Humboldt-Universität zu Berlin,
Institut für Informatik,
Lehrstuhl für Systemarchitektur

Betreuer: Anatolij Zubow
Gutachter: Prof. Jens-Peter Redlich

Selbstorganisation in drahtlosen Ad-Hoc Multi-Hop Maschennetzwerken

Dynamic Host Configuration Protocol und Address Resolution Protocol

Studienarbeit vorgelegt von
Name: Robert Sombrutzki
Matrikelnummer: 166690
E-Mail: sombrutz@informatik.hu-berlin.de

Zusammenfassung

Die Anzahl drahtloser Community-Maschennetzwerke hat in den letzten Jahren stark zugenommen. Da diese auch für unerfahrene Menschen benutzbar sein sollen, ist Selbstorganisation sehr wichtig. Auf der einen Seite werden in dieser Arbeit Ideen zur Selbstorganisation erläutert, bei der die Clienten von Aufgaben wie manueller IP-Konfiguration befreit werden. Auf der anderen Seite soll auch das Community-Maschennetzwerk vollständig selbstorganisierend sein, so dass z.B. kein Administrator oder Provider benötigt wird. Es wird die Architektur des BerlinRoofNets (BRN) erläutert und eine verteilte Realisierung des „Dynamic Host Configuration Protocols“ (DHCP), des „Address Resolution Protocols“ (ARP) und eines Protokolls zur Auswahl von Gateways auf der Basis einer verteilten Hashtabelle (DHT) vorgestellt.
Es werden die Ergebnisse aus Simulationen und aus Messungen in einer realen Testumgebung präsentiert und ausgewertet. Dabei wird der auf einer DHT-basierende DHCP- bzw. ARP-Dienst mit einem traditionellen Ansatz, also der Verwendung eines zentralen Servers und dem netzweiten Fluten der Anfragen, hinsichtlich Zuverlässigkeit sowie benötigter Netzwerkressourcen verglichen. Es zeigt sich, dass die hier vorgestellte Realisierung von DHCP und ARP auf Basis einer verteilten Hashtabelle zuverlässiger und effizienter ist.

Inhaltsverzeichnis

1. Einleitung

Ein drahtloses Community-Netzwerk ist ein Netzwerk aus unterschiedlichen Geräten wie Laptops und Access Points (AP), die drahtlos verbunden sind und mit Hilfe z.B. des 802.11-Standards kommunizieren. Die Knoten sind willkürlich angeordnet und mobil, so dass sich die Netzwerktopologie schnell und unvorhersagbar ändern kann. Solche Netzwerke können Internetverbindung haben oder unabhängig von diesen arbeiten. Durch die autonome Natur solcher Systeme muss ein Auto-Konfigurationsmechanismus vorhanden sein.
Heutzutage spielen drahtlose Multihop-Mesh-Netzwerke als Community-Netzwerke eine wichtige, immer größer werdende Rolle, da sie in Städten eine Internetverbindung zur Verfügung stellen können. Weiterhin werden durch sie Anwendungen möglich wie:

- Geteilter Internetzugang
- Kommunikationsnetze für Rettungsdienste
- Überwachung
- (Ver-)Teilen von Ressourcen (z.B. Backup)

Die Arbeit ist wie folgt aufgebaut. Im ersten Abschnitt werden existierende Community-Netzwerke beschrieben. Auf das BerlinRoofNet, seine Architektur und die angebotenen Dienste wird dabei detaillierter eingegangen. Es wurden grundlegende Dienste wie Dynamic Host Configuration Protocol (DHCP), Address Resolution Protocol (ARP) und Funktionen für Internet-Gateways auf der Grundlage einer verteilten Hashtabelle (DHT) implementiert und untersucht. Die Ergebnisse der Simulationen im Netzwerksimulator ns2 und die Beobachtungen in der BRN-Testumgebung werden erörtert. Es wird gezeigt, dass diese Ansätze verglichen mit traditionellen Ansätzen zu verbesserten Antwortzeiten führen und zudem weniger Netzwerkressourcen benötigt werden.

1.1. Drahtlose Community-Maschennetzwerke

Ein drahtloses Community-Multi-Hop-Maschennetzwerk (Abb. 1) besteht aus Knoten und Clienten. Die Knoten bilden das Maschennetzwerk, wobei jeder automatisch seine in Funkreichweite befindlichen Nachbarn erkennt. Meist ist nur ein Teil von ihnen über DSL oder z.B. WiMax[23] mit dem Internet verbunden. Knoten ohne Internetverbindung können ihre Pakete mit Hilfe der anderen ins Internet senden (Multi-Hop-Routing). Die Clienten sind nicht direkt am Routing im Maschennetzwerk beteiligt, d.h. sie leiten keine Pakete weiter und stellen auch keine Internetverbindung zur Verfügung. Bei ihnen handelt es sich um Endgeräte wie Laptops mit unmodifizierter Software.

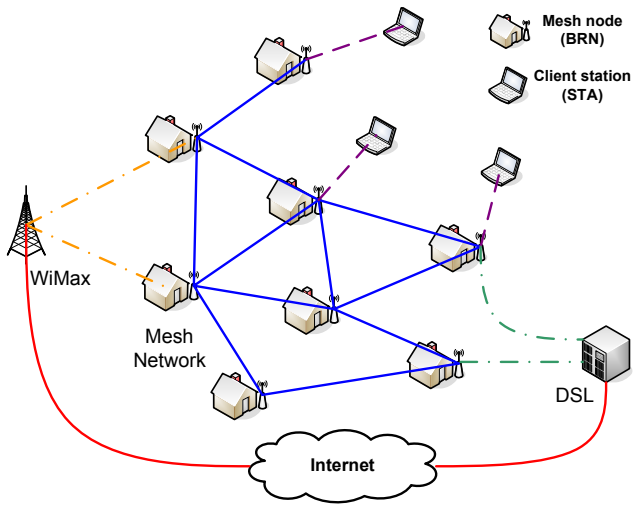

Abbildung 1: Drahtloses Maschennetzwerk

1.2. Selbstorganisation als Grundlage für Community-Netzwerke

Da ein Community-Netzwerk auch von Menschen mit weniger Erfahrung im Umgang mit Computern benutzt wird, sollte es sehr leicht zu benutzen und zu erweitern sein. Deshalb spielt der Begriff Selbstorganisation besondern bei größeren Netzen ein zentrale Rolle. Darunter versteht man, dass neue Geräte sehr schnell dem bestehenden Netz hinzugefügt werden können, ohne dass diese vorher konfiguriert bzw. mit zusätzlicher Software ausgestattet werden müssen. Ein zentraler Server, der neue Teilnehmer automatisch mit Software bzw. einer Konfiguration versorgt, könnte eine Lösung sein, jedoch muss dieser durch einen Administrator gepflegt werden. Die Tatsache, dass die Größe des Netzes nicht vorhersehbar ist und sich durch den Ausfall und das Hinzukommen von Knoten ständig ändern, erschwert die Konfiguration eines solchen Servers, erfordert so viel Zeit und macht es fehleranfällig. Selbstorganisation vereinfacht die Konfiguration und setzt sich aus Self-replication und Autoconfiguration zusammen.

Neue Geräte, die in das Community-Netzwerk kommen, erhalten die benötigte Software von den anderen Knoten im Netz. Dies funktioniert ähnlich wie ein Computervirus, der, auf einem Rechner installiert, sich auf andere Computer kopiert. Dies nennt man Self-replication.

Mit Hilfe von Autoconfiguration werden die Geräte konfiguriert und dabei Parametern wie z.B. IP-Adresse festgelegt. Da kein Server im Netzwerk verfügbar ist, werden die Ressourcen durch das ganze Netze verwaltet. Zeroconf ist eine Technik mit deren Hilfe Geräte eine IP-Adresse bekommen und so Kommunikationsnetze aufbauen können. Dabei wählt ein Gerät selbstständig eine IP-Adresse und fragt dann im Netz, ob diese bereits von einem anderen verwendet wird. Im Fall eines Konfliktes, also wenn die IP bereits vergeben ist, wird eine neue Adresse gewählt und der Vorgang beginnt erneut. Dieses Prinzip kann besonders bei großen Netzen mit vielen Teilnehmern sehr aufwendig und langwierig sein. Eine andere Technik beruht auf der Verwendung von verteilten Datenbanken. Hierbei werden die schon vergebenen Ressourcen in einer verteilten Datenbank, die die Teilnehmer des Netzes bilden, verwaltet.

1.3. BerlinRoofNet

Das BerlinRoofNet (BRN) ist ein experimentelles drahtloses Netzwerk in Berlin. Es besteht aus circa 50 Knoten. Das Hauptziel des BRN ist die Erforschung neuer Protokolle, z.B. für das Routing und verteilter Dienste, wie der hier vorgestellte DHCP-Dienst. Das Netz bietet den Studenten außerdem den Zugang zum Internet und z.B. die Möglichkeit mit Hilfe von Voice over IP (VoIP) miteinander zu telefonieren. Dabei ist ein wichtiges Ziel den Endbenutzer einen Netzwerkzugang in voller Selbstorganisation zu ermöglichen, d.h., dass nur wenig Konfiguration auf dem Gerät des Benutzers nötig ist. Dafür wurden verschiedene Konzepte und Dienste entwickelt, die im folgenden Abschnitt beschrieben werden.

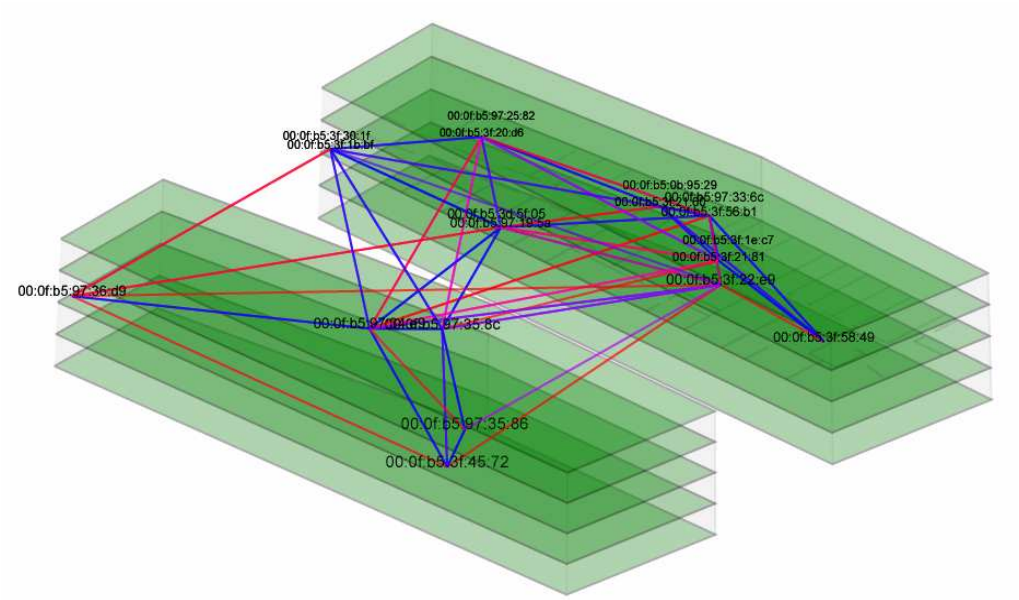

Abbildung 2: Indoor-Testbed

Das BRN ist ein selbstorganisierendes System. Es besitzt es eine 2-Schichtenarchitektur. Die erste Schicht, das BRN Kern-Netzwerk wird von den BRN-Knoten gebildet, während die Clienten die 2. Schicht bilden. Für die Kommunikation zwischen den Clienten und den Knoten wird das 802.11-Protokoll verwendet. Die Knoten arbeiten im Infrastrukturmodus, d.h. sie verhalten sich gegenüber den Clienten wie Access Points. Für die Kommunikation zwischen den Knoten wird das proprietäre BRN-Protokoll verwendet. Dadurch können 2 Arten von Links unterschieden werden:

- BRN-Knoten ↔ BRN-Knoten : BRN-Protokoll, basiert auf 802.11
- Clienten ↔ BRN-knoten : IEEE 802.11 Protokoll

Eine direkte Kommunikation zwischen den Clienten ist also nicht möglich. Sie kommunizieren indirekt mit Hilfe der BRN-Knoten. Der Vorteil von diesem Ansatz ist, dass die Clienten nicht modifiziert werden müssen und dass sich jedes 802.11 kompatible Gerät (z.B. VoIP-Telefon) mit Hilfe des Infrastruktur-Modus mit dem BRN-Netzwerk verbinden kann.

1.3.1. Aufbau

Aus Sicht des Clienten sieht das BRN-Netzwerk wie ein virtueller Access Point aus, mit dem man sich mit Hilfe des IEEE 802.11 Infrastruktur-Modus verbinden kann. Nur für die Kommunikation zwischen den Knoten des BRN-Netzwerkes wird das proprietäre BRN-Protokoll verwendet. Das BRN-Netzwerk kann deshalb als großer virtueller Ethernet-Switch beschrieben werden, der z.B. Ethernet-Frames von einem Clienten empfängt und diese an das Ziel weiterleitet. Die Algorithmen, die für interne Paketweiterleitung verwenden werden, sind dieselben, die man auch beim Layer-3 Routing (IP) verwendet. Ein Vorteil von diesem Ansatz ist, dass das BRN Layer-2 agnostisch ist, weil es nur Ethernet Frames betrachtet, d.h. IPv4 kann ohne viel Veränderung an der BRN-Software durch andere Protokolle ersetzt werden. Außerdem, wird in dieser Arbeit gezeigt, dass es dieser Ansatz möglich macht, bestimmten IPv4 und ARPv4-Verkehr zu optimieren.

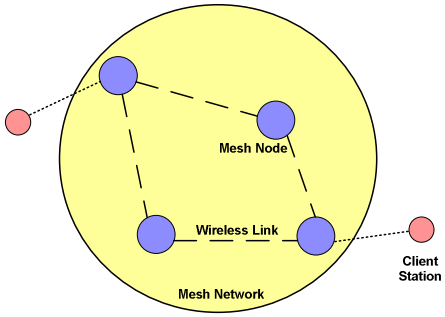

Abbildung 3: BRN als Virtueller verteilter Access Point

1.3.2. Routing

Das Routingprotokoll, welches im BRN verwendet wird, basiert auf *Dynamic Source Routing* (DSR) [21]. Der Hauptunterschied zu anderen Ansätzen ist, dass das Routing auf Layer-2 geschieht: Knoten im Netz werden anhand ihrer Ethernetadresse identifiziert und nicht durch ihre IP. Es wird ETX [22] verwendet, um die Routen im Netz zu finden, die den höchsten Durchsatz bieten. Im Gegensatz zu anderen Ansätzen (z.B. Freifunk-Netzwerk mit OLSR) leiten nur die Knoten des BRN die Pakete weiter, Clienten jedoch nicht.

2. Selbstorganisation

In Community-Netzwerken ist aufgrund des fehlenden Administrators die Selbstorganisation ein wichtiger Aspekt. Sie ermöglicht, dass das Netz alle Dienste ohne Eingriff des Users zur Verfügung stellt, da die Dienste nicht konfiguriert bzw. administriert werden müssen.

2.1. Ähnliche Arbeiten

Zurzeit gibt es mehrere drahtlose Community-Netzwerke in verschiedenen Städten. *NYCwireless* zum Beispiel unterhält offene Hotspots in öffentlichen Plätzen in New York und arbeitet mit öffentlichen und nicht-kommerziellen Organisationen und Personen zusammen. Ein anderes Projekt ist *Seattle Wireless*, dessen Ziel es ist, ein drahtloses Community-Netzwerk in Seattle (Washington) aufzubauen, welches parallel zum Internet arbeitet, um so die Verbindungen zwischen den Netzteilnehmern zu verbessern. Das *Austin Wireless City Projekt* ist ein weiteres Beispiel für ein Community-Netzwerk. Das Ziel ist es, die Verfügbarkeit und Qualität von freien und öffentlichen WiFi-Zugängen in Austin (Texas) zu verbessern. Bei diesem Projekte wird besonders viel Arbeit in die Ausbildung, Unterstützung und Fortbildung von Mitarbeitern investiert, die die freien drahtlosen Hotspots betreuen. Ähnliche Projekte existieren auch in Europa. Die *Freifunk.net*–Initiative unterstützt offene drahtlose Netzwerke in den deutschsprachigen Teilen Europas. Dabei ist es das Ziel, auch dort ein Zugang zum Internet zur Verfügung zu stellen, wo kein traditioneller Breitbandzugang zur Verfügung steht. Das Projekt unterstützt z.B. ein Community-Netzwerk in Berlin, da auch dort in vielen Teilen der Stadt kein Breitbandinternetzugang existiert.

Es gibt außerdem noch einige wissenschaftliche Gruppen, die sich mit drahtlosen Maschennetzwerken beschäftigen. Die *Microsoft Network Research Group* setzt sich besonders mit selbstorganisierender Nachbarschaft in solchen Netzwerken auseinander.
Die *Parallel & Distributed Operating System Group* vom MIT haben das MIT Roofnet aufgebaut, ein experimentelles IEEE 802.11 Maschennetzwerk in Cambridge (Massachusetts).

Die Selbstorganisation ist bei diesen Netzen sehr unterschiedlich gelöst. In einigen der oben erwähnen Maschennetzwerken erfolgt die IP-Vergabe manuell. Die Benutzer des Freigunk.net müssen z.B. ihre Geräte in eine Datenbank eintragen und bekommen dann eine IP, die mit Hilfe der Postleitzahl des Ortes an dem sich der Teilnehmer befindet, bestimmt wird. Beim MIT-RoofNet agiert jeder Knoten des Maschennetzwerkes als DHCP-Server und vergibt private IPs an die assoziierten Clienten. Mit Hilfe von *Network Address Translation* (NAT) wird den Clienten der Zugriff aus das Internet ermöglicht. Clienten, die an unterschiedlichen APs angemeldet sind, können jedoch nicht direkt miteinander kommunizieren.

2.2. Netzwerkdienste

In Abschnitt 1.2. wurden Anforderungen, die man an ein Community-Netzwerk stellt, beschrieben. Im BRN ist die Selbstorganisation gegenüber des Clienten durch eine Reihe von Diensten realisiert: Address Resolution Protocol (ARP), Dynamic Host Configuration Protokoll (DHCP) und Auswahl des Internet-Gateways. Im folgenden Abschnitt wird beschrieben, wie diese Dienste arbeiten und wie sie umgesetzt wurden. Es wird gezeigt, dass Dienste, deren Protokoll zum großen Teil auf Kommunikation mit Hilfe von Broadcasts basieren, wie z.B. ARP und DHCP, in drahtlosen Netzwerken nicht praktikabel sind. Bei ARP wird eine Broadcast-Nachricht verwendet, um zu einer gegebenen IP-Adresse die Mac-Adresse zu ermitteln, wodurch das gesamte Netzwerk mit Paketen geflutet wird. Die Idee von dem hier vorgestellten Ansatz ist es, diese Verfahren zu ersetzten und so den Verbrauch an Netzwerkressourcen zu reduzieren.
Um dies zu erreichen, werden Techniken wie z.B. verteilte Hashtabellen (DHT) verwendet, welche aus dem Bereich der Peer-to-Peer-Netzwerke bekannt sind. Dadurch können die verwendeten Broadcast-Nachrichten durch Unicast-Nachrichten ersetzt werden. Jedoch sind vorhandene Ansätze für DHTs, wie z.B. Chord[12], für drahtlose Netzwerke nicht geeignet und deshalb wurde hier eine modifizierte Variante von Chord verwendet, die an die Eigenschaften von drahtlosen Maschennetzwerken angepasst wurde.
In den folgenden Abschnitten werden die im BRN verwendeten Netzwerkdienste beschrieben.

2.2.1. Dynamic Host Configuration Protokoll (DHCP)

Die Konfiguration der IP-Adressen für einige wenige Rechner ist nicht sehr aufwendig und schnell gemacht. Wenn es jedoch sehr viele Geräte sind, steigt der Aufwand. In Community-Netzwerken, bei denen jeder Benutzer selbst die IP-Adresse eintragen muss, besteht zu dem die Gefahr, dass IPs mehrfach vergeben werden und es so zu Kommunikationsproblemen kommt. Zudem muss jeder Rechner erneut konfiguriert werden, wenn sich zum Beispiel die IP-Adresse des DNS-Servers ändert.

Mit Hilfe der dynamischen IP-Adressvergabe mit Hilfe von DHCP hat man eine zentrale Vergabestelle für IP-Adressen und weitere IP-Konfigurationen wie DNS- und WINS-Server, Gateways und vielen anderen Parametern.

Der Client erfragt bei beim Server eine IP-Adresse, der diese entweder anhand einer statischen oder dynamischen Datenbank vergibt.

2.2.1.1. Begriffe

Lease(-Time):

Eine Lease(-Time) ist eine Zeitdauer, während der Client seine IP-Adresse nutzen darf. Nach Ablauf der „Leihdauer“ darf der Client diese Adresse diese IP-Adresse dann nicht mehr benutzen. Umgangssprachlich bezeichnet man diese Zeit als *Lease*.

Scope:
Ein Scope ist der IP-Adressbereich, aus dem der DHCP-Server seine IP-Adressen vergibt.

DHCP-Option:
Eine DHCP-Option ist eine optionale Information für den TCP/IP-Stack des Clienten. Über die DHCP-Optionen werden weitere Informationen wie zum Beispiel die IP-Adresse des Gateways oder des DNS-Servers an den Clienten übermittelt.

2.2.1.2. Funktionsweise

Erfragen einer IP-Adresse
Ein Client, der keine IP-Adresse hat, z.B. nach dem Booten, schickt zuerst eine *DHCP-Discover*-Nachricht an alle im Netz befindlichen DHCP-Server. Diese Nachricht wird als Broadcast versendet. Als Absenderadresse wird dabei 0.0.0.0 und als Zieladressen 255.255.255.255 (Broadcast) verwendet. Jeder DHCP-Server, der diese Nachricht erhält, schickt dem Clienten eine *DHCP-Offer*-Nachricht. Diese Nachricht enthält die vom Server vorgeschlagene IP für den Clienten. Diese Nachricht wird ebenfalls bei Broadcast versendet.
Der Client erhält also unter Umständen mehrere Angebote von mehreren Servern, von denen er eine IP auswählt. Er sendet danach ein *DHCP-Request* an den Server, der ihm diese IP angeboten hat, worauf dieser eine *DHCP-Acknowledge*-Nachricht schickt. Diese Nachricht enthält weitere Konfigurationsparameter, wie z.B. IP-Adresse des Gateways.
Mit Hilfe eines *ARP-Request*, kann der Client sicherstellen, dass der DHCP-Server nicht durch einen Fehler die IP-Adresse mehrfach vergeben hat. Sollte dies der Fall sein, so kann der Client die IP-Adresse mit einer *DHCP-Decline*-Nachricht zurückweisen.

Verlängern
In der *DHCP-Acknowledge*-Nachricht sendet der Server auch die so genannte Lease-Time mit. Diese gibt an, wie lange der Client die IP verwenden darf. Diese Zeit wird von Administrator des DHCP-Servers eingestellt. Nach Ablauf der Hälfte dieser Zeit, sendet der Client dem Server einen erneuten *DHCP-Request* und fragt damit den Server, ob er die IP-Adresse weiter verwenden darf. Diese Nachricht wird per Unicast verschickt. In der Regel bestätigt der Server diese Anfrage mit einem *DHCP-Acknowledge* und schickt dem Client dabei noch einmal alle benötigten Konfigurationsparameter. Sollte der Server nicht antworten, so kann der Client die IP-Adresse noch bis zum Ablauf der Lease weiter verwenden. Wenn jedoch 87.5 % dieser Zeit abgelaufen sind, so sieht der Standard vor, dass der Client die Lease bei einem beliebigen DHCP-Server verlängert. Dazu sendet er ein *DHCP-Request* per Broadcast an alle DHCP-Server im Subnetz und erfragt eine Verlängerung seiner IP.
Sollte der Client bis zum Ablauf der Lease keine Verlängerung erhalten haben bzw. keine Verlängerung erfragt haben, so muss er seine Konfiguration verwerfen und per *DHCP-Discover* die Adresszuweisung erneut beginnen.

2.2.2. Address Resolution Protocol (ARP)
Die MAC-Adressen von Ethernet-Netzwerkkarten sind weltweit eindeutig und werden von den Herstellern vergeben. Bei einigen Netzwerkprotokollen, wie z.B. IPv6, lässt sich die MAC-Adresse aus der Netzwerkadresse ermitteln. Da IPv4-Adressen jedoch nur aus 32 Bits bestehen, ist es nicht möglich die MAC-Adresse innerhalb der IP-Adresse zu speichern. Zum Versenden eines Paketes muss der MAC-Layer deshalb erst die dazugehörige MAC-Adresse ermitteln. Dies geschieht mit Hilfe des *Address Resolution Protocols* (ARP).
Mit diesem Protokoll kann jeder Rechner in einem Netzwerk die Ethernet-Zieladresse eines anderen Rechners ermitteln.

2.2.2.1. Paketstruktur

Das Typefeld des Ethernet-Frame enthält beim ARP-Request bzw. -Reply den Wert 0x0806 (2054). Das ARP-Paket folgt direkt dem Header. Um die Minimale Framelänge von 64 Byte zu erreichen, die für CRC notwendig ist, müssen an das ARP-Paket zusätzliche Byte angefügt werden (Padding).
Da im Paket Felder für Adresstypen und Protokollgrößen vorgesehen sind, lässt sich ARP auch für andere, zukünftige Protokolle verwenden, obwohl es ursprünglich für IPv4 und MAC-Adressen vorgesehen war. So wird z.B. bei IPv6 die Protokolladressgröße statt auf 4 auf 16 Bytes gesetzt und die Adressfelder werden auf 128 Bits (=16 Byte) verlängert.

Bit 0-7	Bit 8-15	Bit 16-23	Bit 24-31
Hardwareadresstyp (1)		Protokolladresstyp (0x800)	
Hardwareadressgröße (6)	Protokolladressgröße (4)	Operation	
Quell-MAC-Adresse			
Quell-MAC-Adresse		Quell-IP-Adresse	
Quell-IP-Adresse		Ziel-MAC-Adresse	
Ziel-MAC-Adresse			
Ziel-IP-Adresse			

Abb. 4: ARP-Paket

- **Hardwareadresstyp** (2 Byte) enthält den Typ der MAC-Adresse im Paket (für Ethernet: 1).
- **Protokolladresstyp** (2 Byte) enthält den Protokolltyp, der für die MAC-Adresse angefordert wird (für IPv4-Adressen: 0x0800 (2048)).
- **Hardwareadressgröße** (1 Byte) enthält die Größe der MAC-Adresse (für Ethernet: 6).
- **Protokolladressgröße** (1 Byte) enthält die Größe des Protokolls (für IPv4: 4, für IPv6: 16).
- **Operation** (2 Byte) enthält den Wert, der angibt, welche Operation ausgeführt werden soll (1 für ARP-Anforderung, 2 für ARP-Antwort).
- **Quell-MAC-Adresse** (6 Byte) enthält bei einer ARP-Anfrage die MAC-Adresse des Senders. In einer ARP-Antwort enthält es die MAC-Adresse des antwortenden Hosts.
- **Quell-IP-Adresse** (4 Bytes bei IPv4, 16 Bytes bei IPv6) enthält bei einer ARP-Anfrage die IP-Adresse des anfragenden Hosts. In einer ARP-Antwort enthält es die IP-Adresse des antwortenden Hosts.
- **Ziel-MAC-Adresse** (6 Byte) ist in einer ARP-Anforderung undefiniert. In einer ARP-Antwort enthält es die MAC-Adresse des anfragenden Hosts.
- **Ziel-IP-Adresse** (4 Bytes bei IPv4, 16 Bytes bei IPv6) ist bei einer ARP-Anforderung die IP-Adresse des gesuchten Hosts. In einer ARP-Antwort enthält es die IP-Adresse des anfragenden Hosts.

2.2.2.2. Funktionsweise

Möchte ein Rechner die MAC-Adresse eines anderen wissen, von dem er nur die IP-Adresse kennt, so sendet er ein ARP-Request. Diese Anfrage wird per MAC-Broadcast verschickt, d.h. die Zieladresse des Paketes ist die Broadcast-Adresse ff-ff-ff-ff-ff-ff. Dieses Paket erhält jeder Rechner im Netz. Der Host, der die entsprechende MAC-Adresse kennt, antwortet mit einem ARP-Reply. Dabei sendet er die passende MAC-Adresse zurück. Es muss nicht unbedingt der

Host den ARP-Reply senden, der auch die IP-Adresse hat. Andere können die Anfrage ebenfalls entgegennehmen und mit Hilfe ihres ARP-Caches beantworten. Bei Erhalt eines ARP-Reply, fügt der Host die entsprechende Kombination aus MAC- und IP-Adresse in seine ARP-Tabelle ein. Die Einträge werden nach einen gewissen Zeitintervall gelöscht, da diese eventuell nicht mehr aktuell sind. Wie lange die Einträge gespeichert bleiben, hängt von der Implementierung ab und kann z.T. vom Benutzer eingestellt werden.

3. Design - Dienste auf Basis einer DHT

Im diesem Abschnitt werden zu Beginn die allgemeinen Probleme der bisherigen Lösungen für Dienste in Multihop-Maschennetzwerken erläutert. Anschließend wird gezeigt, wie mit Hilfe moderner P2P-Technologie diese Dienste effizient realisiert werden können.

3.1. Verwendung von Broadcast in Protokollen

Broadcast-Nachrichten werden bei vielen Protokollen verwendet, bei denen das Ziel der Anfrage unbekannt ist bzw. nicht klar ist, wer die Anfrage beantworten kann. Ein Beispiel ist das Address-Resolution Protokol (ARP), welches vom Internet Protokoll (IP) verwendet wird, um zu einer IP die Hardware Adresse zu ermitteln. Im Allgemeinen wird eine ARP-Anfrage mit Hilfe eines Ethernet-Broadcast versendet und so von allen Systemen in derselben Kollisions-Domäne empfangen. Dies stellt sicher, dass wenn das Ziel der Anfrage mit dem Netzwerk verbunden ist, es diese erhält. Andere Systeme verwerfen dieses Paket. In einem drahtgebundenen Netzwerk ist dies eine praktikable Lösung, in einem Maschennetzwerk hingegen verursacht es einige Probleme. Da nicht jeder Knoten im Netzwerk alle anderen direkt erreicht, muss die Broadcast-Nachricht von anderen weitergeleitet, d.h. erneut versendet werden. Da sich die Empfangs- und Sendebereiche nahe beieinander liegender Knoten überlagern, verursacht dieses Weiterleiten (Fluten) viele Kollisionen. Dies bezeichnet man als Broadcast-Storm-Problem.

In dieser Arbeit wird durch Verwendung einer verteilten Hashtabelle versucht, dieses Problem zu verhindern. Es wird dabei die Tatsache ausgenutzt, dass die Clienten nicht direkt miteinander kommunizieren, sondern stattdessen die Knoten des BRN für die Kommunikation genutzt werden. Ein Client, der mit einem anderen kommunizieren will, muss sich zunächst mit einem AP assoziieren, um dem Netzwerk beizutreten. Danach bekommt er mit Hilfe von DHCP eine IP. Um die MAC-Adresse des anderen Clienten zu ermitteln, sendet er eine ARP-Anfrage. Bei der hier vorgestellten Lösung wird diese Anfrage von dem BRN-Knoten nicht an die anderen Clienten weitergeleitet, sondern mit Hilfe einer DHT beantwortet. Dadurch wird das sonst übliche Flooding verhindert. Man darf dies jedoch nicht mit einem ARP-Proxy verwechseln, da dieser immer seine MAC-Adresse zurückgibt und nicht die des eigentlichen Zielrechners.

3.2. P2P Technologien

Peer-to-Peer-Systeme (P2P) sind verteilte Systeme ohne zentrale Kontrolle, bei denen auf jedem Computer die gleiche Software läuft. Im Gegensatz zum zentralisierten Ansatz können hier die Teilnehmer zusammen ein P2P-Netzwerk aufbauen ohne in zusätzliche Hardware (z.B. Server) zur Koordinierung zu investieren. Außerdem sind P2P-Systeme durch ihre Dezentralisierung und Verteilung robust gegen jegliche Art von Ausfall, und so besonders zur Langzeitspeicherung und Dauerberechnungen geeignet.
Chord ist ein sehr interessanter Ansatz um den Anspruch der verteiltet Datenspeicherung gerecht zu werden. Es unterstützt eine Hashfunktion, durch die es möglich ist, den Speicherort von gesuchten Daten zu ermitteln. Es stellt dabei lediglich eine Operation zur Verfügung: finde Knoten zu einem gegebenen Schlüssel. Eine Datenlokalisierung kann sehr einfach auf Basis von Chord implementiert werden. Jedem Dateneintrag wird dabei einem Schlüssel

zugewiesen und das Schlüssel/Daten-Paar wird nun auf dem Knoten gespeichert, der für diesen Schlüssel verantwortlich ist. Chord ist auch dann sehr effizient, wenn Knoten neu ins Netzwerk kommen bzw. es verlassen und dies auch sehr oft passiert. Das Feld von P2P hat viel gemeinsam mit Maschennetzwerke, da beide eine robuste, verteilte Lösung für die Suche benötigen. Man stelle sich ein Netzwerk aus hunderten von Knoten und eine Anfrage nach einem Datum vor. Die Lösung, einen Broadcast zu verwenden, um die Information zu erhalten, ist dabei nicht sehr sinnvoll. Auf der anderen Seite schwankt die Verbindung zwischen den Knoten eines drahtlosen Maschennetzwerkes aufgrund sich ständig ändernder Linkqualitäten und der Mobilität der Knoten. Ein gutes Suchsystem muss also mit den ständigen Änderungen des zugrunde liegenden Netzwerkes zurechtkommen. So können z.B. Knoten, die lange Zeit erreichbar waren, plötzlich ausfallen.
Es zeigt sich also, dass es Probleme mit sich bringt, Chord in einem Maschennetzwerk zu verwenden. So sind Knoten, die im Overlay-Netzwerk von Chord dicht beieinander liegen, nicht zwangläufig auch im physikalischen Netzwerk dicht zusammen. Der Grund dafür ist, dass die Hashfunktion von Chord, die geographische Position der Knoten nicht berücksichtigt. Chord wurde für drahtgebundene Netzwerke entwickelt, bei der die geographische Position bzw. die Entfernung zwischen den Rechner keine große Rolle spielt. Bei einem drahtlosen Netzwerk ist für die Übertragungsgeschwindigkeit sehr entscheidend, wie oft ein Paket im physikalischen Netz weitergeleitet werden muss. Aber auch Verbindungen mit der gleichen Anzahl von Hops sind aufgrund der unterschiedlichen Qualitäten der Links, verschieden.

Der folgende Ansatz versucht diese oben erwähnten Probleme zu lösen. Die Autoren von „CHR: Distributed Hash Table for Wireless Ad Hoc Networks“ [18] schlagen vor, als Eingabe für die Hashfunktion auch die geographische Position des Knoten zu nutzen, um diese mit einzubeziehen und die Anzahl der Weiterleitungen im Netzwerk für die Anfragen zu minimieren. Die Positionsbestimmung wird in der Praxis mit Hilfe von „Global Positioning System“ (GPS)[25] gemacht. Der größte Nachteil dieses Ansatzes ist GPS selbst. Die meisten Maschenetzwerke befinden sich innerhalb von Gebäuden oder in dicht bebauten Städten, wo der Einsatz von GPS nicht praktikabel ist.

3.3. Falcon

In diesem Abschnitt werden die Grundprinzipen von Falcon und Lösung für die erläuterten Probleme erklärt. Falcon ist eine erweiterte Version von Chord, die die Charakteristik von drahtlosen Maschennetzwerken berücksichtigt. Da das eigentliche Thema der Arbeit jedoch die Selbstorganisation ist, wird hier nur einen kurzen Überblick geben. Es werden die Unterschiede zwischen Chord und Falcon erläutert, wobei vorrangig die Anfragen an die DHT behandelt werden.

3.3.1. Funktionsweise

Die Originalversion von Chord dient als Grundlage für Falcon. Es wurde einfache Optimierungen zu Chord hinzugefügt, so dass die DHT von der Charakteristik von Maschennetzwerken profitieren kann. Zu einem Hop im Overlay-Netzwerk gehören meist mehrere physikalische Hops im Maschennetzwerk. Die Knoten, welche die DHT-Nachrichten weiterleiten, können versuchen diese über bessere Wege als von Absender vorgesehen, zu schicken.
Als Beispiel dient das Netzwerk aus Abbildung 5. Der Abstand zwischen Knoten 1 und 2 beträgt im Overlay-Netzwerk ein Hop, während im physikalischen Netz 2 Hops dazwischen liegen. Folgende Optimierung ist in einem solchen Fall möglich: Beim Weiterleiten eines DHT-Paketes im physikalischen Netz analysiert der Knoten, der es weitersendet, das Paket und schaut, ob er aufgrund seiner Fingertabelle eine bessere Route zum Ziel im Overlay-Netzwerk kennt. Ist dies der Fall, so ändert er die Route des Paketes. Falcon versucht also

durch Minimierung der Hops im Overlay-Netzwerk, die Anzahl der Hops im physikalischen Netz zu reduzieren. Das ist ein großer Vorteil gegenüber Chord, wo nur die Forwarder im Overlay-Netzwerk das nächste Ziel bestimmen. Dies wird hier als Basisversion von Falcon bezeichnet.

Eine weitere Optimierung besteht darin, die Fingertabelle der physikalischen Nachbarn zu verwenden. Je mehr ein Knoten über das Overlay-Netzwerk weiß, desto kürzer sind die Routen zum Ziel eines Paketes. Ein Knoten hat zum Beispiel k Nachbarn. Wenn er deren Fingertabelle kennt, so erhöht sich die Anzahl der im bekannten Knoten auf das bis k+1-fache. Mit Hilfe der Linkprobe-Pakete, die von ETX-basierten Protokollen verwendet werden, sendet jeder Knoten seine Fingertabelle an seine lokalen Nachbarn. Diese Pakete haben meist eine Gösse von 1 KByte, wobei dabei bisher nur wenige Bytes genutzt werden. Der ungenutzte Platz wird von Falcon benutzt, um die Fingertabelle zu versenden. Da die Linkprobe-Pakete regelmäßig versendet werden (alle 3 s), ist es unwahrscheinlich, dass alte Information verteilt wird.
Der aktive Mechanismus von Chord zum Erkennen ausgefallener Knoten ist sehr aufwendig und wird deshalb bei Falcon wie folgt ersetzt: Anstatt periodisch die Knoten, die in der Fingertabelle stehen, zu überprüfen, wird bei Falcon mit Hilfe der Linkprobe-Pakete ein Ausfall der physikalischen Nachbarn erkannt. Sollte ein Knoten diese Pakete nicht mehr von seinem Nachbarn erhalten, so wird dieser als ausgefallen erkannt. Er informiert dann alle Knoten des Netzwerkes, die den Ausgefallenen in ihrer Fingertabelle haben.

Außerdem wurde zum Vergleich noch eine Chord-Version implementiert bei der jeder Knoten alle anderen im Netz kennt. Dies ist der optimale Algorithmus, da jedes Paket nur ein Hop im Overlay-Netzwerk benötigt. Dieser Algorithmus wird als Chord mit globaler Sicht bezeichnet.

3.3.2. Beispiel

Das folgende Beispiel verdeutlicht die Arbeitsweise von Falcon. Sowohl das physikalische als auch das dazugehörige Overlay-Netzwerk sind in Abbildung 5 dargestellt. Das Netzwerk besteht aus 8 Knoten. Knoten 6 und 1 sind zum Beispiel physikalische Nachbarn, jedoch keine Nachbarn im Overlay-Netzwerk. In diesem Beispiel will Knoten 6 den Wert mit dem Schlüssel 4.5 speichern, für den Knoten 5 zuständig ist.

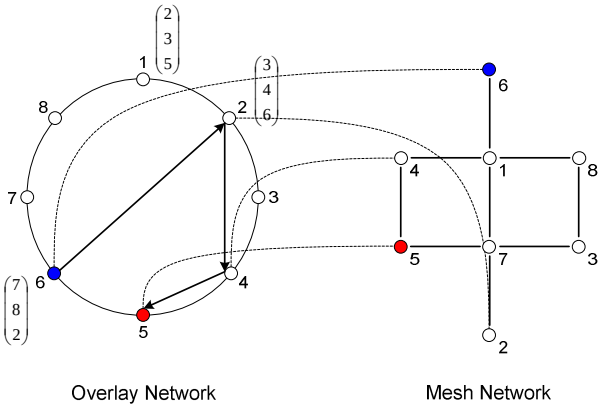

Abbildung 5: Beispiel Overlay-Netz von Chord

Als erstes wird der Fall betrachtet, wenn ein unverändertes Chord zum Einsatz kommt. Knoten 6 ermittelt den Knoten, der für den Wert 4.5 verantwortlich ist. Nach seiner Fingertabelle ist Knoten 2 der dichteste. Knoten 6 wird das Paket an Knoten 2 senden. Im

physikalischen Netz wird das Paket dreimal weitergeleitet (von Knoten 6, 1 und 7). Nun ermittelt Knoten 2 den Knoten 4 als nächsten Empfänger und das Paket wird von Knoten 7 und 1 im physikalischen Netz weitergeleitet. Am Ende ermittelt Knoten 4 die 5 als zuständigen Knoten und schickt das Paket direkt dorthin. Insgesamt sind also 3 Schritte im Overlay-Netzwerk und 7 physikalische Hops nötig.

Basisversion von Falcon

Wie bei Chord ermittelt auch hier Knoten 6 den Knoten 2 als nächsten Hop im Overlay Netzwerk. Im dazugehörigen physikalischen Netz empfängt Knoten 1 das Paket und ermittelt anhand seiner Fingertabelle, dass Knoten 3 dichter am Ziel liegt als Knoten 2. Deshalb wird das Paket dorthin weitergeleitet. Auf dieser Route zu Knoten 3 müssen Knoten 7 und 8 das Paket weiterleitet, die jedoch beide kein dichteres Ziel kennen. Knoten 3 hat in seiner Fingertabelle Knoten 4 und so schickt er das Paket dorthin. In unserem Beispiel sind 3 physikalische Routen möglich: über Knoten 7 und 5, über die Knoten 8 und 1 oder über die Knoten 7 und 1. Im Folgenden wird der erste Fall beschrieben. Das Paket wird dabei von Knoten 7 weitergeleitet und von Knoten 5 empfangen. Dieser erkennt, dass er die Anfrage beantworten kann. Insgesamt sind 5 physikalische Hops nötig und so werden gegenüber Chord 2 Hops gespart.

Erweiterte Version von Falcon

Bei der erweiterten Version von Falcon nutzen die Knoten auch die Fingertabellen ihrer Nachbarn. Mit Hilfe der Fingertabelle von Knoten 1 kann Knoten 6 den Knoten 3 als nächsten Hop im Overlay-Netzwerk ermitteln. Das Paket wird an Knoten 1 übertragen. Knoten 1 entscheidet, dass Knoten 4 dichter am Ziel liegt und sendet das Paket dorthin. Anhand seiner Fingertabelle erkennt Knoten 4, dass Knoten 5 die Anfrage beantworten kann und schickt ihm das Paket zu. Insgesamt werden 3 physikalische Hops benötigt. Chord mit globaler Sicht benötigt dieselbe Anzahl an Hops.

3.4. Dienste auf Basis einer DHT

Die Dienste wie ARP und DHCP benutzen die DHT zum Speichern von Informationen. Dadurch ist bei verteilter Realisierung der Dienste gewährleistet, dass alle beteiligten Knoten eine konsistente Sicht auf die Daten haben. Aufgrund der Redundanz, die die DHT bietet, ist der Dienst zudem robuster und auch beim Ausfall einzelner Knoten funktionsfähig.

3.4.1. DHCP

Mit Hilfe von DHCP ist es möglich Clienten ohne Benutzeraktion automatisch zu konfigurieren. Neben einer IP bekommen die Clienten zusätzliche Informationen wie Netzwerkmaske, Adresse des Internet-Gateways, DNS-Server und WINS-Server. Das Problem bei der Verwendung eines DHCP Servers in einem Community-Netzwerk ist die Notwendigkeit einen Administrator zu haben, der diesen Dienst pflegt. Außerdem ist es durch Probleme wie z.B. der Ausfall von Knoten und Netzwerkpartitionierung nicht möglich einen zuverlässigen und hochverfügbaren Server in einem drahtlosen Maschennetzwerk zu betreiben. Dieser Server wäre ein so genannter Single-Point-of-Failure.

Zur Lösung dieses Problems wurde aus dem zentralen DHCP-Server ein Dienst gemacht, den das gesamte Netz verteilt anbietet. Anstatt die Information über vergebene IP-Adressen lokal zu speichern, wird sie in einer DHT abgelegt. Jede BRN-Knoten tritt gegenüber den Clienten als DHCP-Server auf und nutzt die DHT zur Datenspeicherung. Dadurch ist die Synchronisation zwischen den einzelnen BRN-Knoten sichergestellt, und Probleme mit Inkonsistenz, wie eine mehrfach vergebene IP werden verhindert.

Die Integration des DHCP-Dienstes in die DHT ist wie folgt realisiert. Die Grundidee besteht darin, die verfügbaren IP-Adressen auf einen Hash-Ring wie in Abbildung 6 gezeigt, abzubilden. In diesem Beispiel sind die 254 verfügbaren Adressen äquidistant über den Hash-Ring verteilt. Eine zweite Hashfunktion bildet die Ethernet-Adressen der Knoten auf dem Ring ab, sodass sich beide überlagern. Jeder Knoten der DHT für die IP-Adressen zwischen sich und dem Nachfolger auf dem Hash-Ring zuständig ist. Im Beispiel ist Knoten 1 für alle IP-Adressen zwischen 10.9.0.1 und 10.9.0.63 zuständig, während Knoten 3 den Bereich von 10.9.0.127 bis 10.9.0.190 erhält.

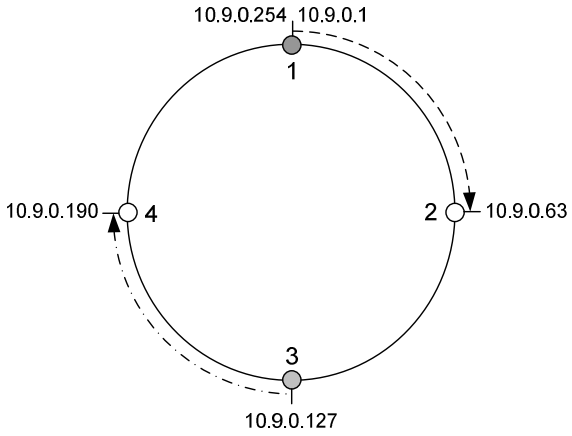

Abbildung 6: IP-Range der Knoten

Der DHCP-Dienst vergibt bei einer Anfrage die IP-Adressen aus seinem Bereich. Nachdem ein Client sich mit einem AP (BRN-Knoten) assoziiert hat, initiiert er ein DHCP-Discover. Der Knoten, mit dem der Client verbunden ist, nimmt aus der DHT eine freie IP (siehe Abbildung 7). Meistens benötigt dies keine Netzwerkkommunikation, da jeder Knoten eine Menge von IPs hat, für die er zuständig ist. Abbildung 6 zeigt ein Beispiel dafür. Ein Client fragt den Knoten 1 nach einer IP. Dieser ist für 10.9.0.1 bis 10.9.0.63 zuständig und kann ohne Kommunikation mit anderen Knoten eine IP aus diesem Bereich vergeben. Sollte jedoch lokal keine IP mehr frei sein, so fragt er benachbarte Knoten nach einer IP.
Wenn der Knoten eine IP hat, antwortet er dem Clienten mit einem DHCP-Offer, worauf dieser ein DHCP-Request mit der ihm angebotenen IP-Adresse sendet. Der BRN-Knoten fügt den Eintrag in die DHT ein und antwortet dem Clienten mit einem DHCP-Acknowledge.

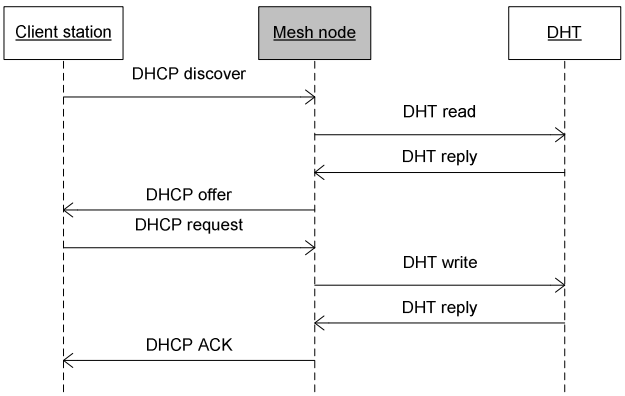

Abbildung 7: Ablauf eines DHCP-Requests

Die vom DHCP-Dienst vergebene IP-Adresse darf vom Clienten nur für eine bestimmte Dauer (Lease-Time) verwendet werden. Während dieser Zeit kann der Client natürlich den AP innerhalb des BRN wechseln. Bevor die Lease-Time abläuft, versucht der Client diese für seine IP zu verlängern. In einem solchen Fall prüft der Access Point mit Hilfe der DHT, ob es dem Client erlaubt ist, die gewünschte IP weiter zu verwenden.
Ein Client kann natürlich auch ohne Ankündigung (z.B. kein IEEE 802.11 Disassoc Paket) verschwinden. Es ist wichtig, dass die vergebenen IP in einem solchen Fall nach Ablauf der Lease-Time wieder verwendet und nicht blockiert werden können. Um dieses zu Problem lösen, speichert jeder Knoten in der DHT den Zeitpunkt, wann die Lease-Time abläuft. Dabei ist das Fehlen einer globalen Uhr in einem solchen Netzwerk ein Problem, da die verschiedenen Uhrzeiten der Knoten zu unterschiedlichen Differenzen zum gegebenen Zeitpunkt (Ende der Lease-Time) führen. Deshalb wird bei Falcon eine Zeit t vor dem Versenden in eine Zeitdifferenz (T – t) umgewandelt (verbleibende Zeit) und mit dieser bei dem Zielknoten wieder die absolute Zeit bestimmt. Dadurch wird keine globale Uhr benötigt.

Um unnötigen Netzwerkverkehr zu verhindern, versucht jeder Dienst seine Information lokal zu speichern, wenn dies möglich ist. Der DHCP-Dienst versucht nur IPs zu vergeben, die auch auf dem Knoten gespeichert werden, um so die Antwortzeiten von DHCP und ARP zu verbessern.

3.4.2. ARP

Das ARP-Protokoll dient zum Auflösen von IP-Adressen, d.h. man ermittelt die zu einer IP gehörende MAC-Adresse. Die ARP-Anfrage wird mittels Broadcast versendet und erreicht so den Zielhost, der seinerseits eine Antwort schickt. Mit Hilfe der DHT wird dieser Broadcast durch ein Unicast ersetzt. Der Knoten mit dem sich der Client assoziiert hat, sendet einen ARP-Broadcast nicht weiter, sondern liest aus dem Paket die IP und ermittelt mit Hilfe der DHT die dazugehörige MAC-Adresse. Er generiert dann eine ARP-Antwort und sendet diese an den Clienten. Bei identischen Anfragen von mehreren Clienten wird die DHT nur einmal befragt, da gleichzeitige Anfragen nach einer IP von dem Knoten zusammengefasst werden. Das ARP-Element, welche diese Funktionalität mit Hilfe von Click realisiert, verfügt außerdem über einen ARP-Cache, in dem er IP- und MAC-Adressen speichern kann, um so Anfragen schneller zu beantworten.

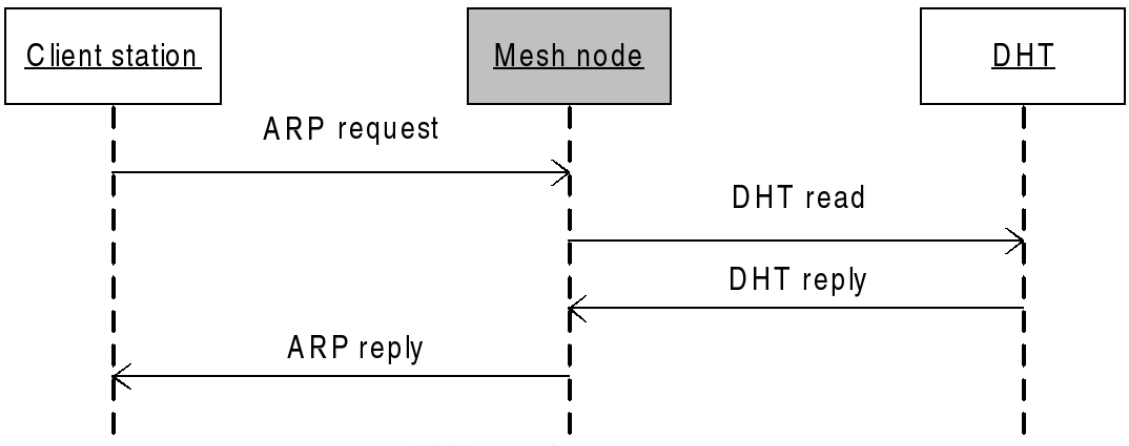

Abbildung 8: Ablauf einer ARP-Anfrage

Die Anfrage nach bestimmten IP-Adressen beantwortet das ARP-Element immer mit seiner eigenen MAC-Adresse. Diese IP-Adressen werden z.B. für den Gateway verwendet, so dass der Client alle Pakete, die ins Internet weitergeleitet werden sollen, direkt an den Access Point schickt. Dieser ermittelt dann ein verfügbaren Internet-Gateway und sendet die Paket dorthin (siehe nächsten Abschnitt).

3.4.3. Internet-Gateway

Die meisten der Benutzer eines Community-Netzwerkes nutzen das Internet. Die Knoten, die Verbindung zum Internet haben, werden als Internet-Gateways bezeichnet. Zwei Protokolle sind wichtig, um das Internet für alle im Maschennetzwerk verfügbar zu machen: Ermittlung und Auswahl von Internet-Gateways.

Als erstes muss ein Knoten herausfinden, ob es Verbindung zum Internet hat. Um auch hier die Selbstorganisation sicherzustellen, muss dies jeder Knoten autonom machen, was sich jedoch sehr schwierig gestaltet. Ein Knoten testet seine Internetverbindung, indem er versucht, über die drahtgebundenen Verbindungen bekannten Rechner (z.B. www.google.de) zu erreichen. Wenn er feststellt, dass er Verbindung zum Internet hat, muss er die anderen Knoten darüber informieren. Bei dem Ansatz, der vom MIT RoofNet und Freifunk verwendet wird, informiert jeder Knoten durch Flooding periodisch seine Nachbarn. Dies benötigt jedoch viele Netzwerkressourcen. Deshalb wird beim BRN die DHT für die Verteilung der Information verwendet. Unter einem bekannten Schlüssel (Gateway-Schlüssel) werden alle Internet-Gateways und eine dazugehörige Metrik, die die Internetverbindungsqualität beschreibt, abgelegt. Der Metrikwert wird aus Verbindungsgeschwindigkeit und Latenz berechnet. Für den Gateway-Schlüssel werden so genannte Subkeys verwendet, die es ermöglichen, unter einem Schlüssel mehrere Gateways und die jeweilige Metrik abzuspeichern. Die für eine Aktualisierung eines Schlüssels nötigen Lese- und Schreiboperation, wurde bei Falcon durch eine Updatefunktion ersetzt, mit der es möglich ist, den Wert eines Subkeys zu ändern oder neue Subkeys mit dem entsprechenden Wert hinzuzufügen, ohne das Risiko haben, durch konkurrierende Schreiboperationen, Daten zu verlieren,.

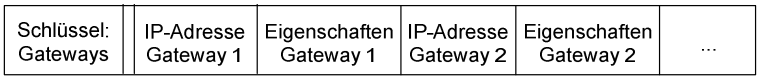

Abbildung 9: Aufbau des Gatewayeintrags

Beim BRN werden alle Pakte von Clienten, deren Zieladresse nicht innerhalb des BRN-Netzes liegen, an einen Internet-Gateway weitergeleitet. Da die Clienten private, im Internet nicht geroutete Adressen erhalten, verwendet der Internet-Gateway „*Network Address Translation*“ (NAT). BRN-Knoten, die eine Internetverbindung brauchen, benutzen das Internet-Gateway-Selection-Protokoll. Sie fragen die DHT nach dem Gateway-Schlüssel und erhalten eine Liste aller verfügbaren Gateways und ihre jeweiligen Eigenschaften. Anhand der Metriken und der Route zu ihnen, bestimmt der Knoten den besten. Die Routingmetrik fließt dabei stärker in die Bewertung ein, um so die Netzlast im BRN gering zu halten. Ein Internet-Gateway wird solange verwendet, wie er erreichbar ist, seine Metrik nicht unter einen bestimmten Schwellwert fällt und keiner mit einer besseren Metrik verfügbar ist. Für existierende TCP-Verbindungen ändert sich der Gateway nicht, um diese nicht zu unterbrechen.

3.4.4. Weitere Dienste

Es ist möglich auch andere Dienste wie DNS und SIP mit Hilfe der DHT zu realisieren. Im Falle von DNS sind die Domain-Namen die Schlüssel und die IP-Adressen die Werte für die DHT. Um Rückwärtsauflösung zu ermöglichen, wird unter der IP zusätzlich der Name abgelegt. Die Kombination von DHCP und DNS ist ebenfalls ein nützlicher Dienst für selbstorganisierende Community-Netze, da so die Rechner unter Domainnamen erreichbar sind, die der Nutzer selbst wählen kann (z.B. meinrechner.berlinroofnet.de).

4. Implementierung

Die verteilte Hashtabelle und die Dienste wurden für das Click-Framework implementiert, da es mit diesem möglich ist, die Software sowohl im Simulator als auch auf realer Hardware (WGT634U) zu testen. Zu den implementierten Elementen gehören:

- Falcon (DHT)
- DHCP-Server
- ARP-Dienst
- DHCP-Client
- ARP-Client
- Flooding

4.1. Click

Click ist ein vom MIT entwickelter, modularer Software-Router. Mit Hilfe der umfangreichen Click-API lassen sich schnell neue Router-Konfigurationen entwickeln. Der Click-Router besteht aus Modulen, so genannten Elementen, welche Pakete bearbeiten. Die einzelnen Elemente implementieren einfache Router-Funktionen wie Klassifizierung, Warteschlangen, Scheduler und Schnittstellen zu Netzwerkgeräten. Als Konfiguration dient ein Graph, der aus einzelnen, miteinander verbundenen Elementen besteht. Die Pakete bewegen sich dabei entlang der Kanten zwischen den Elementen. Verschiedene Features einzelner Elemente ermöglichen so große und komplexe Konfigurationen.
Die einzelnen Elemente sind C++-Klassen, die alle von der einer Basis-Klasse abgeleitet sind.

Click lässt sich sowohl als Programm im Userspace als auch als Linux-Kernelmodul verwenden. Letzteres findet im BRN-Projekt Verwendung, da es bessere Performance ermöglicht. Das Userspace-Programm bietet jedoch die Möglichkeit mit Tools wie dem GNU-Debugger und Valgrind das Programm auf Fehler zu untersuchen und kam deshalb bei der Entwicklung ebenfalls zum Einsatz.

Für den Simulator NS2 existiert eine Erweiterung (NSClick), die es möglich macht, die Elemente in der Simulation zu testen. Dies bietet große Vorteile bei der Entwicklung, da das Debuggen vereinfacht wird und bei mehreren Tests immer die gleichen Testbedingungen vorliegen. Zusätzlich werden Daten wie Anzahl versendeter Pakete und Kollisionen protokolliert und man kann mit Hilfe dieser Informationen die verschiedenen Ansätze direkt miteinander vergleichen.

4.2. Falcon-DHT

Die Falcon-DHT ist die Grundlage der Dienste DHCP und ARP. Sie besteht aus einem Click-Element mit 3 Ein- und 3 Ausgängen.

Eingangsports:
Port 0: DHT
Port 1: Dienste (DHCP, ARP, …)

Port 2: DSR

Ausgangsports:
Port 0: DHT
Port 1: Dienste (DHCP, ARP, ...)
Port 2: DSR

Die DHT-Knoten kommunizieren über den Port 0. Die Anfragen der Dienste (DHCP, ARP) erhält die DHT über Port 1 und die Antworten werden über Ausgangsport 1 zurückgesendet. Mit Hilfe der Ein- und Ausgangsport 2 wird die Verbesserung der DHT realisiert. DHT-Pakete, die ein Knoten erhält, die aber nicht direkt für ihn bestimmt sind, werden dort übergeben und inspiziert. Sollte der Knoten eine bessere Route zum Ziel des Paketes kennen, so wird es umgeleitet.
Die genaue Funktionsweise der DHT, die möglichen Operationen und ihre Realisierung sind Thema einer weiteren Arbeit und werden hier nicht erläutert.

DHT-Protokoll
Das DHT-Protokoll dient zur Kommunikation sowohl zwischen den Knoten als auch der Dienste mit der DHT. Ein Paket besteht aus einem 14 Byte langen Header und einem variablen Teil (Payload).

Aufbau des Headers:

- **Sender :** Absender-Dienst des Paketes
- **Empfänger :** Empfänger-Dienst des Paketes
- **Sender-Adresse :** Adresse des Absenders
- **Primärer Sender :** Dienst, der das Paket erzeugt hat
- **Flags :** Anzeige von Fehlern
- **Code :** Auszuführender DHT-Befehl
- **ID :** eindeutige ID des Paketes
- Payloadlänge: Länge des Payloads (Schlüssel, Werte, Routinginformation)

Die Felder Sender bzw. Empfänger geben den Dienst an, der dieses Paket abgeschickt hat bzw. empfangen soll. Es wurden dafür den Diensten verschiedenen Werte zugeordnet:

Dienst	ID
DHT	1
DHCP	2
ARP	3
GATEWAY	4

Die Sender-Adresse ist die MAC-Adresse des Absenders. Dadurch ist es möglich, dass der Empfänger die Antwort direkt an den ursprünglichen Sender zurückschicken kann, auch wenn dieser ihm das Paket nicht direkt gesendet hat.
Im Feld Primärer Sender ist der ursprüngliche Dienst vermerkt, der die Anfrage gestellt hat. Dies ist nötigt, da die DHT sich selbst in das Sender-Feld einträgt, wenn sie das Paket zu einem anderen DHT-Knoten weiterleiten muss.
Fehler werden mit Hilfe des Flags-Feld signalisiert. Es werden dabei 2 der 8 Bits verwendetet:

- Bit 0: Status bzw. Ergebnis der DHT-Operation (0: OK 1: Fehler)
- Bit 7: DHT-Daten: Dieses Flag beim Kopieren von Daten zwischen DHT-Knoten gesetzt.

Das Feld „Code“ enthält die DHT-Operation:

Bit	Funktion
0	READ
1	INSERT
2	WRITE
3	REMOVE
4	LOCK
5	UNLOCK
6	DHT_DATA
7	ROUTING

Die ID ist eine vom Dienst frei wählbare Zahl, die von der DHT nicht verändert wird. Sie soll es ermöglichen, Antworten von der DHT den entsprechenden Anfragen schneller zu zuordnen. Die Länge des Payloads (Schlüssel und Werte, Routing-Information der DHT, Daten der DHT) werden im Feld Payloadlänge angegeben.

Bei einer Anfrage wird in dem Feld Sender die Nummer des Dienstes (z.B. für DHCP die 2) eingetragen und als Empfänger die DHT angegeben. Mit Hilfe des Feldes Code wird die Operation festgelegt. Der Schlüssel und eventuelle Werte befinden sich im Payload. Die ID wird ebenfalls vom Dienst gewählt. Dieses Paket wird nun an Port 1 der DHT übergeben.
In der DHT wird zuerst der Schlüssel aus dem Payload genommen und anhand der Fingertabelle überprüft, ob der Knoten selbst oder ein anderer dafür zuständig ist. Im letzten Fall wird das Paket über Port 0 an einen entsprechenden anderen Knoten gesendet und im Feld „Primärer Sender“ wird der Absender vermerkt und das Feld „Sender“ enthält dann DHT. So kann der Empfänger erkennen, dass es sich um ein weitergeleitetes Paket handelt.
Der Knoten, der für den Schlüssel zuständig ist, liest aus dem Feld „Code“ die Operation und im Fall einer Schreiboperation die Werte aus dem Payload.
Aus dem Ergebnis erzeugt er wieder ein DHT-Paket bei dem nun Sender und Empfänger des Anfragepakets vertauscht werden bzw. der Empfänger nun jener ist, der im Feld „Primärer Sender“ vermerkt ist (weitergeleitetes Paket). Im Flag-Feld wird der Status der Operation hinterlegt. Das Paket wird nun an die Absenderadresse zurückgesendet.

4.3. DHCP-Server

Der DHCP-Server besitzt 2 Eingangs- und 2 Ausgangsports. Die DHCP-Pakete der Clienten empfängt er über Eingangsport 0, die Antworten sendet er über Ausgangsport 0 zurück. Mit der DHT kommuniziert er über Ein- und Ausgangsport 1.

Das Element hat folgende Konfigurationsparameter:

- MAC-Adresse des Knoten
- Netzwerkadresse
- Subnetzmaske
- IP des Routers
- IP-Adresse des DHCP-Servers
- IP-Adresse des DNS-Servers
- Name des DHCP-Servers
- Domain-Name

Der DHCP-Server muss für die verschiedenen Anfragen der Clienten (DHCP-Discover, DHCP-Request, DHCP-Release) unterschiedliche DHT-Operationen ausführen und dann anhand der Antworten entscheiden, ob er dem Clienten eine positive (ACK) oder negative

Antwort (NACK) zurücksendet.

DHCP-Discover
Beim Empfangen einer DHCP-Discover-Nachricht überprüft der DHCP-Server, ob der Client eine bestimmte IP wünscht. Sollte dies nicht der Fall sein, so bestimmt er eine zufällige IP. Im zweiten Schritt fragt er die DHT, ob die IP bereits vergeben ist. Er benutzt ein DHT-Read und wertet die Antwort aus. Sollte die IP bereits vergeben sein, so wählt der DHCP-Server wählt eine neue IP und startet erneut eine Anfrage an die DHT.
Wenn eine freie IP gefunden wurde, so wird ein DHCP-Offer-Paket an den Clienten gesendet.

DHCP-Request
Bei einem DHCP-Request prüft der DHCP-Server zuerst, ob die gewünschte IP tatsächlich unbenutzt ist. Dies wird mit Hilfe einer Lese-Operation bei der DHT festgestellt. Sollte die IP durch einen anderen Clienten belegt sein, so sendet der DHCP-Server ein DHCP-NACK.
Mit einer Schreiboperation wird der neue Client für die IP eingetragen, wenn sie noch frei ist. Dabei wird auch die Lease-Time gesetzt. Ist die Schreiboperation erfolgreich, so erhält der DHCP-Client ein ACK und kann die IP verwenden. Beim Fehlschlagen der Schreiboperation, erhält der Client ein DHCP-NACK und er muss erneut nach einer IP-Adresse fragen.

DHCP-Release
Beim Empfangen einer DHCP-Release Nachricht, entfernt der DHCP-Server mit Hilfe einer Remove-Operation den Schlüssel (IP-Adresse) aus der DHT. Der Client erhält keine Rückmeldung.

4.4. ARP-Dienst
Der ARP-Dienst besitzt 2 Eingangs- und 2 Ausgangsports. Die ARP-Request der Clienten empfängt er über Eingangsport 0 und die Antworten sendet er über Ausgangsport 0 zurück. Er kommuniziert mit der DHT über Ein- und Ausgangsport 1.

Das Element hat folgende Konfigurationsparameter:

- **IPAddress :** IP des virtuellen Servers (Gateway, DNS, DHCP,....).
- **MacAddress :** MAC-Adresse, welche zurückgesendet wird, wenn ein Client nach der IP des Routers fragt.

Aus einem empfangenen ARP-Request ermittelt der ARP-Diesnt die IP-Adresse und vergleicht sie zuerst mit der IP-Adresse des Routers (siehe Konfiguration). Sollten beide identisch sein, so sendet das Element eine ARP-Antwort mit der MAC-Adresse des Routers
Bei anderen IP-Adressen, prüft der ARP-Dienst, ob er mit Hilfe seines Caches die Anfrage beantworten kann. Im Cache befinden sich die Ergebnisse der letzten ARP-Anfragen. Sollte dies nicht erfolgreich sein, so wird überprüft, ob im Moment schon eine DHT-Anfrage zu dieser IP (Schlüssel) gestellt wird. Dadurch wird verhindert, dass mehrere Anfragen der Clienten zu einer IP zu mehren DHT-Anfragen und somit zu unnötiger Netzlast führen. Eine neue DHT-Anfrage wird also erst dann vorgenommen, wenn die Antwort nicht im Cache zu finden ist und gerade keine DHT-Anfrage zu dieser IP gestellt wird.
Die DHT-Anfrage ist eine Leseoperation, bei der die IP als Schlüssel dient. In der Antwort der DHT steht neben der IP auch die gesuchte MAC-Adresse, wenn das Lesen erfolgreich war. Sollte es keine MAC-Adresse zu der gesuchten IP geben, so wird mit Hilfe des Flags im DHT-Paket der Fehler angezeigt. Der ARP-Dienst verwirft in einem solchen Fall den ARP-Request des Clienten. Sollte das Lesen erfolgreich sein, so sendet der ARP-Dienst eine ARP-Antwort mit der entsprechenden MAC-Adresse zurück.

4.5. DHCP-Client

Das Element *DHCP-Client* simuliert beliebig viele DHCP-Clienten und testet dabei die verschiedenen Funktionen (Discover, Request, ...). Es ist dabei möglich auch gezielt bestimmte IP zu benutzen. Es hat sowohl einen Eingangs- als auch einen Ausgangsport.

Das Element hat folgende Konfigurationsparameter:

- **StartMacAddresse:** Erste MAC-Adresse, die bei den Anfragen benutzt wird. Für jede weitere Anfrage wird diese Adresse um eins erhöht.
- **StartIPAddresse:** Erste IP-Adresse, die bei den Anfragen benutzt wird. Für jede weitere Anfrage wird diese Adresse um eins erhöht.
- **IPRange:** Anzahl der IP-Adressen, die per DHCP-Request angefragt werden sollen.
- **Start:** Zeitpunkt, bei dem die erste Anfrage gemacht werden soll.
- **Pause:** Zeitdauer zwischen 2 verschiedenen DHCP-Anfragen

Nach der angegebenen Startzeit (in Sekunden) startet das Element. Diese Verzögerung ist nötig, um der DHT ausreichen Zeit zu geben, um die Fingertabellen aufzubauen. Insgesamt holt es sich so viele IP-Adressen wie bei der Konfiguration angegeben wurden (IPRange). Um sich gegenüber dem DHCP-Server immer als neuer Client auszugeben, wird bei jeder Anfrage eine andere MAC-Adresse verwendet. Die erste wird mit dem Parameter *StartMacAddress* festgelegt. Die erste IP-Adresse, die der Client holen und in der DHCP-Option „Requested IP“ schreiben soll, wird mit dem Parameter *StartIPAddress* festgelegt.

Ablauf einer DHCP-Anfrage

Der Client sendet ein DHCP-Discover mit der gewählten IP, worauf ihm der DHCP-Server mit einem DHCP-Offer antwortet. Die vom Server angebotene IP muss nicht mit der vom Clienten gewünschten übereinstimmen, da diese bereits in Benutzung sein kann und aus diesem Grund nicht vergeben wird. Der Client schickt nun ein DHCP-Request an den Server, bei dem er die IP, die ihm der Server angeboten hat, angibt. Wenn der Client ein DHCP-Ack erhält, speichert er die Zeit, die seit dem Senden des DHCP-Discover vergangen ist. Sollte der Server ein DHCP-NACK senden, speichert der Client diese Anfrage als unbeantwortet und beginnt die nächste. Da der Client sowohl die Fehler als auch die Zeit für erfolgreiche DHCP-Anfragen speichert, kann man so Aussagen über Zuverlässigkeit des Dienstes und durchschnittliche Dauer einer DHCP-Anfrage machen.

4.6. ARP-Client

Das Click-Element *ARP-Client* dient zum Testen des ARP-Dienstes. Es erzeugt ARP-Anfragen und wertet die Antworten und deren Dauer aus. Das Element besitzt einen Eingangs- und einen Ausgangsport.

Es hat folgende Konfigurationsparameter:

- **ClientIPAdresse:** IP-Adresse des simulierten Clienten.
- **ClientMacAddresse:** Mac-Adresse des simulierten Clienten.
- **StartIPAddress:** Erste IP-Adresse.
- **IPRange:** Anzahl der IP-Adressen.
- **Start:** Zeitpunkt der ersten Anfrage.
- **IntervalTime:** Zeit zwischen zwei Anfragen.
- **Count:** Anzahl der Anfragen insgesamt. Bei jeder dieser Anfragen wird zufällig eine IP-Adresse aus dem angegebenen Bereich gewählt.
- **AtOnce:** Maximale Anzahl unbeantworteter Anfragen

- **Timeout:** Zeit in ms, die der Client auf die Antwort zu einer Anfrage wartet bis die Anfrage als unbeantwortet gilt. Antworten, die nach dieser Zeit kommen, werden verworfen und nicht gewertet.

Das Element simuliert einen Clienten mit einer festgelegten IP- und MAC-Adresse und erzeugt ARP-Anfragen, die er über den Ausgangsport zum ARP-Dienst sendet. Mit Hilfe der Konfigurationsparameter werden der nachgefragte IP-Bereich und die Anzahl der Anfragen festgelegt. Durch den Parameter *AtOnce* kann gesteuert werden, wie viele Anfragen der Client parallel machen darf. So können mehrere Clients simuliert werden. Mit dem Parameter „Timeout“ wird festgelegt, nach welcher Zeit eine ARP-Anfrage als unbeantwortet gewertet werden soll.

Um der DHT genug Zeit zum Aufbau zu geben, kann mit der Startzeit festgelegt werden, wann der Client mit den Anfragen beginnen soll.

4.7. Broadcast

Das Click-Element *Flooding* implementiert ein einfaches Fluten des Netzwerkes. Dabei kommen keine Optimierungen, wie z.B. Byzan, zum Einsatz. Alle Broadcastpakete werden mit einer Nummer versehen und jeder, der ein Paket empfängt, prüft, ob er die Kombination aus Paketnummer und MAC-Adresse des Absenders schon gespeichert hat. Sollte dies nicht der Fall sein, so wird das Paket durch beide Ausgangsport zum lokalen Dienst und zum Weitersenden herausgegeben.

Es besitzt 2 Ein- und 2 Ausgangsports. Am ersten Eingangsport werden Pakete vom lokalen Diensten/Clients entgegengenommen. Über den zweiten Eingangsport werden empfangene Broadcast-Pakete entgegengenommen. Durch den ersten bzw. zweiten Ausgangsport werden die Pakte an den lokalen Dienst bzw. das Netz weitergereicht.

Für die Simulation wurde in dieses Element eine Verzögerung bei Versenden hinzugefügt, da es sonst durch gleichzeitiges Weiterleiten von Broadcastpaketen durch mehrere Knoten, zu sehr vielen Kollisionen kommt. Um die Verzögerung zu steuern, lässt sich neben minimaler und maximaler Verzögerung auch ein Mindestabstand zwischen 2 Pakete angeben. Beim Versenden eines Paketes wird eine zufällige Zeit zwischen minimaler und maximaler Verzögerung bestimmt und solange mit dem Weiterleiten des Paketes gewartet, bis diese Zeit vorbei ist und auch der minimale Abstand zwischen 2 Pakete eingehalten wird.

Das Element hat folgende Konfigurationsparameter:

- **MinimalJitter:** Minimale Zeitdauer in Millisekunden, die bis zur Weitersenden eines Broadcastspaketes gewartet werden soll.
- **MaximalJitter:** Maximale Zeitdauer in Millisekunden, die bis zur Weitersenden eines Broadcastpaketes gewartet werden soll.´
- **MinimaleJitterBetweenPackets:** Minimale Zeitdauer zwischen der Weiterleitung von 2 Paketen.

4.8. Integration ins BRN

Die neuen Elemente wurden in den vorhandenen click-Konfigurationgraphen (Abbildung 10) des BRN eingefügt. Dazu war es nötig, die entsprechenden Pakete (DHT, DHCP, ARP) zu klassifizieren und an die entsprechenden Elemente zu senden. Die neuen Elemente wurden in der Komponente „dhcp_arp_dht“ (Abbildung 10 und 11) zusammengefasst. Diese besitzt 2 Ein- und 3 Ausgänge.

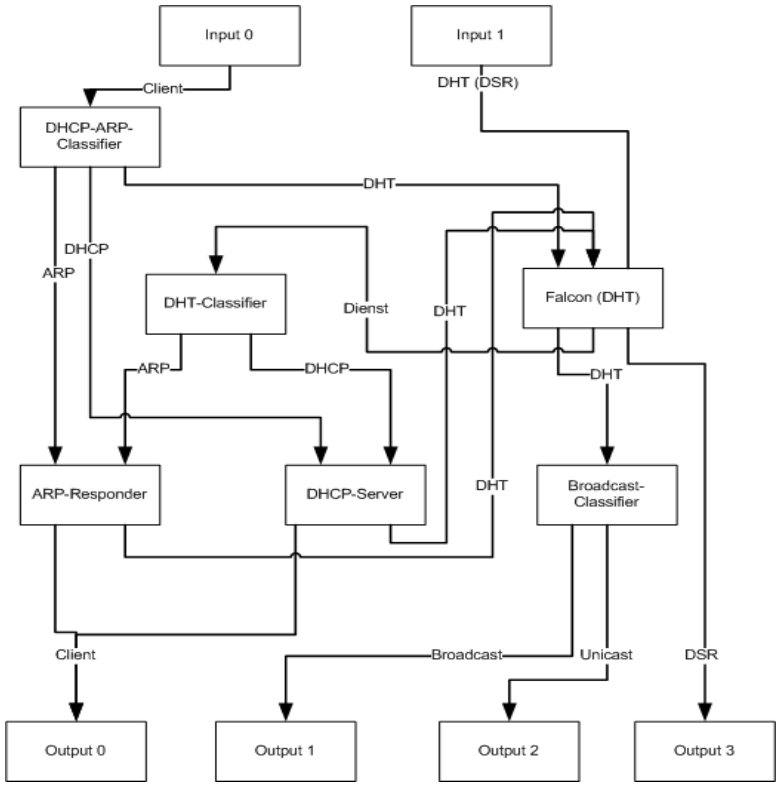

Abbildung 10: Click: DHCP-ARP-Element

Die Ein- und Ausgänge des „dhcp_arp“-Elementes sind mit folgenden anderen Komponenten verbunden:

Eingänge:
0: DHCP-ARP-BRN
1: DSR-Takeout

Ausgänge:
0: Client-Stationen
1: Andere Stationen (Broadcast)
2: DSR [0] (Route-Request)
3: DSR [1] (Routing)

Von den eingehenden Paketen müssen sowohl ARP- und DHCP- als auch DHT-Pakete an die DHCP-ARP-Komponente übergeben werden. Dazu werden die empfangenen Pakete entsprechend klassifiziert.
DHCP-Pakte sind IP-Pakete und die Protokollnummer im MAC-Frame ist „0800“. Es handelt sich dabei außerdem um UDP-Pakete mit dem Zielport 67. DHT-Pakete werden mit Hilfe des BRN-Protokolls verschickt (MAC-Frame-Protokoll: „8086“).

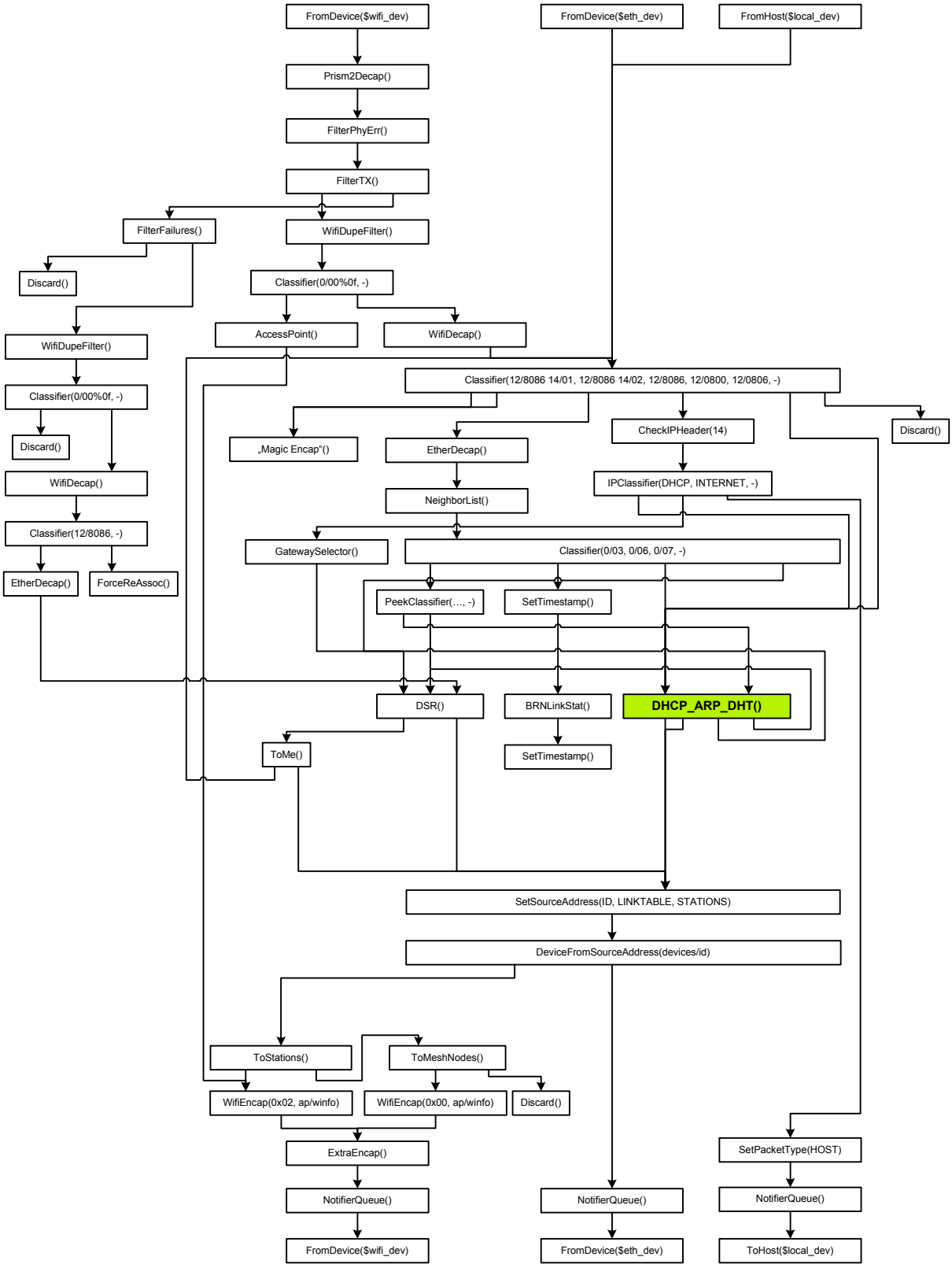

Abbildung 11: Click-Graph

Für die Optimierung (Takeout) wurde direkt vor dem Routing-Element (DSR) ein Klassifizierer eingefügt, der DHT-Pakete herausnimmt und an die DHCP-ARP-Komponente übergibt. Die Pakete, bei denen die DHT keinen besseren Knoten als Ziel finden konnte, werden direkt an wieder an das DSR übergeben. Für Pakete, die jedoch ein neues Ziel haben,

muss jedoch erst eine neue Route bestimmt werden. Deshalb werden sie an Eingangsport 0 von DSR übergeben.

Für die Simulation wurden die beiden Komponenten DHCP- und ARP-Client innerhalb der DHCP-ARP-Komponente direkt mit dem DHCP-Server und dem ARP-Dienst verbunden. Die DHCP- bzw. ARP-Anfragen, die über die Netzwerkgeräte hineinkamen, wurden verworfen. Der DHCP-Server bzw. ARP-dienst kommunizierte nur mit den beiden entsprechenden Komponenten.

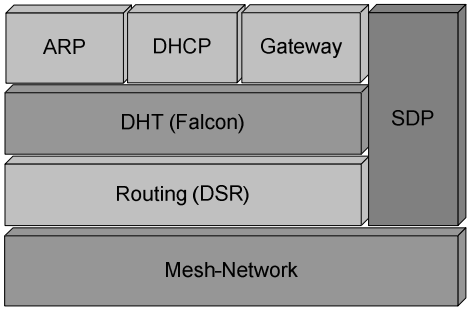

Abbildung 12: Schematischer Aufbau von Click

5. Simulation, Ergebnisse & Diskussion

In diesem Abschnitt wird anhand der Resultate aus Simulationen gezeigt, dass Dienste wie DHCP und ARP viel effizienter mit Netzwerkressourcen umgehen, wenn sie auf Basis einer DHT und nicht mit Hilfe von Broadcastnachrichten (Flooding) realiesiert werden. Für die Simulation des Protokolls wurde der Netzwerksimulator NS2 zusammen mit der Click-API für ns2 (NSClick) verwendet. Ns2 ist ein diskreter Ereignissimulator, der an der Universität von Kalifornien in Berkeley entwickelt wurde. NSClick eine Integration der bekannten Click-API in NS2. Dies hat den großen Vorteil, dass der selbe Code, der für die Simulation verwendet wurde, auch für die Messungen auf realer Hardware, die im nächsten Abschnitt beschrieben werden, genutzt werden konnte.

Für die Simulation des Link-Layers wurde die IEEE 802.11 Medium Access Control (MAC) mit *Distributed Coordination Function* (DCF) verwendet, jedoch ohne RTS/CTS. Die Übertragungsrate betrug 54 Mbit/s und als Radiomodell wurde das „Two-Ray ground“ - Modell des NS2 benutzt.

Die DSR-Implementierung garantiert nur die bestmögliche Verbindung zwischen zwei Endpunkten, aber die DHT benötigt zuverlässige End-zu-End-Kommunikation. Die DHT realisiert diese Zuverlässigkeit mit Hilfe eines Acknowledgement-Mechanismus. Ausgehende Pakte werden in einer extra Warteschlage gespeichert und es wird ein Retransmission-Timer gestartet. Wird bis zum Ablauf der Zeit keine Bestätigung (Antwort-Paket) empfangen, so wird das Paket erneut versendet. Nach einer vorgegebenen Anzahl erfolgloser Retransmissions gilt der Knoten als ausgefallen.

Des Weiteren wurde bei der Simulation der Paketversand der DHT verzögert, um so das Problem der gleichzeitigen Übertragung im Simulator zu reduzieren. Dadurch konnte das Problem der Kollisionen im ns2 besonders für Flooding signifikant reduziert werden.

5.1. Methodik

Ziel der Simulation war es die Performanz und Zuverlässigkeit der Dienste DHCP und ARP auf Basis der DHT zu analysieren und mit dem Ansatz des einfachen Floodings zu vergleichen.

Für verschiedene Netzwerklasten, Anfrage-Pattern und Topologien des Netzwerkes wurden folgende Werte bestimmt:

- Zuverlässigkeit: Wie viele Anfragen wurden in einer vorgegebenen Zeit erfolgreich beantwortet?
- Netzlast: Wie viele Pakete wurden versendet, um die Anfrage zu beantworten?
- Kollisionen: Wie häufig trat ein Paketverlust durch Kollisionen auf?
- Antwortzeit: Wie lange dauerte eine Anfrage?

Es wurden 2 verschiedene Netzwerktopologien verwendet. Die erste ist eine Kette aus 2, 4, 6, 8 und 10 Knoten. Der Abstand zwischen den Knoten betrug 80 m. Die Entfernung wurde so gewählt, dass nur benachbarte Knoten direkt miteinander kommunizieren konnten. Aufgrund des „Hidden Terminal"-Problems ist dieser Fall besonders für das Flooding interessant. Die zweite Topologie ist ein Grid (Quadrat) aus 2, 4, 16, 25, 36, 49, 64 und 121 Knoten und mit einem Abstand von je 55 m.

Es wurden Simulationen mit 1, 4 und 16 Clienten durchgeführt. Zu beginn erhielt jeder Client per DHCP eine IP. Danach wurden 10 zufällige IP-Adressen mit Hilfe von ARP-Anfragen aufgelöst. Die Verzögerung zwischen 2 Anfragen betrug 450 ms und der Timeout lag bei 5 Sekunden. Ein Click-Element simulierte die Clienten, die mit dem Knoten assoziiert waren. Es generierte die Anfragen und gab die Resultate aus.

Die Grundlast des Netzes (Linkprobe etc.), des Setup-Aufwandes sowie der Aufwand zum beantworten der Anfragen wurden extra gemessen, um so die Simulationsresultate der DHT besser mit denen von Flooding vergleichen zu können. Selbst wenn kein Nutz- oder DHT-Datenverkehr existiert, wird durch ETX-linkprobe-Pakete Netzlast erzeugt. Dieses wird als Grundlast bezeichnet. Die Ergebnisse von Flooding und DHT wurden damit korrigiert. Im Gegensatz zu Flooding benötigt die DHT eine Setup-Phase, die in einer zusätzlichen Simulation gemessen wurde, um den Aufwand dafür zu bestimmen. Außerdem wurde die Simulation mit einer DHT wiederholt, die globales Wissen hatte. Dabei kannte jeder Knoten alle anderen im Netz. Dadurch konnte man das Optimum festlegen, welches die untere Grenze für die Simulation und realen Tests war.

5.2. Resultate und Diskussion

In Abbildung 13 und 14 sind die Resultate der Simulation dargestellt. Sie zeigen, dass die Verfügbarkeit beim DHT-Ansatz wesentlich höher ist als bei Flooding. Der Anteil an beantworteten Anfragen sinkt bei Flooding mit zunehmender Anzahl von Knoten sehr stark. So werden bei 10 Knoten und 16 Clienten pro Knoten bei Flooding 91 Prozent der Anfragen beantwortet, während bei Einsatz einer DHT 100 Prozent erreicht werden. Außerdem werden wie in Abbildung 13 zusehen ist, die Anzahl der gesendeten Pakete zur Beantwortung einer ARP-Anfrage gegenüber Flooding reduziert. Ein anderer wichtiger Punkt ist, dass mit wachsender Anzahl von Knoten die Anzahl der Kollisionen bei Flooding sehr stark ansteigt. Zum Beispiel kommt es in einem Netz mit 10 Knoten und 16 Clienten bei Flooding zu 661 Kollisionen, während die DHT nur 16 erzeugt. Ein anderer wichtiger Punkt ist die bessere Antwortzeit der DHT gegenüber Flooding. Das Diagramm 13(d) zeigt die Verteilung der Antwortzeiten für die DHT und Flooding mit 4, 8 und 10 Knoten und jeweils 16 Clienten. Bei

4 Knoten ist die DHT in der Lage alle Anfragen innerhalb von 60 ms zu beantworten, während Flooding 150 ms dafür benötigt. Mit steigender Anzahl von Knoten steigt auch die Antwortzeit, z.B. von 60 ms auf 90 ms bei der DHT, wenn die Anzahl der Knoten von 4 auf 10 steigt. Im Gegensatz dazu werden bei Flooding mit 10 Knoten gar nicht alle Anfragen in der angegeben Zeit beantwortet. Nur 92 % der werden in 250 ms beantwortet.

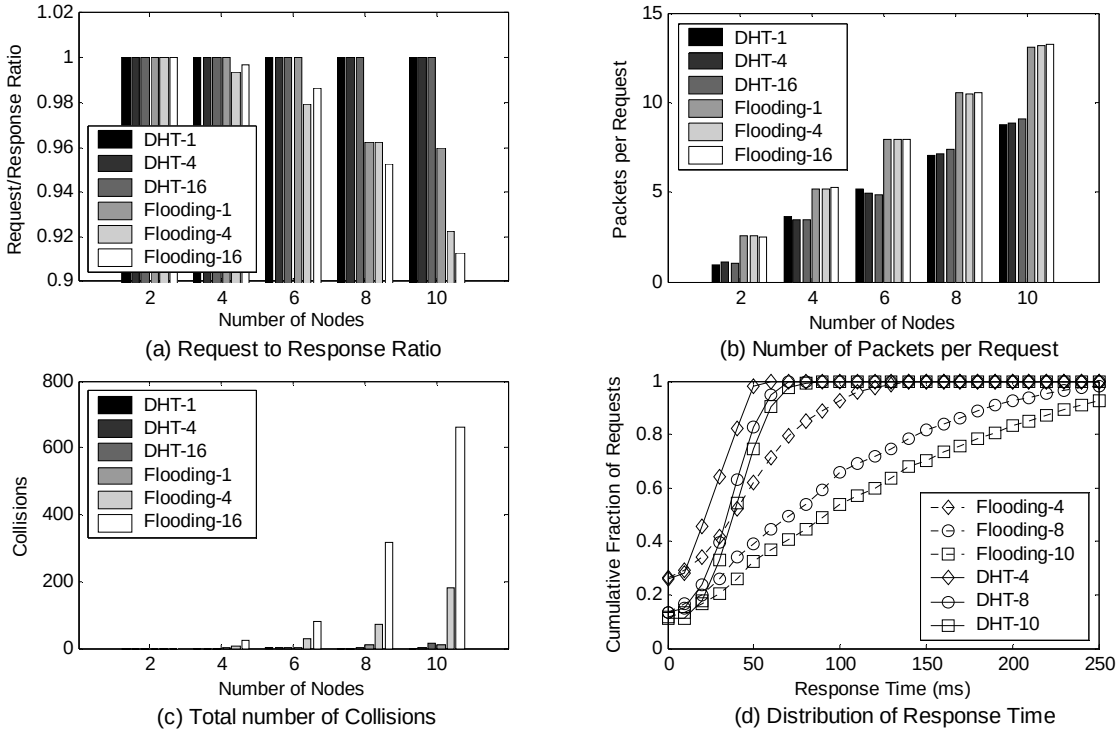

Abbildung 13: Simulationsergebnisse (Kette)

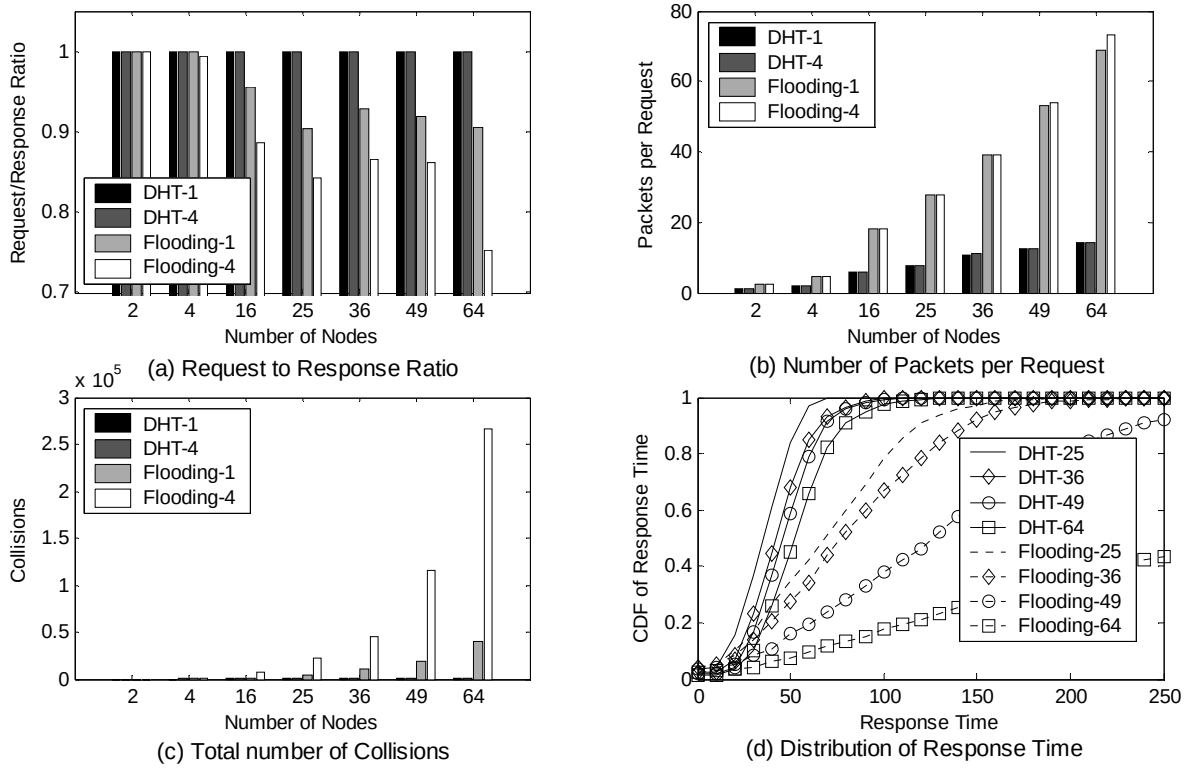

Abbildung 14: Simulationsergebnisse (Grid)

Sie bestätigen die Beobachtung, die bei der Kette gemacht wurden (siehe oben). Der Wert für ARP-Antworten liegt für den DHT-Ansatz für alle Netzgrößen (2 bis 64) bei 100 Prozent, während er beim Flooding sehr stark sinkt. Die DHT benötigt weniger Pakete und verursacht weniger Kollisionen. Auch hier sind die Antwortzeiten bei der DHT deutlich kürzer als beim Flooding.

5.3. Erfahrungen in der realen Testumgebung

Für die praktische Bewertung wurden 20 Knoten der BRN-Testumgebung verwendet. Die Tests in einer realen Umgebung verdeutlichen den praktischen Nutzen von diese vorgestellten Ansatz und bestätigen den Leistungsvorteil der DHT gegenüber Flooding, der schon in der Simulation deutlich wurde.
Es wurden verschiedene Funktions- und Performanztest durchgeführt. Beim Funktionstest wurde überprüft, ob verschiedene Clienten (Betriebssysteme) und Geräte (Labtop, VoIP-Telefon) mit dem Dienst zusammenarbeiten. Beim Performanztest wurden die Zuverlässigkeit und die Geschwindigkeit getestet.

Abbildung 15: WGT 634U

5.3.1. WLAN-Router

Als Knoten des Maschennetzwerkes werden modifizierte Versionen von Netgear's WGT634u Router verwendet. Es handelt sich um einen Wireless Router mit einer MIPS-CPU (BROADCOM) mit 200 MHZ Taktfrequenz, 32 MB RAM, 8 MB Flash, einer Atheros AR5213 basierten IEEE 802.11b/g WiFi Karte, einem Ethernetanschluss mit VLAN-Unterstützung und einem USB 2.0 Host-Port. Das wichtigste Merkmal dieses Gerätes ist die Atheros-basierte WiFi-Karte, für welche Open-Source-Treiber erhältlich sind.

Auf den Geräten läuft ein angepasstes Linux (Version 2.6). Außerdem kommt die Click-API des MIT als Basis für die Routing-Protokolle und Dienste zum Einsatz. Alle Dienste wie ARP, DHCP, Internet-Gateway und die Routing-Protokolle laufen im Kernelmodus des Linux-Systems. Dienste wie ARP, DHCP und Gateway-Protokoll benutzen die Falcon-DHT, die ihrerseits auf das Routing (DSR) aufsetzt.

5.3.2. Funktionstest

Die Studenten, welche das BRN-Netz benutzen, verwenden verschiedene Clienten. Es kommen nicht nur Notebooks mit Windows, Linux und Mac OS zum Einsatz, sondern auch Embedded-Geräte wie VoIP-Telefone und PDAs. Es wurden verschiedene Test durchgeführt und einige interessante Dinge festgestellt. Linux und Windows z.B. erneuern ihre IP auf verschiedene Art. Nach einem Reboot sendet ein Linux-System ein DHCP-Request, wobei in

dem Feld „Requested IP“ die vorherige IP angegeben ist. Der DHCP-Server kann mit einem ACK zu bestätigen. Bei Erhalt einer negativen Antwort, startet der Client die DHCP-Anfrage von vorne und es wird ein DHCP-Discover gesendet. Windows-Stationen senden immer ein DHCP-Discover und setzen dort in dem Feld „Requested IP“ ihre alte IP ein. Der DHCP-Server kann dies berücksichtigen, wenn er ihm mit einem DHCP-Offer eine neue IP anbietet. Er kann dem Client jedoch auch eine andere IP anbieten, wenn die vom Client gewünschte schon vergeben ist.

5.3.3. Vergleich

Um die Ergebnisse der Simulation zu überprüfen, wurden verschiedene Versuche durchgeführt. Für die Testläufe wurden 20 Knoten benutzt, die sich in 2 Gebäuden befinden und den größten Teil des Maschennetzwerkes bilden. Ein Client fragte an verschiedenen Stellen alle IPs ab, die in der DHT gespeichert waren. Das Resultat zeigt Abbildung 15.

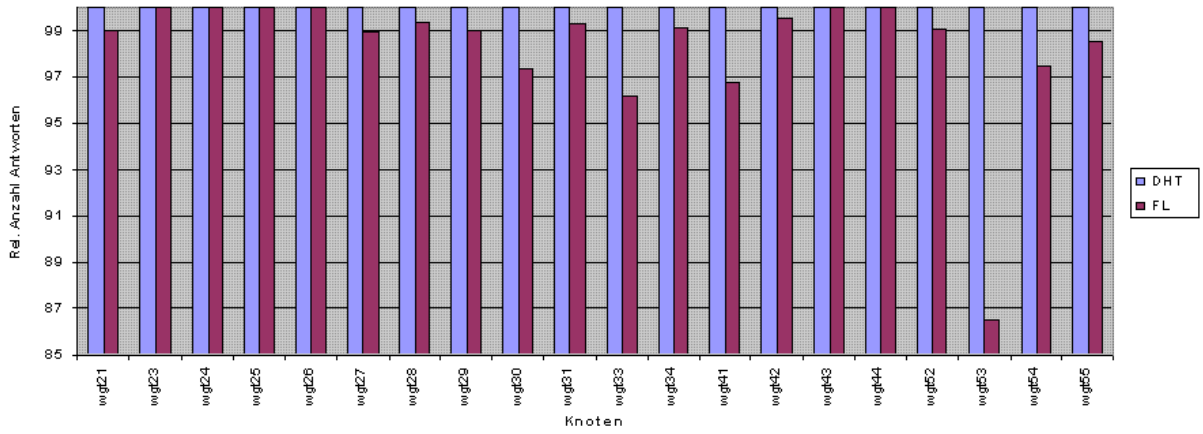

Abbildung 16: Ergebnisse vom Indoortest

Die X-Achse zeigt die Knoten mit dem sich der Client assoziiert hatte. Die Abbildung 15 zeigt das bei Flooding (FL) je nach Position des Clienten 87 bis 100 % der Anfragen beantwortet wurden, während bei der DHT immer 100 % beantwortet wurden. Die Resultate zeigen, dass es möglich und sinnvoll ist, Dienste wie DHCP und ARP verteilt und selbstorganisierend zu realisieren. Durch Einsatz von Unicast statt netzweites Flooding kann die Verfügbarkeit der Dienste verbessert werden.

6. Zusammenfassung und Ausblick

Es wurden in dieser Arbeit Dienste (DHCP, ARP, Internet Gateway) und ihre Umsetzung in Community-Netzwerken vorgestellt, wobei dabei die Selbstorganisation besonders hervorgehoben wurde. Außerdem konnte mit Hilfe der Simulation und der Test in einem kleinen Maschennetzwerk gezeigt werden, dass der Einsatz einer DHT als Basis von Diensten wie DHCP und ARP sinnvoll ist und gegenüber ineffizienten Algorithmen wie Broadcast Vorteile bei der Verfügbarkeit, Effizienz und Leistung der Dienste bringt.
Die weiteren Arbeiten werden sich mit der Realisierung anderer Dienste wie DNS und SIP und der Verbesserung der DHT, wie effiziente Erkennung von Ausfällen von Knoten und Verbesserung der Verfügbarkeit durch Redundanz, befassen.

7. Literatur

[1] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level Measurements from an 802.11b Mesh Network. SIGCOMM 2004, 2004.

[2] Smoot Carl-Mitchell and John S. Quarterman. Using ARP to Implement Transparent Subnet Gateways. RFC1027, Network Working Group, 1987.

[3] D. Couto. High-throughput routing for multi-hop wireless networks, 2004.

[4] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multiradio, multi-hop wireless mesh networks, 2004.

[5] M. Hattig. ZeroConf Requirements. Internet Draft. Zeroconf WG, 2001.

[6] Freifunk community network in Berlin, http://freifunk.net/

[7] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, Mobile Computing, volume 353. Kluwer Academic Publishers, 1996.

[8] Gautam Kulkarni, Alok Nandan, Mario Gerla, and Mani Srivastava. A radio aware routing protocol for wireless mesh networks, 2005.

[9] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In Mobile Computing and Networking, pages 61–69, 2001.

[10] Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. The click modular router. In Symposium on Operating Systems Principles, pages 217–231, 1999.

[11] Sze-Yao Ni, Yu-Chee Tseng, Yuh- Shyan Chen, and Jang-Ping Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. Department of Computer Science and Information Engineering National Central University, Chung-Li, 32054, Taiwan, 1999.

[12] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. pages 149–160.

[13] Kevin Fall and Kannan Varadhan, editors. *ns* notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. http://www-mash.cs.berkeley.edu/ns/.

[14] Dr. Thomas Eversberg and Dr. Michael Hey, http://www.dlr.de/rd/fachprog/nav/indoor/indoor.html

[15] Nsclick simulation environment, http://systems.cs.colorado.edu/Networking/nsclick/

[16] IEEE 802.11 specification, http://grouper.ieee.org/groups/802/11/

[17] Madwifi, wifi driver, http://madwifi.org/

[18] Filipe Araujo, Luus Rodrigues, Jorg Kaiser, Changling Liu and Carlos Mitidieri. CHR: Distributed Hash Table for Wireless Ad Hoc Networks. http://www.di.fc.ul.pt/~ler/reports/debs05.pdf

[19] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. MobiCom 2000. http://www.eecs.harvard.edu/~htk/publication/2000-mobi-karp-kung.pdf

[20] MIT Roofnet, http://pdos.csail.mit.edu/roofnet/doku.php

[21] Dynamic Source Routing, de.wikipedia.org/wiki/Dynamic_Source_Routing

[22] The ETX Protocol, http://www.cuwin.net/manual/techdocs/etx

[23] WMAN, www.wimaxforum.org

[24] Wireshark, www.wireshark.org

[25] GPS, de.wikipedia.org/wiki/Global_Positioning_System