandenen Knoten verteilt, sondern nur an umliegende.
Das DHT-Protokoll definiert neben dem Suchverfahren auch das dafür notwendige Routing (siehe Kapitel 2.4), da nur so die Knoten Informationen erhalten, an welche Nach-

barknoten sie Suchanfragen weiterleiten können. Außerdem organisiert es die Aufnahme neuer Knoten in die existierende DHT, sowie die Entfernung nicht mehr vorhandener Knoten.
Die allgemeinen charakteristischen Eigenschaften von *Distributed Hash Tables* verdeutlichen die Ähnlichkeit zu drahtlosen Maschennetzwerken:

- **Selbstorganisation:** das gesamte (Overlay-)Netz wird automatisch aufgrund bestimmter Regeln aufgebaut.
- **Skalierbarkeit:** das (Overlay-)Netz sollte die Möglichkeit zu Erweiterungen auf große Anzahl der Knoten erlauben.
- **Lastenverteilung:** Die Daten werden möglichst gleichmäßig auf alle Knoten verteilt.
- **Robustheit:** Das System sollte zuverlässig sein, falls einige Knoten ausfallen oder das System verlassen.

## 2.4 Routing

Mit Routing wird im Allgemeinen die Vermittlung von Nachrichten in einem Netzwerk bezeichnet. Es beschreibt die Aufgaben der Vermittlungsschicht im *Open Systems Interconnection*-Referenzmodell (OSI)([5], S.581). Grundlegende Bestandteile des Routings sind Routing-Algorithmus und Weiterleitung (*Forwarding*). In Abbildung 5 ist die Zusammenarbeit beider Komponenten zu sehen.

### 2.4.1 Arbeitsweise

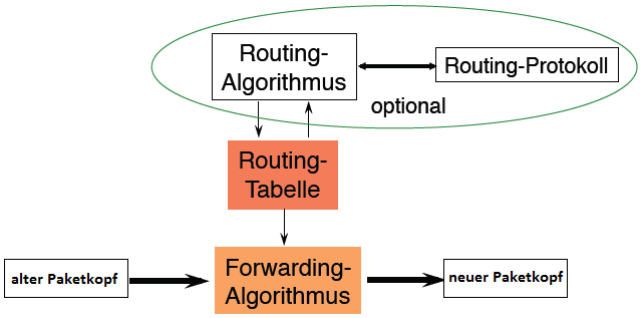



Abbildung 5: Routingschema

Der Routingalgorithmus verwendet ein Protokoll, das für jeden Knoten Kenntnisse über mögliche Übertragungsrouten zusammenträgt. Die gefundenen Routen werden in der Routingtabelle eines Netzwerkknotens festgehalten. In dieser kann er mögliche Routen

nachschlagen um Pakete gezielt weiterleiten zu können. Die Wahl der (besten) Route ist Aufgabe des *Forwarding*-Algorithmus.

### 2.4.2 Routing-Protokolle

Das Routingprotokoll legt fest, wie sich Knoten ihr Wissen über das Netzwerk austauschen. Grundlegende Verfahrensweisen sind Distance Vector Routing (z.B. AODV [7]) und Link State Routing (z.B. OLSR [8]). Beim Link State Routing sendet ein Knoten nur Informationen, die er über seine Nachbarn hat, durch das ganze Netz. Nach endlicher Zeit kennt jeder Knoten die gesamte Netzwerktopologie und kann selbst den für sich geeignetsten Weg berechnen.
Beim Distance Vector Routing hingegen erhält jeder Knoten nur eine eingeschränkte Sicht auf die Netzwerktopologie. Er seinen Nachbarn, wie gut andere Knoten von ihm aus erreichbar sind. Durch dieses Verfahren wird die Bestimmung eines geeigneten Weges schon während des Informationsaustauschs ermittelt.
Die Verbreitung von Routinginformationen kann statisch oder adaptiv erfolgen. Bei statischem Routing werden keine Routingprotokolle benötigt. Die Routingtabellen werden bspw. durch einen Netzwerkadministrator initial gefüllt und ändern sich danach nicht mehr. Dadurch entsteht keine zusätzliche Belastung des Kommunikationssystems. Bei adaptivem Routing übernehmen die Routingprotokolle diese Aufgabe. Die Einträge in den Routingtabellen werden durch sie bei Topologieänderungen des Netzwerks angepasst. Dadurch können Netzwerke flexibel sein und Wegstörungen, verursacht durch Knotenausfall, Verbindungsproblemen, Signalstörungen u.a., können umgangen werden.
Adaptive Routingprotokolle unterscheiden sich zusätzlich durch den Zeitpunkt der Informationsbeschaffung. Werden die Routinginformationen beschafft, bevor sie für die Übertragung von Nutzdaten benötigt werden, spricht man von proaktiven Protokollen. Sie aktualisieren das Wissen der Knoten in bestimmten zeitlichen Abständen, bzw. bei Änderungen in der Netzwerktopologie. In reaktiven Protokollen wird eine Route nur auf Anforderung, also zum Zeitpunkt einer Suchanfrage entdeckt. Zum Vergleich mit proaktiven Algorithmen ist der Energieverbrauch in reaktiven Protokollen effizienter und der Speicherbedarf ist geringer. Andererseits wird der Zeitpunkt verzögert, an dem der Empfänger das Paket enthält.

### 2.4.3 Forwarding

Jeder Neztwerkknoten benutzt den *Forwarding*-Algorithmus um zu entscheiden, welcher Nachbarknoten der geeignetste ist um ein Paket zum Zielknoten weiterzuleiten. Um diesen zu finden werden die Einträge in der Routingtabelle benutzt. Dabei können abhängig vom Routingverfahren in der Tabelle auch mehrere Routen zu dem gleichen Ziel vorhanden sein. Dann werden zusätzliche Informationen, wie das Alter des Tabel-

leneintrags, der Verbindungsstatus oder eine Metrik[1] zu den Routen gespeichert um die bestmögliche Verbindung wählen zu können. Empfängt ein Knoten ein weiterzuleitendes Paket ist das *Forwarding* außerdem für die Anpassung der Steuerinformationen im Paketkopf zuständig. Dies sind z.B. die Adresse der nächsten Zwischenstation, die maximale *Hop*-Anzahl des Pakets oder eine Aufzeichnung der bisher genommenen Route. Die Aktualisierung der nächsten Zwischenstation ist besonders in drahtlosen Netzwerken wichtig, da mehrere Nachbarknoten das Paket empfangen können. Ein empfangender Knoten kann so feststellen, ob er derjenige ist, der das Paket bearbeiten bzw. weiterleiten muss.

### 2.4.4 DHT-basiertes Routing

DHTs wurden für die Verteilung von Daten und Ressourcen in drahtlosen Netzwerken entwickelt. Die Verteilung neuer Daten und die Suche nach bereits verteilten Daten soll dabei möglichst effizient erfolgen. Drahtlose Netzwerke machen es jedoch durch ihre unbekannten und veränderlichen Topologien schwierig effektiv Routen zwischen zwei Knoten zu bestimmen. Bis dato entwickelte Routingverfahren wie OLSR, AODV oder DSR ([9]) hatten für die Wegfindung folgenden Lösungsansatz. Ist ein direkter Weg zu einem anderen Knoten nicht bekannt, wird das Netzwerk standardmäßig mit *Route Requests*[1] geflutet und so der Weg ermittelt. Der Netzwerkverkehr ist dementsprechend hoch und kann nur teilweise eingeschränkt werden. Ansätze wie *Hop*-Limits[2] für die *Route Requests* oder das Belauschen der Nachbarn um existierende Wege zu erfahren helfen den Datenverkehr etwas zu verringern.

Ein ganz anderer Lösungsansatz für das Routing in drahlosen Netzwerken entstand mit der Entwicklung der DHTs. Durch die Verwendung virtueller Overlaynetzwerke war es nun möglich Routingverfahren auf eine feste Struktur auszurichten. Da sich die Overlaystruktur in ihrer Grundform nicht ändert, konnte das Routing optimal daran angepasst werden. Ein Knoten benötigt im Overlay nur wenig Routinginformationen um einen Weg zu einem beliebigen anderen Knoten zu finden (bei Chord und DART reichen bspw. jeweils $O(log\ n)$ Einträge für ein Netzwerk mit $n$ Knoten) und die maximale *Hop*-Anzahl einer Route ist im Overlay sehr klein (bei Chord und DART $O(log\ n)$ *Hops*). Die Overlaynetzwerke können je nach Bedarf in verschiedenen Schichten des OSI-Referenzmodells integriert werden. Dementsprechend unterscheiden unterscheiden sie sich auch in ihrem Routing.Es gibt:

- DHT-Routing in der Anwendungsschicht

[1] Die Metrik ist ein Wert für die Güte einer Verbindung, die sich aus verschiedenen Eigenschaften einer Route zusammensetzt. So können Signalstärke, *Hop*-Anzahl oder Bandbreite Faktoren der Metrik sein. Welche Faktoren einfließen hängt mit der Art des Netzwerkes zusammen. Verschiedene Metriken können in [6] nachgelesen werden.

[1] *Route Requests* sind Pakete mit denen ein Knoten den Weg zu einem anderen Knoten sucht.

[2] mit diesem Ansatz werden *Route Requests* vorerst nur mit einer begrenzten Anzahl an *Hops* vom Sender aus verschickt. Sollte dadurch keine Route gefunden werden, wird die erlaubte *Hop*-Anzahl erhöht und ein neuer *Route Request* gesendet

- DHT-basiertes Routing in der Vermittlungsschicht

Viele DHT-Protokolle sind auf die Anwendungsschicht des OSI-Modells ausgelegt. So ist es möglich sie einfach in bestehende Netzwerksysteme zu integrieren und bei Bedarf auszutauschen oder wieder zu entfernen. Es ist zu beachten, dass ein *Link* in der virtuellen Overlaystruktur mehreren *Links* auf physikalischer Ebene (siehe Abb. 2) entsprechen kann. Deswegen ist es bei der ersten Variante nötig für das DHT-Routing in der Anwendungsschicht ein zusätzliches Routingprotokoll in der Vermittlungsschicht zu integrieren, welches das Routing auf der physikalischen Ebene übernimmt.
Die zweite Variante sind DHT-basierte Routingprotokolle, wie z.B. DART ([10]) und VRR ([11]), die bereits in der Vermittlungsschicht liegen. Sie stellen aufgrund ihrer Nöhe zum physikalischen Netzwerk eine Verbindung zwischen virtueller und realer Netzstruktur anders her. Da die physikalischen Nachbarn eines Knotens im Protokoll der Vermittlungsschicht bekannt sind, können sie während des Aufbaus oder der Aktualisierung des Overlaynetzwerks mit in den Overlay-Routingtabellen der Knoten vermerkt werden. Die Stärke dieser Variante ist, dass Aufbau und Aktualisierung mit gezielten Nachrichten innerhalb des Netzwerks erfolgen können und ein Fluten im Gegensatz zu den obigen Rotuingverfahren deswegen nicht nötig ist.

# 3 DART

***D**ynamic **A**ddress **R**ou**t**ing* ist ein DHT-basiertes Routingverfahren, dass an der University of California entwickelt wurde. Es enstand aus der Motivation, das Routing skalierbarer zu machen, als es bisherige Verfahren tun. Die nächsten Abschnitte beschreiben die Struktur und Arbeitsweise von DART.

## 3.1 Struktur

Bei DART erhält jeder Knoten zusätzlich zu der schon vorhandenen Adresse, die ihn eindeutig identifiziert (im Folgendem *Identifier* genannt), z.B. die IP-Adresse, eine zweite lokale Binär-Adresse für die Lage des Knoten im Overlaynetzwerk. Bei DART wird die Overlaystruktur in Form eines Binärbaums aufgebaut (siehe Abb. 6 a). Dabei repräsentieren die Blätter aktuell vergebene Adressen, während die inneren Knoten für Gruppen mit gleichem Adresspräfix stehen. Die Länge des übereinstimmenden Präfixes sagt etwas über die Lage der Knoten zueinander aus. Alle Knoten, die aufgrund des gleichen Präfix einen virtuellen Teilbaum aufspannen, bilden ein physikalisch zusammenhängendes Teilnetz. Diese Eigenschaft wird auch als *Prefix Subgraph Constraint* bezeichnet. Je größer die Übereinstimmung der Präfixe ist, desto kleiner ist der Teilbaum und somit auch das zusammenhängende Teilnetz und desto näher liegen die Knoten beieinander. Diese Eigenschaft führt zum *Forwarding*-Verfahren von DART, das in Abschnitt 3.3.1 beschrieben wird.

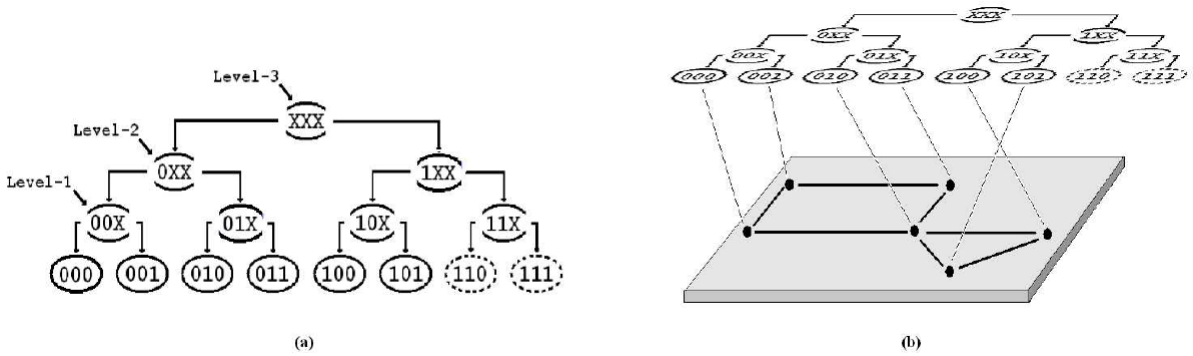



Abbildung 6: Beziehung zwischen Overlaynetzwerk und physikalischer Netzwerktopologie in DART

## 3.2 Routingtabelle

Sei im folgenden $l$ die Adresslänge der DART-Adressen. Jeder Knoten gehört dann zu einem Level-k-Teilbaum, der durch einen (l-k) langen Adresspräfix definiert ist. Level-0 Teilbäume sind dementsprechend die Knoten selbst, bzw. die Blätter des Adressbaums, ein Level-1-Teilbaum hat ein (l-1) Bit langes Präfix und kann zwei Blätter enthalten. Jeder Knoten besitzt einen sogenannten Level-k-*Sibling* für jedes k von 0 bis l-1 (in Abbildung 7 dargestellt als Dreiecke). Dabei handelt es sich um den Level-k Teilbaum, der den gleichen Vaterknoten hat wie der Level-k-Teilbaum des Knotens.

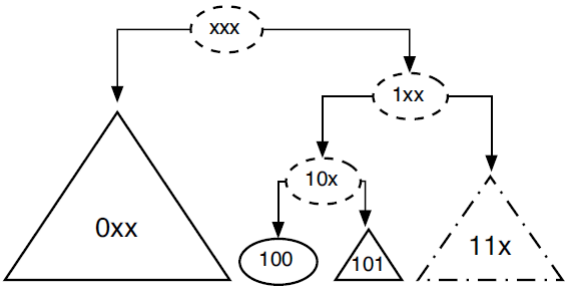



Abbildung 7: Geschwister-Teilbäume von Knoten 100

Zu jedem dieser *Siblings* wird ein Eintrag in der Routingtabelle des Knotens erstellt. Wurde aus einem der *Siblings* noch keine Adresse an einen Knoten vergeben, ist der zugehörige Eintrag leer (siehe Level-1 Eintrag in Abb. 8).
Ein Eintrag enthält die Adresse des Nachbarn, der zu den Knoten des *Siblings* führt, die zugehörigen Kosten, einem *Identifier* und einem *Routelog*. Der *Identifier* wird benutzt, damit Inkonsistenzen erkannt und behoben werden können, die durch gleichzeitige Zuweisung einer Adresse an mehrere Knoten entstehen. Der *Routelog* verhindert, dass Pakete im Kreis geschickt werden. Mehr dazu wird in [10] beschrieben.

| | nexthop | cost | route_log[] | id |
|---|---|---|---|---|
| Level 2 | 011 | 1 | 1 \| 0 \| 0 | xyz |
| Level 1 | - | - | - | - |
| Level 0 | 101 | 1 | 0 \| 0 \| 1 | abc |

Abbildung 8: Routingtabelle von Knoten 100

## 3.3 Funktionsweise

### 3.3.1 Forwarding

Möchte ein Knoten ein Paket versenden, wählt er unter seinen Einträgen denjenigen aus, dessen zugehöriger *Sibling* den Zielknoten enthält. Dies entspricht dem Eintrag/*Sibling* von Level ((Adresslänge-1)-Präfixlänge). Durch das *Prefix Subgraph Constraint* ist gesichert, dass das Paket in ein immer kleiner werdendes zusammenhängendes Teilnetz weitergeleitet wird, in dem sich der Zielknoten befindet und letztlich dort angelangt.
Ein Beispiel: Knoten [100] aus Abbildung 6 möchte ein Paket zu Knoten [001] senden. Der übereinstimmende Präfix ist null Bit lang. Dementsprechend hat der Level-((3-1)-0)-*Sibling* [0XX] die größte Übereinstimmung mit [001] also wird das Paket an Knoten [011] weitergeleitet (entsprechend der Routingtabelle aus Abb. 8). [011] hat [001] als Nachbarn und sendet das Paket direkt dorthin.

### 3.3.2 Adressauflösung

Ein Knoten kennt von anderen Knoten nur den *Identifier*. Um aber ein Paket an einen Knoten schicken zu können muss der Sender die DART-Adresse des Zielknotens kennen. Damit jeder Teilnehmer feststellen kann, welcher Knoten welche lokale Adresse besitzt, wird eine DHT verwendet. Eine global bekannte Hashfunktion liefert zu jedem *Identifier* $x$ die DART-Adresse des sog. Ankerknotens von $x$. Das ist der Knoten der die gewünschte DART-Adresse des Zielknotens $x$ kennen soll (Abb. 9). Sollte die Adresse nicht belegt sein, ist der Knoten für die Adresszuordnung von $x$ verantwortlich, dessen Adresse dem Hashergebnis am ähnlichsten ist. Die Wahl wird durch eine XOR-Verknüpfung von Hashergebnis und Adresse bewertet, wobei das kleinste Ergebnis gewinnt. Da ein Knoten kein globales Wissen hat, vergleicht er nur die XOR-Ergebnisse seiner Routingeinträge und sendet seine Anfrage zu dem eingetragenen nächsten *Hop*, der den größten Wert liefert. Folgendes Beispiel bezüglich Abbildung 6 soll das Vorgehen verdeutlichen: Der Knoten [100] möchte ein Paket an einen bestimmten Knoten senden und erhält von der Hashfunktion [111] als zuständige Adresse. Da [111] jedoch nicht belegt ist, vergleicht [100] die nächsten *Hops* [011] und [101] seiner Routingtabelle (siehe Abb. 8). Durch XOR-Verknüpfung erhält man für [011] den Wert $011_2$ *xor* $111_2 = 100_2 = 4_{10}$ , für [101] den Wert $101_2$ *xor* $111_2 = 010_2 = 2_{10}$ und für [100] selbst $100_2$ *xor* $111_2 = 011_2 = 3_{10}$. Damit liefert [101] den kleinsten Wert und die Adressanfrage wird dorthin weitergeleitet. [101] hat Routingeinträge mit den Adressen [100] und [001] als nächsten *Hop*. Knoten

[001] hat den Wert $001_2$ *xor* $111_2 = 110_2 = 6_{10}$, sodass [101] der Knoten mit dem kleinsten Wert bleibt und somit für die gesuchte Adresszuordnung kennen muss.

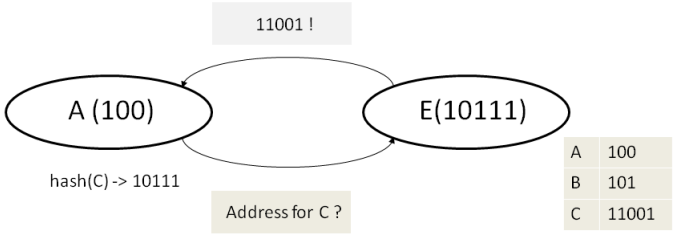



Abbildung 9: Adressauflösung bei Paketsendung von A nach C

### 3.3.3 Adressvergabe

DART ist ein proaktives Routingverfahren, Knoten senden also periodisch Pakete mit ihren aktuellen Routingtabellen (sog. *Routingupdates*) an ihre Nachbarn. Wenn ein Knoten $n$ eine neue Adresse benötigt, weil er dem Netzwerk beitritt oder seine Position verändert, kann er aus den *Routingupdates* eine Adresse bestimmen. Dazu wählt er aus allen empfangenen Routingtabellen den leeren Eintrag, der dem höchsten Level-k-*Sibling* eines Nachbarn zugeordnet ist. Anders ausgedrückt wird eine Adresse aus dem größten freien Adressraum gewählt, den die Nachbarn des Knoten kennen. Der neue Knoten baut dann seine Routingtabelle auf, sendet ein *Routingupdate* an alle Nachbarn und macht sich damit dem Netzwerk bekannt. Dabei muss dieses *Routingupdate* einerseits an die Knoten des Level-(k+1)-Teilbaums gesendet werden, denn nur sie können einen leeren Eintrag/*Sibling* haben. Andererseits muss die DHT in dem Ankerknoten von $n$ aktualisiert werden. Nach dem *Prefix Subgraph Constraint* sind *Routingpdates* also abgesehen vom Update des Ankerknotens auf Teilnetze beschränkt. Zur Verdeutlichung der Adressvergabe betrachte man folgendes Beispiel aus Abbildung 10.

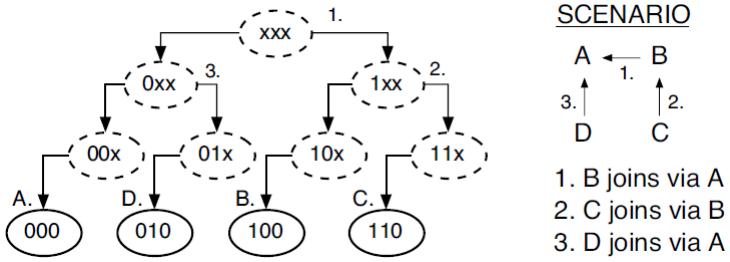



Abbildung 10: Adressvergabe in einem kleinen Netzwerk

Knoten A erhält als erster Knoten die Adresse [000], alle Einträge seiner Routingtabelle sind leer. Knoten B tritt dem Netzwerk über A bei und wählt Adresse [100] aus dem Level-2-*Sibling* von A. A aktualisiert dementsprechend seinen Level-2-Eintrag in der

Routingtabelle. Knoten C tritt über B bei. B's Level-1-*Sibling* ist der größte Adressbereich den B kennt. C wählt daraus die Adresse [110]. B aktualisiert seine Routingtabelle. A muss dies nicht tun, denn dort existiert schon ein Eintrag für seinen Level-2-*Sibling*. Schließlich tritt D über A bei und wählt [010] aus dem leeren Level-1-*Sibling* von A. Nur A muss seine Routingtabelle aktualisieren.

# 4 BRN-DART

BRN-DART ist eine Implementierungsvariante von DART, die an der Humboldt Universität Berlin im Rahmen des BerlinRoofNet-Projektes ([12]) entwickelt wurde. Sie bildet die Grundlage für die Untersuchung möglicher Optimierungen, die in Kapitel 5 betrachtet werden. Um diese nachvollziehen zu können sollen folgend einige Unterschiede zu DART verdeutlicht werden.

## 4.1 Routingtabelle

Die Einträge der Routingtabelle stellen keinen Bezug zu einem Level-k-*Sibling* her. Stattdessen werden nur die Adressen der physikalischen Nachbarn gespeichert. Die Anpassung hat zwei Vorteile: Besonders in großen Netzwerken ist die Anzahl der Nachbarn meist kleiner als die der *Siblings*. Es müssen dann im Vergleich zu DART weniger Routinginformationen gespeichert werden. Außerdem beschränken sich Netzwerkupdates auf Nachbarn. In DART können sie durch das gesamte Netzwerk verlaufen.

## 4.2 Adressvergabe

Beitretende Knoten erhalten von ihren Nachbarn genau wie in DART eine Adresse aus dem größten freien Adressraum, den diese kennen. Die Umsetzung ist aber eine andere. Unter allen Nachbarn kennt der Knoten den größten freien Adressraum, dessen Adresse am kürzesten ist. Dieser Knoten erweitert seine eigene Adresse um eine Null am Ende, der hinzugekommene Knoten erhält die gleiche Adresse nur mit einer Eins anstatt einer Null. Damit ist der freie Adressraum gleichmäßig auf die beiden Knoten aufgeteilt worden. Durch dieses Prinzip lassen sich Adresszuteilungen sehr einfach bewältigen. Im Gegensatz zu DART muss die Routingtabelle nicht betrachtet werden. Wendet man das Prinzip auf Abbildung 10 an, erhält man nachstehende Adresszuweisungen:
Knoten A braucht als erster Knoten keine Adresse, da noch keine Kommunikation möglich ist. Knoten B tritt dem Netzwerk über A bei. A erhält Adresse [0] und B erhält [1]. Knoten C tritt über B bei. C erweitert seine Adresse um eine Null zu [10], C tauscht die 0 aus und erhält somit [11]. Schließlich tritt D über A bei und erhält nach dem gleichen Prinzip [101]. Die Adresse von A wird zu [100] erweitert.

## 4.3 Forwarding

In BRN-DART wählt ein Knoten unter seinen Nachbarn denjenigen aus, der ihn entlang der Kanten im virtuellen Adressbaum zum Zielknoten führt. Das ist primär derjenige,

dessen Adresspräfix bis zur letzten 1 in der Zieladresse vorkommt. Gibt es mehrere solcher Knoten, wird das Paket an denjenigen mit dem längsten übereinstimmenden Präfix weitergeleitet. Sollte keiner der Nachbarn das Kriterium erfüllen, wird der Nachbar, welcher das kürzeste Präfix bis zur letzten 1 hat, gewählt. Diese Art der Weiterleitung bewirkt, dass das Paket solange Richtung Wurzel geschickt wird bis es sich in einem Teilbaum des Zielknotens befindet. Von dort geht es wieder in Richtung der Blätter. Viele der Routen unterscheiden sich gegenüber DART, wobei sowohl kürzere als auch längere möglich sind (vgl. Abb 11 und 12). Kürzere Routen begründen sich auf die Routingtabelle von DART, die nur einen Nachbar pro *Sibling* einträgt, wohingegen BRN-DART alle Nachbarn, die im *Sibling* liegen berücksichtigt. Marcello Caleffi und Luigi Paura haben dieses Problem von DART ebenfalls erkannt und in [**?**] eine erweiternde DART-Version namens M-DART vorgestellt. Längere Routen entstehen, wenn in der Routingtabelle von DART ein Eintrag gewählt wird, dessen nächster *Hop* ein Präfix hat, das nicht bis zur letzten 1 mit dem Präfix des Zielknotens übereinstimmt. BRN-DART würde diesen nächsten *Hop* nicht wählen, sondern - sofern kein anderer Nachbar aus dem *Sibling* bekannt ist - einen Knoten wählen, der ihn in Richtung der Wurzel des Binärbaums führt. Im Gegensatz zu DART wird so aber der Abstand zum Zielknoten länger. Eine Lösung dafür wird in dieser Arbeit vorgestellt.

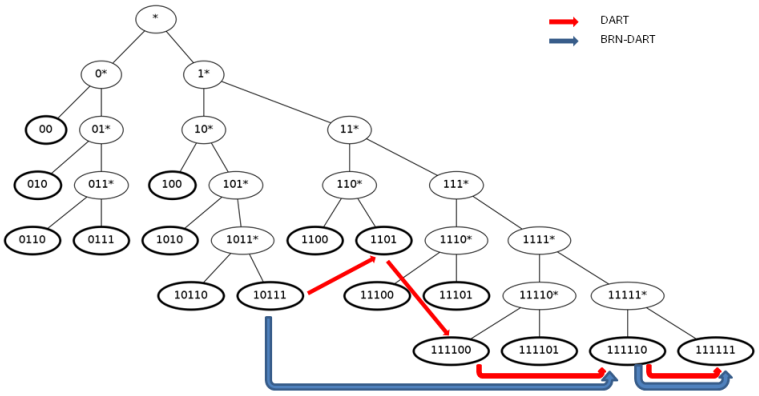



Abbildung 11: *Forwarding* im Binärbaum

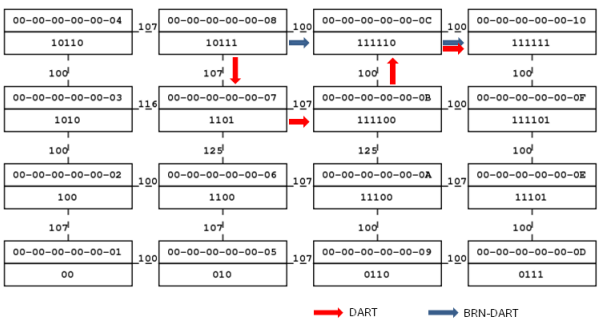



Abbildung 12: *Forwarding* im Netzwerk

# 5 Evaluation

In diesem Kapitel wird gezeigt, dass BRN-DART Möglichkeiten bietet Routen zwischen Sender und Empfänger im Vergleich zur Standard-Implementierung zu verkürzen. Dazu wird zuerst eine Verbesserungsstrategie des *Forwarding*-Algorithmus vorgestellt, die einen anderen Präfixvergleich verwendet als der Standard. Dann wird eine Strategie präsentiert, die durch Einbeziehen von Nachbarwissen die Weglänge weiter verkürzt. Es wird zudem eine Kombination der beiden Strategien betrachtet. In der Simulation werden die durch die Verbesserungen entstehenden Routen sowohl mit den Routen der Standard-Implementierung verglichen, als auch mit denen, die durch globales Wissen der Netzwerktopologie als optimale Routen bestimmt wurden. Die Beurteilung der Strategien findet anhand zweier Metriken statt, der *Expected Transmission Count* (ETX)- und der *Hop Coun*t-Metrik. Der Wert der ETX-Metrik richtet sich nach der Wahrscheinlichkeit, dass ein Paket erfolgreich zwischen zwei Knoten A und B übertragen wird. Dazu senden sich die Knoten gegenseitig *Linkprobes* und ermitteln die Übertragungswahrscheinlichkeit von A nach B (PSR_AB) und von B nach A (PSR_BA). Die ETX-Metrik des *Links* ist dann $\frac{100}{PSR\_AB \cdot PSR\_BA}$.
Die *Hop Count*-Metrik entspricht der Anzahl der *Hops*, die für den Weg benötigt wurden. Sie soll zusätzlich zur ETX-Metrik betrachtet werden, weil die ETX-Metrik von DART nicht bei der Wegewahl beachtet wird. Durch den Präfixvergleich wählt ein Knoten beim *Forwarding* für denselben Empfänger immer denselben Nachbarn, auch wenn er andere Knoten kennt, deren Verbindung eine bessere ETX-Metrik haben. Da die ETX-Metrik aber durch ihre Aussage über die Übertragungswahrscheinlichkeit ein wichtiges Kriterium für ein erfolgreiches Senden ist, soll sie in der Auswertung mit betrachtet werden.

## 5.1 Methodik

### 5.1.1 Verbesserungsstrategien

Die nächsten beiden Abschnitte beschreiben zwei Strategien, die BRN-DART verbessern sollen. Die erste soll versuchen vorhandenes Wissen besser auszunutzen, die zweite soll zusätzliches Wissen miteinbeziehen. Die Kombination beider Strategien ist ohne zusätzliche Anpassungen möglich und wird deswegen nicht weiter erläutert.

#### 5.1.1.1 Präfixvergleich

Wie schon in Abschnitt 4.3 angedeutet, hat das *Forwarding*-Konzept Optimierungspotenzial in der Wahl des nächsten *Hop*. Beim Weiterleiten wird ein Paket zuerst Richtung Wurzel geschickt, bis es den Knoten erreicht, der nach dem Prinzip der Address Allocation einen Knoten kennen muss, der sich im selben *Sibling* wie der Zielknoten befindet. Das garantiert zwar, dass eine Route zum Ziel gefunden wird, der Weg zu diesem Knoten bedeutet jedoch meist einen Umweg. Knoten, die sich auf dem Weg zu ihm befinden, können bereits Nachbarn haben, die ebenfalsls zu dem *Sibling* gehören. Das *Forwarding* ignoriert sie aber, obwohl sie schneller zum Zielknoten führen würden. Die Verbesse-

rungsstrategie soll diese Nachbarn berücksichtigen. Alter und neuer Algorithmus sehen wie folgt aus:

**Algorithmus 1** Berechne besten Nachbarn zum Ziel (Standard-Version)

```
präfix_best = '';
ForEach Nachbar
    präfix_nachbar = präfix_bis_zur_letzten_1(Nachbar)
    if ((präfix_nachbar ist Präfix von Ziel) AND
        (Länge(präfix_nachbar) > Länge(präfix_best))) then
        bester_knoten = Nachbar;
        präfix_best = präfix_nachbar;
    end if
end for
if (bester_knoten ist leer) then
    ForEach Nachbar
        präfix_nachbar = präfix_bis_zur_letzten_1(Nachbar)
        if (bester_knoten ist leer) OR
          (Länge(präfix_nachbar)) < Länge(präfix_best)) then
            bester_knoten = Nachbar;
            präfix_best = präfix_nachbar;
        end if
    end for
end if
return bester_knoten;
```

**Algorithmus 2** Berechne besten Nachbarn zum Ziel (optimierte Version)

```
präfix_best = '';
ForEach Nachbar
    präfix = längstes_gemeinsames_präfix(Nachbar, Ziel);
    if (Länge(präfix) > Länge(präfix_best)) then
        bester_knoten = Nachbar;
        präfix_best = präfix;
    else if (Länge(präfix) == Länge(präfix_best)) then
        if (präfix_bis_zur_letzten_1(Nachbar) < präfix_bis_zur_letzten_1(bester_knoten)) then
            bester_knoten = Nachbar;
            präfix_best = präfix;
        end if
    end if
end for
return bester_knoten;
```

#### 5.1.1.2 Erweitertes Nachbarwissen

In BRN-DART kennt jeder Knoten seine Nachbarn. Dies ist Voraussetzung um ein funktionierendes Kommunikationsnetz aufbauen zu können. Um sich bekannt zu machen, muss ein Knoten mit seiner Umgebung kommunizieren. Das in BRN-DART benutzte Verfahren heißt *Linkprobing* ([20]): Die Knoten tauschen mit ihren Nachbarn Pakete aus

um sich gegenseitig bekannt zu machen. Dabei verwenden sie verschiedene Paketgrößen um die Qualität ihrer Verbindungen festzustellen. Insbesondere die größeren Pakete bieten genügend unbenutzten Platz, den die Optimierungsstrategie nutzen kann um ihn mit den Routinginformationen eines Knotens anzureichern. Dadurch kann erreicht werden, dass jeder Knoten die Nachbarn seiner Nachbarn kennt (2-*Hop*-Nachbarschaft). Mit diesem Wissen können dann kürzere Wege entdeckt werden. So kann in Abbildung 13 dank der Strategie ein Weg mit sechs *Hops* auf vier *Hops* reduziert werden, da Knoten *0B* durch sein erweitertes Wissen nun einen kürzeren Weg zum Knoten *03* kennt.

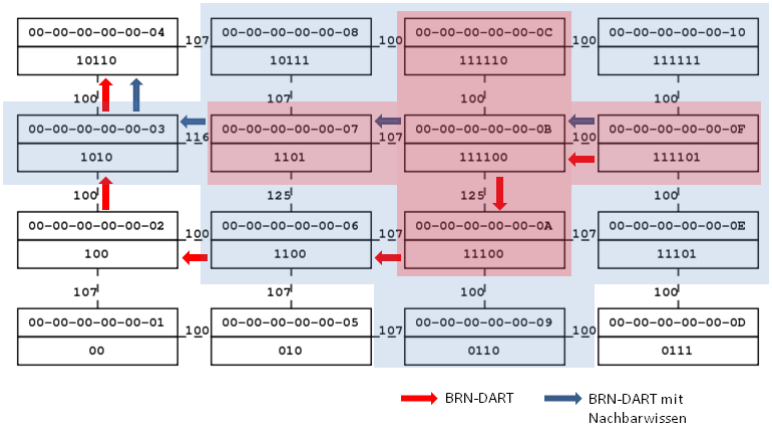



Abbildung 13: *Forwarding* mit erweitertem Nachbarwissen:
Rot - Standardwissen von *0B*, Blau - erweitertes Wissen von *0B*

### 5.1.2 Testumgebung

Ziel der Simulation war es die Wegewahl in BRN-DART mit verschiedenen Anpassungen zu analysieren und sie mit der Standard-Wegewahl von BRN-DART zu vergleichen. Dafür wurde der NS2-Simulator benutzt, ein Programm zur Simulation von paketvermittelten Computernetzen. Die Netzwerkknoten wurden als möglichst quadratische Gitter angeordnet (sog. Grid-Topologie, siehe Abb. 13). Es sind Netzwerke mit 25, 50 und 100 Knoten untersucht worden. Mit NS2 wurde für jedes Netzwerk eine Momentaufnahme gemacht. Genauer wurden die simulierten Routingtabellen und Netzwerkverbindungen in eine XML-Datei exportiert und an ein C++-Programm übergeben, welches anhand dieser Daten die verschiedenen Strategien nachgestellt hat. Es wurden zu jedem Netzwerk die Kosten aller möglichen Routen ermittelt. Dabei hat das Programm für jede Route die ETX-Metrik und die *Hop Count*-Metrik errechnet. Für die Kostenberechnung der optimalen Route zwischen zwei Knoten wurde der Dijkstra-Algorithmus benutzt.

## 5.2 Ergebnisse und Diskussion

Die Diagramme in Abbildung 14, 15 und 16 zeigen *Hop Count*- und ETX-Metriken der durch die Strategien gewählten Routen. Die Verbesserungsstrategie durch Präfixvergleich

ist mit S1, die duch Nachbarwissen mit S2 und deren kombinierte Version mit S3 benannt.

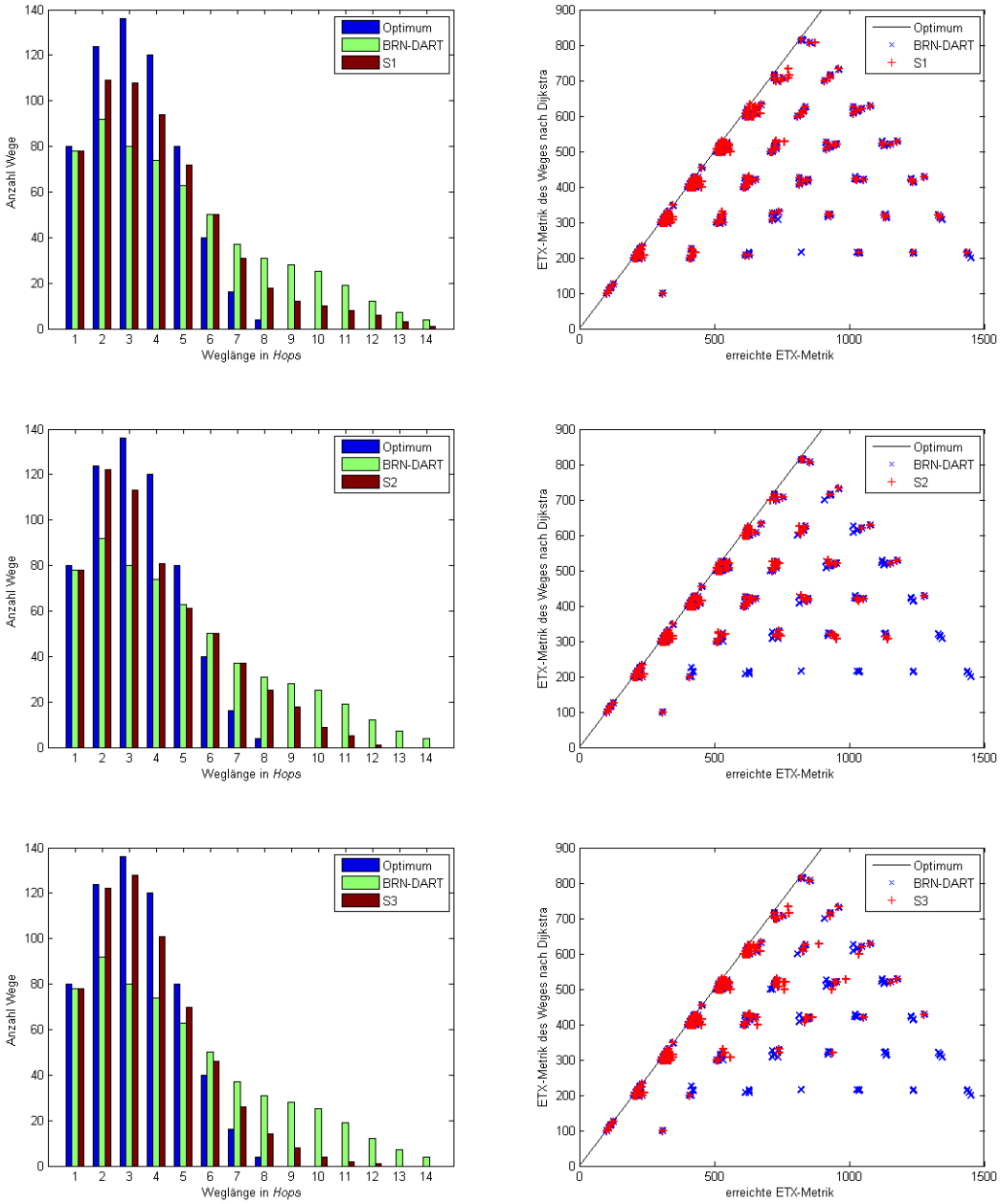



Abbildung 14: *Hop Count*- und ETX-Metriken aller Routen im Netzwerk (25 Knoten)

In allen Abbildungen ist eine deutliche Verbesserung sowohl in der *Hop Count*-Metrik, als auch in der ETX-Metrik der Routen zu erkennen. Dabei erzielt Strategie S3 die größten Verbesserungen. Insbesondere an den *Hop Count*-Diagrammen ist zu erkennen,

dass sich durch die Strategien eine starke Annäherung an das Optimum erzielen lässt. Die Verbesserungen sind anteilig aller Routen mit wachsender Netzwerkgröße zwar geringer (Abb. 16/ S1 und S2) aber trotzdem vorhanden und bleiben vor allem in der kombinierten Variante sehr deutlich (Abb. 16/ S3).

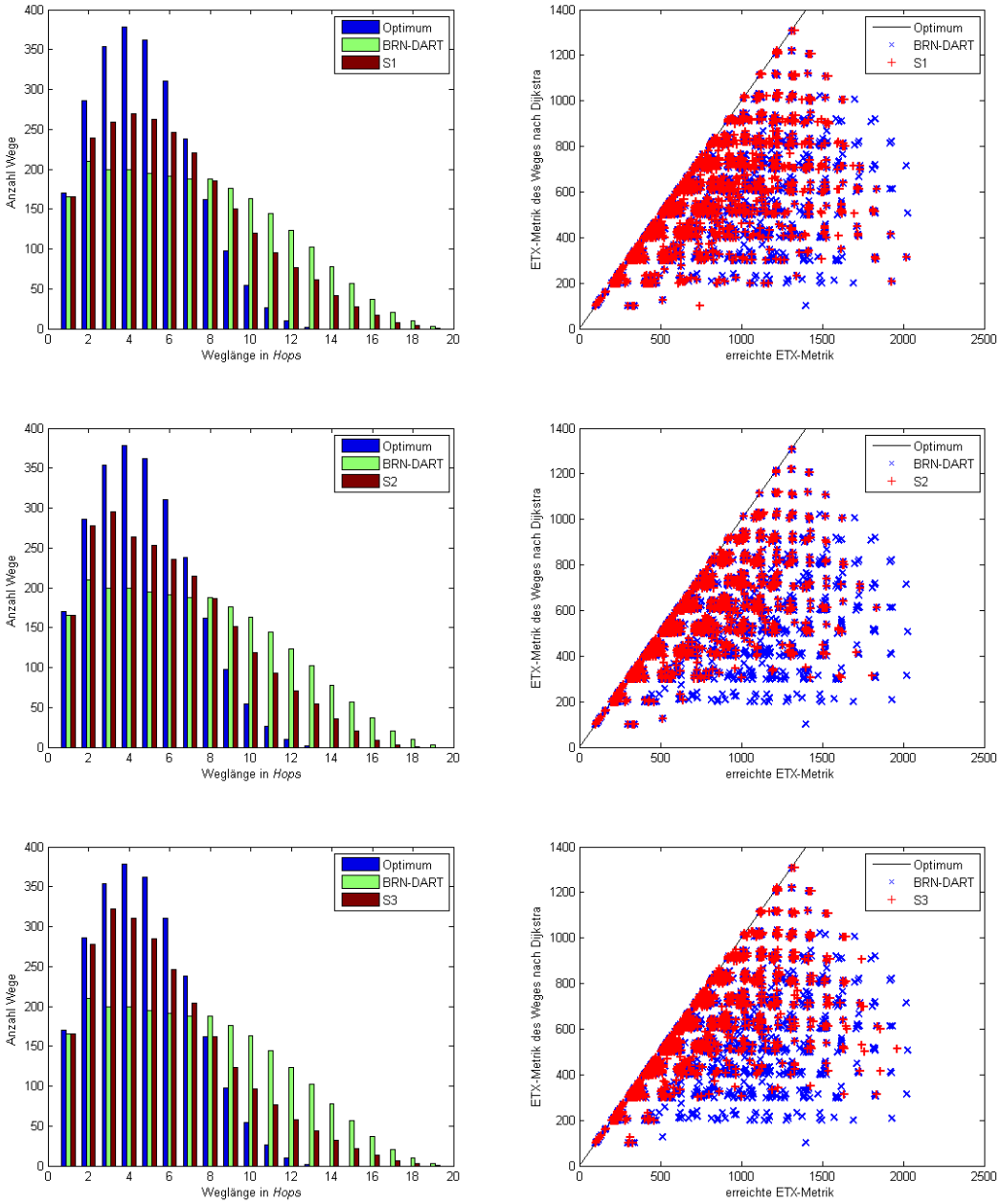



Abbildung 15: *Hop Count*- und ETX-Metriken aller Routen im Netzwerk (50 Knoten)

Die ETX-Metrikdiagramme zeigen, dass Verbesserungen hauptsächlich dort stattfin-

den, wo die größten Differenzen zwischen Optimum und BRN-DART herrschen. Vorrangig sind das die Wege, die im Optimum sehr kurz sind. So wurden vor allem Wege stark verbessert, die eine ETX-Metrik von bis zu 400 im Optimum haben. Dabei wurden sogar bis zu 14-fach größere Werte auf des Optimum oder zumindestens das doppelte Optimum reduziert (Bsp. in Abb. 16/ S3 - Optimum 200 : von 3400 auf maximal 400).

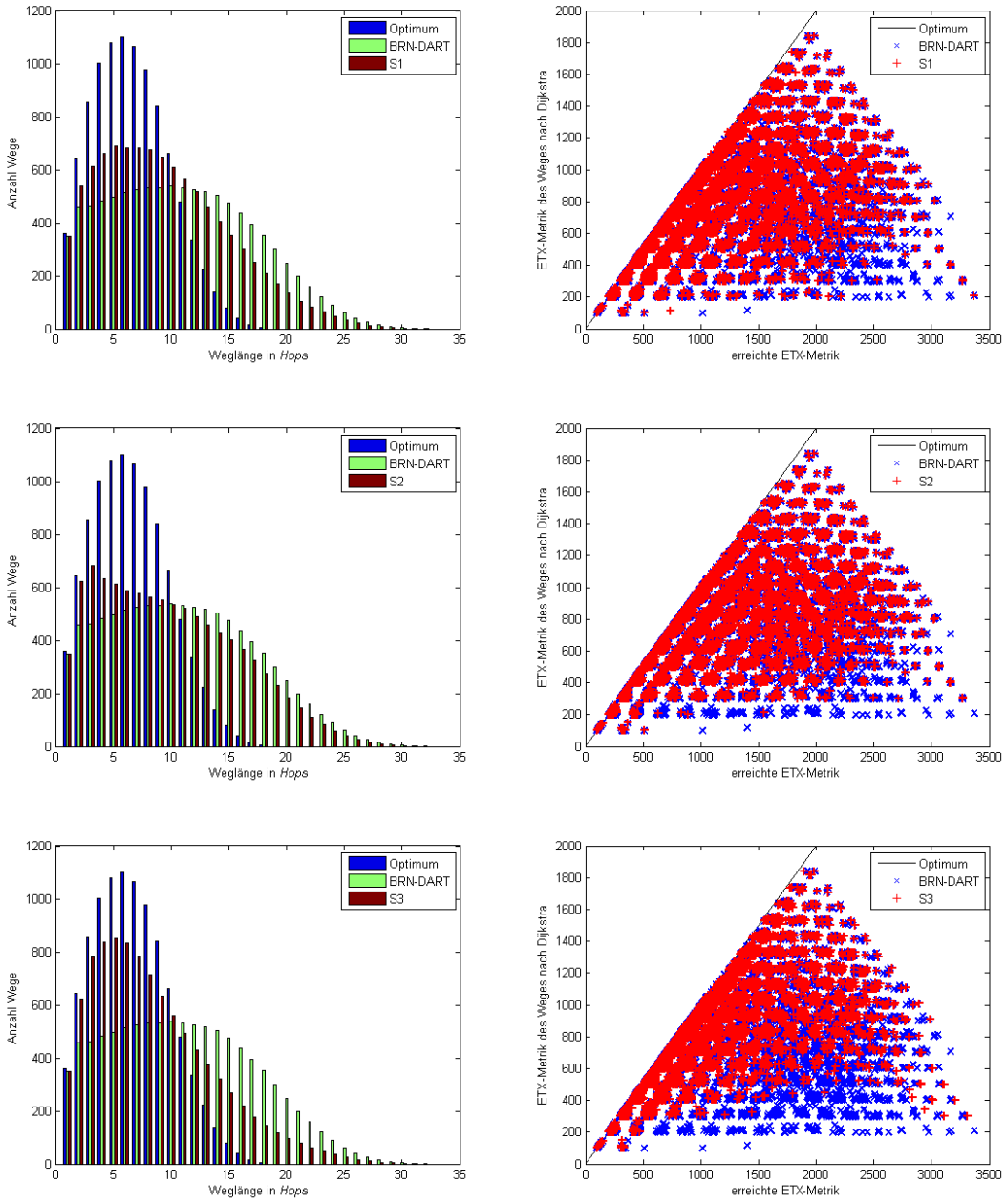



Abbildung 16: *Hop Count*- und ETX-Metriken aller Routen im Netzwerk (100 Knoten)

Genaue Ergebnisse sind in den Tabellen 1-3 im Anhang ablesbar. Dort kann man bezüglich der *Hop Count*-Metrik folgende Erkenntnisse gewinnen.
Aus den Spalten 4 bis 6 ist ersichtlich, dass für fast alle Weglängen Verbesserungen erzielt werden konnten. Ausnahme bilden hier nur Wege der Länge 3. Diese fallen jedoch aufgrund ihre minimalen Optimierbarkeit und ihres geringen Vorkommens kaum ins Gewicht. Weiterhin ist zu erkennen, dass mit steigender Weglänge der Anteil verbesserter Wege steigt. Das bedeutet je länger ein optimierbarer Weg ist, desto wahrscheinlicher greifen die Verbesserungsstrategien. Dies ist eine hervorragende Eigenschaft, denn mit wachsender Länge einer Strecke wird deren Fehleranfälligkeit erhöht, besonders in mobilen Netzwerken. Die Strategien können diese Fehleranfälligkeit durch das Verkürzen der Wege vermindern.
Von den drei Strategien war S3 diejenige, die für jede Weglänge die meisten Verkürzungen erzielte. S1 hatte im Netzwerk mit 100 Knoten bessere Werte als S2. In den anderen beiden Netzwerken war dies umgekehrt.
Für alle drei Strategien gilt, dass mit größer werdenden Netzwerken und mit dementsprechend wachsender Anzahl optimierbarer Wege, der proz. Anteil an verbesserten Wegen pro Weglänge sinkt. So lag der Anteil im kleinsten untersuchten Netzwerk teilweise bei 80-100% (siehe Weglängen 10-14), im größten waren es maximal 60%. Trotzdem ist der Wert immer noch hoch.
Die *Hop*-Anzahl, die bei den verbesserten Wegen gespart werden konnten, wurden anteilig an den maximal möglichen Einsparungen mithilfe der Werte aus Spalte 7-12 berechnet. Die Werte lagen hauptsächlich zwischen 70% und 90%, selten unter 60%.
In der durchschnittlichen *Hop*-Verkürzung pro Weglänge unterschieden sich die Strategien um maximal 2 *Hops* bei Wegen unter 20 Hops, um 2-4 *Hops* bei Wegen mit 20 bis 32 *Hops*.
Betrachtet man alle Wege zusammen, hat Strategie S1 in dem kleinsten Netzwerk 52,49% der unnötigen *Hops* gespart, im mittleren Netzwerk 43,15% und im größten 37,89%. Für Strategie S2 liegen die Werte bei 60,09%, 54,12% und 27,61%. Die Kombination beider Strategien ergab ein Ersparnis von 80,91%, 63,72% bzw. 60,34%. Man erkennt, dass mit steigender Netzwerkgröße der Anteil der Einsparungen gemessen an der Anzahl überflüssiger *Hops* abfällt. Die Einsparungen sind aber dennoch beachtenswert, da insbesondere S3 die unnötigen *Hops* um mehr als die Hälfte reduziert. Betrachtet man die ETX-Metrik, ist ein ähnliches Verhältnis bei den Strategien zu finden. So steigt z.B. wie bei der *Hop Count*-Metrik mit der Länge des Weges die Wahrscheinlichkeit einer Verbesserung. Dennoch sind kleine Unterschiede zu den *Hop*-Messungen zu erkennen, die aber dadurch zu erklären sind, dass die Strategien ebenso wie BRN-DART selbst die ETX-Metrik in der Wegewahl nicht miteinbeziehen. Dementsprechend kann es auch zur Erhöhung der ETX-Metrix kommen, bspw. wenn einzelne Verbindungen in einem kürzeren und dadurch vermeintlich besseren Weg ein schwächeres Signal oder Interferenzen besitzen. In den Tabellen 4-6 äußert sich dieser Umstand wie folgt. Sie zeigen zunächst, dass die Wahrscheinlichkeit für eine bessere ETX-Metrik etwas höher ist als für eine bessere *Hop*-Anzahl (Spalte 4-6). Wie aber in Tabelle 7 zu sehen ist, gibt es genauso die Möglichkeit, dass sich die ETX-Metrik eines Weges erhöht. Gleiches lässt sich

zum Grad der Verbesserung pro Weglänge sagen (Spalte 7-15). Auch hier sind die Werte sehr ähnlich zu den Messungen der *Hop Count*-Metrik. Rechnet man die durchschnittlichen Metrikwerte in *Hops* um (ein *Hop* enspricht hier einer ETX-Metrik von 100-170), so erkennt man dies sehr deutlich. An diesen Messungen wird auch die Möglichkeit der Verschlechterung sichtbar. So sind trotz der Ähnlichkeit zu den *Hop*-Messungen der Abstand der erreichten ETX-Metriken zum Optimum etwas größer als der Abstand der *Hops*. Genauer gesagt unterscheiden sich die Messungen für jede Weglänge um ca. 2%. Letztendlich äußert sich dieser Unterschied auch in den Gesamtwerten. So konnten mit S1 49,40% im kleinen, 38,94% im mittleren bzw. 35,32% im großen Netzwerk der durch Umwege entstandenen Metriken eingespart werden. Mit S2 waren es 56,97%, 50,54% und 25,85% und mit S3 75,51%, 55,00% und 60,01%. Trotz der kleinen Unterschiede ist insgesamt erkennbar, dass auch die ETX-Metriken durch die Strategien stark verbessert werden konnten.

# 6 Fazit und Ausblick

Im letzten Abschnitt konnte gezeigt werden, dass die Implementierung BRN-DART des DHT-basierten Routingprotokolls DART in seiner Wegewahl optimiert werden kann. Insbesondere die Anzahl der *Hops* kann erheblich reduziert werden. Alle vorgestellten Strategien konnten deutliche Verbesserungen erzielen. Die Kombination aus S1 und S2 war dabei am Erfolgreichsten. Mithilfe dieser Optimierungsstrategie konnten Einsparungen unnötiger Kosten von über 50% erreicht werden. Diese Einsparungen verteilten sich gleichmäßig auf über 50% der optimierbaren Routen.
Betrachtet man nun noch die Kosten, die nötig sind um die Strategie zu integrieren, scheint die Benutzung ein Muss zu sein. So benötigt S1 gar keine Ressourcen und für S2 müssen lediglich die für das Routing benutzten *Linkprobe*-Pakete um die Routingeinträge eines Knoten erweitert werden. Da sich die Anzahl der Routingeinträge auf Nachbarn beschränkt und somit in $O(1)$ liegt , ist diese Erweiterung minimal.
Für die Zukunft ist die Integration in ein reales Maschennetzwerk geplant, durch deren Untersuchung die Simulationsergebnisse für eine reale Testumgebung bestätigt werden sollen.
Außerdem bietet DART noch weitere Optimierungsmöglichkeiten bezüglich des *Forwardings*, die untersucht werden sollen. So kann die Strategie, welche Nachbarwissen benutzt, erweitert werden, indem der Radius der Knoten, von denen zusätzliches Wissen bezogen wird, vergrößert werden. In drahtlosen Maschennetzwerken ist die maximale Grenze besonders interessant, da durch die Mobilität der Knoten das Wissen umso schneller veraltet je weiter die Knoten voneinander entfernt sind und je mehr Bewegungen im Netzwerk auftreten. Desweiteren sollen Strategien untersucht werden, die die ETX-Metrik berücksichtigen.

# A Messergebnisse

Folgende Werte sind in den Spalten der Tabellen 1-3 zu finden:

**Spalte 1:** Hopanzahl der Routen in der Standardimplementierung.

**Spalte 2:** Anzahl der Routen , die die Weglänge aus Spalte 1 haben.

**Spalte 3:** Anzahl der Routen aus Spalte 2, die nach dem Dijkstra-Algorithmus kürzer sein könnten.

**Spalte 4-6:** proz. Anteil der Routen aus Spalte 3, die durch die jeweilige Strategie verbessert werden konnten.

**Spalte 7-9:** durchschnittlicher optimaler *Hop Count* (nach Dijkstra), der für die Routen aus Spalte 4-6 erzielt werden kann.

**Spalte 10-12:** durchschnittlicher *Hop Count*, der für die Routen aus Spalte 4-6 durch die jeweilige Strategie erzielt worden ist.

| Weglänge | Anzahl Wege | optimierbar | verbessert in % | | | Ø optimale Hopzahl | | | Ø erreichte Hopzahl | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 78 | 0 | 0 | 0 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 2 | 92 | 0 | 0 | 0 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 3 | 80 | 2 | 0 | 0 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 4 | 74 | 14 | 8 | 16 | 16 | 2 | 2 | 2 | 2 | 2 | 2 |
| 5 | 63 | 24 | 16 | 25 | 32 | 3 | 3 | 3 | 3 | 3 | 3 |
| 6 | 50 | 29 | 24 | 30 | 44 | 3 | 3 | 3 | 4 | 3 | 3 |
| 7 | 37 | 27 | 32 | 43 | 59 | 4 | 4 | 4 | 4 | 4 | 4 |
| 8 | 31 | 27 | 45 | 58 | 68 | 4 | 4 | 4 | 5 | 5 | 5 |
| 9 | 28 | 28 | 61 | 71 | 86 | 5 | 4 | 5 | 5 | 6 | 5 |
| 10 | 25 | 25 | 60 | 80 | 88 | 4 | 4 | 4 | 5 | 6 | 5 |
| 11 | 19 | 19 | 58 | 84 | 89 | 4 | 4 | 4 | 4 | 7 | 5 |
| 12 | 12 | 12 | 50 | 92 | 92 | 4 | 4 | 4 | 4 | 7 | 5 |
| 13 | 7 | 7 | 57 | 100 | 100 | 3 | 3 | 3 | 4 | 9 | 4 |
| 14 | 4 | 4 | 75 | 100 | 100 | 2 | 2 | 2 | 2 | 2 | 2 |

Tabelle 1: *Hop*-Messergebnisse zum Netzwerk mit 25 Knoten

| Weglänge | Anzahl Wege | optimierbar | verbessert in % | | | Ø optimale Hopzahl | | | Ø erreichte Hopzahl | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 165 | 0 | 0 | 0 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 2 | 210 | 0 | 0 | 0 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 3 | 199 | 3 | 0 | 0 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 4 | 200 | 29 | 6 | 12 | 12 | 2 | 2 | 2 | 2 | 2 | 2 |
| 5 | 195 | 53 | 10 | 16 | 19 | 3 | 3 | 3 | 3 | 3 | 3 |
| 6 | 191 | 77 | 15 | 22 | 30 | 3 | 3 | 4 | 4 | 3 | 4 |
| 7 | 188 | 99 | 20 | 27 | 38 | 4 | 4 | 4 | 4 | 4 | 4 |
| 8 | 188 | 125 | 28 | 37 | 48 | 5 | 4 | 5 | 5 | 5 | 5 |
| 9 | 176 | 134 | 35 | 47 | 57 | 5 | 5 | 5 | 5 | 6 | 5 |
| 10 | 163 | 139 | 41 | 57 | 66 | 5 | 5 | 5 | 5 | 6 | 6 |
| 11 | 144 | 131 | 47 | 65 | 71 | 5 | 5 | 6 | 6 | 7 | 6 |
| 12 | 123 | 117 | 50 | 69 | 75 | 5 | 5 | 6 | 6 | 7 | 7 |
| 13 | 102 | 100 | 52 | 73 | 76 | 5 | 6 | 6 | 7 | 8 | 7 |
| 14 | 78 | 78 | 54 | 74 | 78 | 6 | 6 | 6 | 8 | 9 | 8 |
| 15 | 57 | 57 | 54 | 79 | 81 | 6 | 6 | 6 | 9 | 10 | 9 |
| 16 | 37 | 37 | 54 | 81 | 84 | 6 | 6 | 6 | 11 | 11 | 10 |
| 17 | 21 | 21 | 62 | 90 | 90 | 6 | 6 | 6 | 13 | 13 | 11 |
| 18 | 10 | 10 | 60 | 90 | 90 | 6 | 5 | 5 | 14 | 13 | 11 |
| 19 | 3 | 3 | 67 | 100 | 100 | 6 | 5 | 5 | 15 | 16 | 12 |

Tabelle 2: *Hop*-Messergebnisse zum Netzwerk mit 50 Knoten

| Weglänge | Anzahl Wege | optimierbar | verbessert in % | | | Ø optimale Hopzahl | | | Ø erreichte Hopzahl | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 349 | 0 | 0 | 0 | 0 | n.a | n.a | n.a | n.a | n.a | n.a |
| 2 | 457 | 0 | 0 | 0 | 0 | n.a | n.a | n.a | n.a | n.a | n.a |
| 3 | 462 | 8 | 0 | 0 | 0 | n.a | n.a | n.a | n.a | n.a | n.a |
| 4 | 482 | 62 | 5 | 10 | 10 | 2 | 2 | 2 | 2 | 2 | 2 |
| 5 | 495 | 119 | 9 | 15 | 18 | 3 | 3 | 3 | 3 | 3 | 3 |
| 6 | 514 | 179 | 14 | 21 | 26 | 4 | 4 | 4 | 4 | 4 | 4 |
| 7 | 524 | 225 | 17 | 24 | 31 | 4 | 4 | 4 | 4 | 5 | 4 |
| 8 | 531 | 266 | 19 | 25 | 35 | 5 | 5 | 5 | 5 | 5 | 5 |
| 9 | 531 | 302 | 22 | 26 | 39 | 6 | 5 | 6 | 6 | 6 | 6 |
| 10 | 537 | 347 | 26 | 27 | 44 | 6 | 6 | 6 | 6 | 7 | 6 |
| 11 | 533 | 385 | 29 | 27 | 49 | 6 | 6 | 6 | 7 | 8 | 7 |
| 12 | 526 | 416 | 33 | 28 | 54 | 7 | 6 | 7 | 8 | 8 | 7 |
| 13 | 517 | 436 | 36 | 28 | 58 | 7 | 6 | 7 | 8 | 9 | 8 |
| 14 | 504 | 446 | 39 | 27 | 60 | 7 | 6 | 7 | 9 | 9 | 8 |
| 15 | 474 | 436 | 42 | 26 | 62 | 7 | 6 | 7 | 9 | 9 | 9 |
| 16 | 437 | 414 | 43 | 24 | 63 | 7 | 6 | 7 | 10 | 9 | 9 |
| 17 | 395 | 383 | 45 | 24 | 64 | 7 | 6 | 7 | 10 | 10 | 9 |
| 18 | 353 | 349 | 47 | 25 | 65 | 7 | 6 | 7 | 10 | 9 | 9 |
| 19 | 301 | 301 | 48 | 25 | 65 | 7 | 6 | 7 | 10 | 9 | 9 |
| 20 | 248 | 248 | 48 | 26 | 65 | 7 | 5 | 7 | 10 | 8 | 9 |
| 21 | 199 | 199 | 48 | 27 | 63 | 7 | 5 | 7 | 10 | 8 | 9 |
| 22 | 158 | 158 | 48 | 30 | 62 | 6 | 5 | 7 | 11 | 9 | 9 |
| 23 | 121 | 121 | 46 | 33 | 60 | 6 | 5 | 7 | 11 | 10 | 9 |
| 24 | 89 | 89 | 46 | 35 | 60 | 6 | 5 | 7 | 11 | 11 | 9 |
| 25 | 62 | 62 | 47 | 37 | 60 | 6 | 6 | 7 | 12 | 13 | 10 |
| 26 | 42 | 42 | 48 | 40 | 60 | 6 | 5 | 7 | 12 | 12 | 9 |
| 27 | 27 | 27 | 48 | 41 | 59 | 5 | 5 | 6 | 13 | 13 | 9 |
| 28 | 16 | 16 | 50 | 44 | 63 | 5 | 4 | 5 | 13 | 10 | 8 |
| 29 | 8 | 8 | 50 | 50 | 50 | 4 | 4 | 5 | 13 | 11 | 6 |
| 30 | 5 | 5 | 40 | 40 | 60 | 2 | 2 | 3 | 11 | 2 | 3 |
| 31 | 2 | 2 | 0 | 0 | 50 | n.a | n.a | 3 | n.a | n.a | 3 |
| 32 | 1 | 1 | 0 | 100 | 100 | n.a | 2 | 2 | n.a | 2 | 2 |

Tabelle 3: *Hop*-Messergebnisse zu Netzwerk mit 100 Knoten

Folgende Werte sind in den Spalten der Tabellen 4-6 zu finden:

**Spalte 1:** *Hop*-Anzahl der Routen in der Standardimplementierung.

**Spalte 2:** Anzahl der Routen , die die Weglänge aus Spalte 1 haben.

**Spalte 3:** Anzahl der Routen aus Spalte 2, die nach dem Dijkstra-Algorithmus kürzer sein könnten.

**Spalte 4-6:** proz. Anteil der Routen aus Spalte 3, die durch die jeweilige Strategie verbessert werden konnten.

**Spalte 7-9:** Durchschnittswert der ETX-Metrik , der für die Routen aus Spalte 4-6 in der Standardimplementierung erzielt wurde.

**Spalte 10-12:** optimaler Durchschnittswert der ETX-Metrik (nach Dijkstra), der für die Routen aus Spalte 4-6 erzielt werden kann.

**Spalte 13-15:** Durchschnittswert der ETX-Metrik, der für die Routen aus Spalte 4-6 durch die jeweilige Strategie erzielt worden ist.

| Weglänge | Anzahl Wege | optimierbar | verbessert in % | | | Ø Metrik standard | | | Ø Metrik optimal | | | Ø Metrik erzielt | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 78 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 2 | 92 | 23 | 30 | 4 | 4 | 218 | 214 | 214 | 205 | 207 | 207 | 205 | 207 | 207 |
| 3 | 80 | 34 | 15 | 0 | 0 | 323 | n.a. | n.a. | 306 | n.a. | n.a. | 306 | n.a. | n.a. |
| 4 | 74 | 52 | 23 | 23 | 27 | 423 | 412 | 413 | 308 | 207 | 235 | 309 | 209 | 238 |
| 5 | 63 | 57 | 25 | 28 | 37 | 526 | 521 | 520 | 353 | 308 | 317 | 374 | 318 | 326 |
| 6 | 50 | 49 | 29 | 31 | 45 | 626 | 618 | 620 | 370 | 327 | 354 | 408 | 333 | 365 |
| 7 | 37 | 37 | 32 | 43 | 59 | 720 | 715 | 718 | 395 | 395 | 419 | 442 | 427 | 457 |
| 8 | 31 | 31 | 48 | 58 | 71 | 823 | 817 | 820 | 466 | 432 | 475 | 508 | 495 | 503 |
| 9 | 28 | 28 | 61 | 71 | 86 | 927 | 920 | 925 | 482 | 455 | 483 | 517 | 570 | 540 |
| 10 | 25 | 25 | 60 | 80 | 88 | 1031 | 1025 | 1029 | 434 | 439 | 447 | 480 | 603 | 522 |
| 11 | 19 | 19 | 58 | 84 | 89 | 1134 | 1130 | 1133 | 431 | 420 | 427 | 471 | 704 | 518 |
| 12 | 12 | 12 | 50 | 92 | 92 | 1231 | 1233 | 1233 | 386 | 382 | 382 | 422 | 758 | 478 |
| 13 | 7 | 7 | 57 | 100 | 100 | 1340 | 1338 | 1338 | 311 | 313 | 313 | 366 | 912 | 404 |
| 14 | 4 | 4 | 75 | 100 | 100 | 1446 | 1444 | 1444 | 205 | 207 | 207 | 207 | 207 | 207 |

Tabelle 4: ETX-Metrik-Messergebnisse zu Netzwerk mit 50 Knoten

| Weglänge | Anzahl Wege | optimierbar | verbessert in % | | | Ø Metrik standard | | | Ø Metrik optimal | | | Ø Metrik erzielt | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 165 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 2 | 210 | 52 | 40 | 8 | 8 | 235 | 241 | 241 | 209 | 206 | 206 | 209 | 206 | 206 |
| 3 | 199 | 78 | 27 | 3 | 19 | 349 | 367 | 353 | 315 | 300 | 296 | 323 | 321 | 321 |
| 4 | 200 | 118 | 28 | 22 | 36 | 456 | 447 | 454 | 336 | 225 | 299 | 351 | 227 | 306 |
| 5 | 195 | 137 | 27 | 25 | 45 | 568 | 570 | 568 | 394 | 329 | 381 | 428 | 341 | 409 |
| 6 | 191 | 147 | 29 | 31 | 54 | 676 | 680 | 676 | 440 | 384 | 438 | 476 | 398 | 466 |
| 7 | 188 | 156 | 31 | 34 | 58 | 779 | 789 | 782 | 497 | 451 | 506 | 537 | 498 | 539 |
| 8 | 188 | 169 | 36 | 41 | 64 | 881 | 893 | 886 | 519 | 468 | 539 | 562 | 545 | 590 |
| 9 | 176 | 163 | 41 | 50 | 71 | 982 | 995 | 989 | 543 | 493 | 564 | 597 | 610 | 635 |
| 10 | 163 | 155 | 46 | 60 | 76 | 1089 | 1099 | 1093 | 558 | 521 | 583 | 624 | 657 | 672 |
| 11 | 144 | 138 | 51 | 67 | 77 | 1193 | 1201 | 1196 | 579 | 553 | 590 | 676 | 732 | 706 |
| 12 | 123 | 121 | 52 | 70 | 77 | 1297 | 1303 | 1297 | 573 | 563 | 597 | 715 | 799 | 748 |
| 13 | 102 | 102 | 52 | 73 | 76 | 1403 | 1406 | 1401 | 566 | 572 | 602 | 777 | 871 | 810 |
| 14 | 78 | 78 | 54 | 74 | 78 | 1506 | 1510 | 1504 | 589 | 583 | 614 | 898 | 962 | 900 |
| 15 | 57 | 57 | 54 | 79 | 81 | 1609 | 1611 | 1608 | 600 | 588 | 603 | 1027 | 1088 | 994 |
| 16 | 37 | 37 | 54 | 81 | 84 | 1714 | 1716 | 1713 | 613 | 581 | 588 | 1224 | 1244 | 1124 |
| 17 | 21 | 21 | 62 | 90 | 90 | 1819 | 1820 | 1820 | 651 | 587 | 587 | 1452 | 1442 | 1228 |
| 18 | 10 | 10 | 60 | 90 | 90 | 1924 | 1924 | 1924 | 647 | 547 | 547 | 1597 | 1418 | 1173 |
| 19 | 3 | 3 | 67 | 100 | 100 | 2023 | 2022 | 2022 | 611 | 512 | 512 | 1696 | 1684 | 1318 |

Tabelle 5: ETX-Metrik-Messergebnisse zu Netzwerk mit 50 Knoten

| Weglänge | Anzahl Wege | optimierbar | verbessert in % | | | Ø Metrik standard | | | Ø Metrik optimal | | | Ø Metrik erzielt | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 | S1 | S2 | S3 |
| 1 | 349 | 0 | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. | n.a. |
| 2 | 457 | 111 | 41 | 16 | 16 | 228 | 234 | 234 | 209 | 205 | 205 | 209 | 205 | 205 |
| 3 | 462 | 189 | 26 | 8 | 22 | 338 | 345 | 343 | 314 | 310 | 291 | 317 | 312 | 318 |
| 4 | 482 | 287 | 24 | 21 | 32 | 445 | 443 | 445 | 355 | 254 | 309 | 359 | 255 | 316 |
| 5 | 495 | 364 | 23 | 24 | 37 | 555 | 553 | 552 | 420 | 345 | 382 | 429 | 355 | 398 |
| 6 | 514 | 431 | 26 | 28 | 40 | 662 | 661 | 660 | 477 | 405 | 440 | 492 | 422 | 460 |
| 7 | 524 | 474 | 27 | 28 | 41 | 771 | 770 | 769 | 552 | 484 | 505 | 573 | 529 | 536 |
| 8 | 531 | 502 | 29 | 29 | 43 | 881 | 882 | 880 | 615 | 532 | 559 | 648 | 609 | 604 |
| 9 | 531 | 518 | 30 | 29 | 45 | 990 | 991 | 990 | 667 | 586 | 612 | 719 | 714 | 675 |
| 10 | 537 | 533 | 32 | 29 | 49 | 1100 | 1101 | 1100 | 698 | 617 | 655 | 777 | 777 | 736 |
| 11 | 533 | 533 | 33 | 29 | 53 | 1209 | 1210 | 1210 | 734 | 652 | 698 | 848 | 873 | 804 |
| 12 | 526 | 526 | 36 | 29 | 57 | 1317 | 1319 | 1318 | 750 | 666 | 724 | 899 | 910 | 855 |
| 13 | 517 | 517 | 39 | 29 | 61 | 1426 | 1427 | 1427 | 767 | 678 | 752 | 956 | 990 | 909 |
| 14 | 504 | 504 | 41 | 28 | 63 | 1533 | 1534 | 1533 | 775 | 674 | 772 | 1002 | 1005 | 953 |
| 15 | 474 | 474 | 43 | 26 | 65 | 1641 | 1643 | 1641 | 786 | 672 | 789 | 1048 | 1049 | 995 |
| 16 | 437 | 437 | 44 | 24 | 65 | 1748 | 1750 | 1748 | 779 | 644 | 786 | 1073 | 1041 | 1011 |
| 17 | 395 | 395 | 45 | 24 | 65 | 1855 | 1856 | 1855 | 769 | 638 | 788 | 1089 | 1054 | 1025 |
| 18 | 353 | 353 | 47 | 25 | 66 | 1960 | 1958 | 1960 | 739 | 605 | 770 | 1085 | 998 | 1014 |
| 19 | 301 | 301 | 48 | 25 | 66 | 2065 | 2060 | 2066 | 716 | 591 | 757 | 1076 | 976 | 1009 |
| 20 | 248 | 248 | 48 | 26 | 65 | 2171 | 2164 | 2172 | 693 | 556 | 734 | 1091 | 902 | 983 |
| 21 | 199 | 199 | 48 | 27 | 63 | 2275 | 2267 | 2276 | 678 | 549 | 721 | 1126 | 916 | 994 |
| 22 | 158 | 158 | 48 | 30 | 62 | 2377 | 2368 | 2380 | 666 | 547 | 715 | 1172 | 934 | 1005 |
| 23 | 121 | 121 | 46 | 33 | 60 | 2477 | 2470 | 2482 | 650 | 557 | 719 | 1204 | 1044 | 1028 |
| 24 | 89 | 89 | 46 | 35 | 60 | 2578 | 2572 | 2584 | 628 | 559 | 715 | 1247 | 1174 | 1042 |
| 25 | 62 | 62 | 47 | 37 | 60 | 2678 | 2673 | 2686 | 626 | 575 | 716 | 1332 | 1372 | 1056 |
| 26 | 42 | 42 | 48 | 40 | 60 | 2781 | 2778 | 2787 | 592 | 540 | 675 | 1328 | 1343 | 1013 |
| 27 | 27 | 27 | 48 | 41 | 59 | 2885 | 2880 | 2893 | 558 | 513 | 647 | 1360 | 1437 | 1002 |
| 28 | 16 | 16 | 50 | 44 | 63 | 2992 | 2988 | 2996 | 482 | 436 | 547 | 1358 | 1109 | 833 |
| 29 | 8 | 8 | 50 | 50 | 50 | 3083 | 3083 | 3088 | 404 | 404 | 455 | 1396 | 1129 | 594 |
| 30 | 5 | 5 | 40 | 40 | 60 | 3159 | 3159 | 3162 | 204 | 204 | 271 | 1161 | 204 | 271 |
| 31 | 2 | 2 | 0 | 0 | 50 | n.a. | n.a. | 3269 | n.a. | n.a. | 307 | n.a. | n.a. | 307 |
| 32 | 1 | 1 | 0 | 100 | 100 | n.a. | 3369 | 3369 | n.a. | 207 | 207 | n.a. | 207 | 207 |

Tabelle 6: ETX-Metrik-Messergebnisse zu Netzwerk mit 100 Knoten

Folgende Werte sind in den Spalten der Tabelle 7 zu finden:

**Spalte 1:** Anzahl der Knoten im Netzwerk.

**Spalte 2-4:** proz. Anteil der Routen, die in ihrer Metrik durch die jeweilige Strategie.

**Spalte 5-7:** Durchschnittswert der Metrik, um den die Routen aus Spalte 2-4 durch die jeweilige Strategie verschlechtert wurden.

| Knoten im Netzwerk | Wege mit vergrößerter Metrik in % | | | verschlechterte Metrik pro Weg(Ø) | | |
|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S1 | S2 | S3 |
| 25 | 9 | 1,5 | 6 | 31 | 32 | 57 |
| 50 | 8 | 2,5 | 10 | 40 | 17 | 77 |
| 100 | 6 | 2,5 | 20 | 46 | 12 | 59 |

Tabelle 7: Wege mit Vergrößerung der ETX-Metrik

# Literaturverzeichnis

[1] Robert Sombrutzki, Anatolij Zubow, Mathias Kurth, Jens-Peter Redlich: Self-Organization in Community Mesh Networks: The Berlin RoofNet. Proceedings of IEEE OpComm2006, 11 pages.

[2] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, August 1999, pp. 152-162.

[3] James Aspnes, Gauri Shah. Skip graphs. SODA '03 Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms. Pages 384 - 393. Society for Industrial and Applied Mathematics, 2003

[4] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans , Frank Dabek,

[5] H.P.Gumm,M.Sommer. Einführung in die Informatik, 2004

[6] Richard Draves, Jitendra Padhye, Brian Zill. Comparison of Routing Metrics for Static Multi-Hop Wireless Networks, 2004

[7] C. Perkins, E. Belding-Royer, S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. Feb. 2003. http://www.ietf.org/internet-drafts/draftietf- manet-aodv-13.txt.(13.10.2012)

[8] Thomas Heide Clausen, Gitte Hansen, Lars Christensen Gerd Behrmann: The Optimized Link State Routing Protocol, Evaluation through Experiments and Simulation, 2001

[9] David B. Johnson, David A. Maltz .Dynamic Source Routing in Ad Hoc Wireless Networks. Computer Science Department, Carnegie Mellon University, 2004

[10] Jakob Eriksson, Michalis Faloutsos, Srikanth Krishnamurthy. DART: Dynamic Address RouTing for Scalable Ad Hoc and Mesh Networks

[11] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, Antony Rowstron. Virtual Ring Routing: Network Routing Inspired by DHTs, 2006

[12] http://sar.informatik.hu-berlin.de/research/projects/2005-BerlinRoofNet/berlin_roof_net.htm(06.09.2012)

[13] Anselm Ringleben. Performance of DHT over WMN, 2009

[14] Zheng Yao. Ad Hoc Routing, 2005

[15] Matthew Caesar, Miguel Castro1, Edmund B. Nightingale, Greg O'Shea, Antony Rowstron. Virtual Ring Routing: Network Routing Inspired by DHTs

[16] Marcello Caleffi. Mobile Ad Hoc Networks: the DHT paradigm

[17] Hans P. Reiser, Rüdiger Kapitza. Verteilte Hashtabellen, 2003

[18] Mikhail Tarasov. Verteilte Hash Tabelle, 2008

[19] Marcello Caleffi, Luigi Paura. M-DART: Multi-path Dynamic Address RouTing

[20] Zhigang Jin, Minfang Yu, Yantai Shu. Improving wireless TCP performance by link probing. Electrical and Computer Engineering, 2003. IEEE CCECE 2003.

[21] http://sarwiki.informatik.hu-berlin.de(03.09.2012)

[22] http://de.wikipedia.org/wiki/Hashfunktion(06.09.2012)