



Universität Stuttgart

Fakultät Informatik, Elektrotechnik
und Informationstechnik

NEXUS

Fachstudie Nr. XXXX

Hardwareplattformen und Systemsoftware für drahtlose vermaschte Kommunikationsnetze

Alexander Egorenkov
Sergey Telezhnikov
Valeri Schneider

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Kurt Rothermel
Betreuer:	Dipl.-Inf. Lars Geiger
begonnen am:	November 2007
beendet am:	Januar 2008
CR-Nummer:	XXXX



Institut für Parallele
und Verteilte Systeme
Abteilung Verteilte Systeme
Universitätsstraße 38
D-70569 Stuttgart



Abstract

TODO: †LAYOUT

TODO

Mesh-Netze (engl. Wireless Mesh Network, WMN) sind drahtlose Ad-Hoc-Netze, bestehend aus stationären Mesh-Routern, die einen Routing-Backbone bilden, und mobilen oder stationären Mesh-Clients. Die Mesh-Clients kommunizieren über den Backbone mit anderen Mesh-Clients oder erlangen über den Backbone Zugang zum Internet. Mesh-Netze können dabei auch größere Bereiche, beispielsweise ganze Städte abdecken (entsprechende Stadtnetze werden aktuell z.B. durch Google installiert).

Ein entsprechendes Mesh-Netz soll für die Forschungszwecke für den Sonderforschungsbereich (SFB) Nexus an der Universität Stuttgart eingerichtet werden.

Diese Fachstudie befasst sich mit der Ausarbeitung einer Empfehlung für die Beschaffung entsprechender Geräte (Hardwareplattformen und Systemsoftware) für den Aufbau eines WMN.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Grundlagen von WMNs	1
1.1.1	Motivation	1
1.1.2	Drahtloses Ad-Hoc Netzwerk	1
1.1.3	WMN	2
1.1.4	IEEE 802.11	3
1.1.4.1	IEEE 802.11a	4
1.1.4.2	IEEE 802.11b	4
1.1.4.3	IEEE 802.11g	4
1.1.4.4	IEEE 802.11n	5
1.1.4.5	IEEE 802.11s	5
1.1.5	Ad-Hoc Routing-Protokolle	5
1.1.5.1	Positionsbasierte Routingverfahren	6
1.1.5.2	Topologiebasierte Routingverfahren	6
1.1.5.3	Proaktive Verfahren	6
1.1.5.4	Reaktive Verfahren	6
1.1.5.5	Hybride Verfahren	7
1.1.5.6	OLSR	7
1.1.5.7	B.A.T.M.A.N.	8
1.2	Existierende Lösungen und Projekte	8
2	Aufgabenstellung	9
3	Anforderungen	9
3.1	Anforderungen an Hardware	10
3.2	Anforderungen an Software	10
3.3	Zusätzliche Anforderungen	10
4	Hardware-Lösungen für den Aufbau eines Mesh-Netzwerkes	12
4.1	PCs + WLAN-Karten	12
4.1.1	PCI-WLAN-Karten	13
4.1.1.1	Linksys WMP55AG	14
4.1.1.2	Netgear WAG311	16
4.1.1.3	D-Link DWL-A520	18
4.1.1.4	Gigabyte GN-WPEAG	20
4.1.1.5	Andere PCI-WLAN-Karten	21
4.1.2	Mini-PCI WLAN-Karten	22
4.1.2.1	Wistron CM9 Atheros AR5213A	23
4.1.2.2	Intel PRO/Wireless 3945	25
4.1.2.3	Intel PRO/Wireless 2915	27
4.1.2.4	Intel Wireless WiFi Link 4965AGN	29
4.1.3	PCMCIA WLAN-Karten	31

4.1.3.1	Proxim Orinoco Gold 8480-WD	32
4.1.3.2	Netgear WAG511	34
4.1.3.3	SMC 2536W-AG	36
4.1.3.4	Linksys WPC55AG	38
4.1.3.5	Andere PCMCIA-WLAN-Karten	39
4.2	WLAN-Router	41
4.2.1	SoHo-Router	41
4.2.1.1	Linksys WRT54G v1.0	43
4.2.1.2	Linksys WRT55AG	45
4.2.1.3	Asus WL500G/GP	47
4.2.1.4	Andere WLAN-Router	48
4.2.2	Professionelle Router	49
4.2.3	Access Points	50
4.3	PDAs und Handys	51
5	Systemsoftware für Mesh-Netzwerk	53
5.1	Linux MadWiFi-Treiber	53
5.2	OpenWRT	54
5.3	Mesh-Routing Software	55
5.3.1	olsr.org OLSR daemon	55
5.3.2	Open-Mesh B.A.T.M.A.N. daemon	56
5.3.3	Meshcom Driver	56
6	Test	58
6.1	Hardware	58
6.2	Software	59
6.2.1	Betriebssystem	59
6.2.2	Treiber für WLAN-Karten	59
6.2.3	olsrd	60
6.2.4	Visualisierung	62
6.2.5	dhcpcd	62
6.2.6	Firewall	63
6.3	Inbetriebnahme	64
6.4	Ergebnisse	64
6.4.1	Topologie	64
6.4.2	Routing-Tabelle	65
7	Fazit	67
7.1	Übersicht	67
7.2	Empfehlung für ein geeignetes WMN	67

8	Anhang	70
8.1	olsrd.conf	70
8.2	olsrd Startup-Skript	77
8.3	topology.pl	79
9	Erklärung	83

Abbildungsverzeichnis

1	Linksys WMP55AG	14
2	Netgear WAG311	16
3	D-Link DWL-A520	18
4	Gigabyte GN-WPEAG	20
5	Wistron CM9 Atheros AR5213A	23
6	Intel PRO/Wireless 3945	25
7	Intel PRO/Wireless 2915	27
8	Intel Wireless WiFi Link 4965AGN	29
9	Proxim Orinoco Gold 8480-WD	32
10	Netgear WAG511	34
11	SMC 2536W-AG	36
12	Linksys WPC55AG	38
13	Linksys WRT54G v1.0	43
14	Linksys WRT55AG	45
15	Asus WL500G/GP	47
16	Topologie	58
17	Topologie dargestellt mit dem olsr.org Plugin dot_draw	65
18	Routing-Tabelle eines Knotens dargestellt mit dem olsr.org Plugin httpinfo	66
20	Übersicht und Bewertung von Hardware	68
22	Übersicht und Bewertung von Software	69

TODO: †Abkuerzungsverzeichniss

TODO

1 Einleitung

In diesem Abschnitt werden einige wichtige Begriffe, die im Laufe des Dokuments auftauchen werden, erläutert und beschrieben.

1.1 Grundlagen von WMNs

1.1.1 Motivation

Ein drahtloses vermaschtes Netz (engl. Wireless Mesh Network, WMN) besteht aus einer Menge von Knoten, die über drahtlose Kommunikationstechniken wie beispielsweise IEEE 802.11 (1.1.4) Nachrichten austauschen. Die Vermaschung der Knoten ermöglicht dabei nicht nur den Austausch von Nachrichten zwischen unmittelbar benachbarten Knoten, sondern auch die Vermittlung von Nachrichten an entfernte Knoten über mehrere Knoten hinweg. Die Vermittlungsfunktionalität wird dabei oft von dedizierten Vermittlungsknoten (engl. Mesh Router) bereitgestellt, die somit eine drahtlose Kommunikationsinfrastruktur für die Klienten (engl. Mesh Client) bilden. Durch den Einsatz vergleichsweise kostengünstiger Hardwarekomponenten und die Vermaschung der Knoten ermöglichen WMNs die kostengünstige Vernetzung auch größerer Gebiete. Entsprechende Netze werden beispielsweise von Community-Projekten wie dem Freifunk-Projekt (1.2) oder Firmen wie Google (1.2) bereits heute in der Praxis für den Aufbau größerer Netze eingesetzt, um beispielsweise kostengünstige Internetzugänge für Stadtteile oder ganze Städte zu realisieren.

WMNs sind auch für den Sonderforschungsbereich (SFB) Nexus an der Universität Stuttgart [1] von großem Interesse. Im Zentrum der Forschungen des SFB stehen Umgebungsmodelle für mobile kontextbezogene Systeme. Umgebungsmodelle sind digitale Abbilder der physischen Welt, die von kontextbezogenen Systemen genutzt werden, um sich selbständig an die physische Umgebung des Benutzers anzupassen. Ein einfaches Beispiel sind ortsbezogene Anwendungen, die beispielsweise aufgrund der aktuellen geographischen Position eines Geräts automatisch Informationen über nahe Restaurants, Sehenswürdigkeiten, usw. selektieren können. Zur Kommunikation, insbesondere mit mobilen Geräten, werden dabei hybride Systeme betrachtet, in denen sowohl eine infrastrukturbasierte Kommunikation als auch die direkte Ad-hoc-Kommunikation zwischen mobilen Endsystemen möglich ist. Hierbei spielen WMNs als eine spezielle Ausprägung eines hybriden Kommunikationssystems eine wesentliche Rolle.

1.1.2 Drahtloses Ad-Hoc Netzwerk

Ein drahtloses Ad-Hoc Netzwerk ist ein dezentralisiertes drahtloses Netzwerk. Das Netzwerk heisst Ad-Hoc, weil jeder Knoten im Netzwerk Daten von anderen Knoten im Netzwerk

weiterleiten kann, und so wird die Entscheidung, welcher Knoten die Weiterleitung von Daten eines Knotens übernimmt, dynamisch auf der Grundlage der Netzwerkkonnektivität getroffen. Das Weiterleiten von Paketen in einem Netzwerk nennt man Routing. Und im Gegensatz zu Ad-Hoc Netzwerken übernehmen diese Aufgabe in einem verdrahteten Netzwerk, wie z.B. das Internet, nur spezielle Knoten, so genannte Router. Auch in drahtlosen Netzwerken mit Infrastruktur übernehmen diese Aufgabe spezielle Knoten, so genannte Access Points, die die Kommunikation zwischen drahtlosen Knoten verwalten, da in solchen drahtlosen Netzwerken drahtlose Knoten direkt nicht kommunizieren [2].

Der dezentralisierte Charakter der drahtlosen Ad-Hoc Netzwerken macht sie geeignet für verschiedene Anwendungen, wo man sich auf zentralisierte Knoten nicht verlassen kann. Ausserdem verbessert diese Eigenschaft die Skalierbarkeit der drahtlosen Ad-Hoc Netzen im Vergleich zu drahtlosen Netzwerken mit Infrastruktur [2].

Minimale Konfiguration und schnelle Verteilung machen drahtlose Ad-Hoc Netzwerke geeignet für Notsituationen, wie z.B. Naturkatastrophen oder militärische Konflikte. Der Einsatz eines dynamischen und adaptiven Routing-Protokolls macht es möglich, diese Art von Netzwerken schnell zu installieren [2].

Drahtlose Ad-Hoc Netzwerke können in 3 Gruppen unterteilt werden:

- Mobile Ad-Hoc Networks (MANETs)
- Wireless Mesh Networks (WMNs)
- Wireless Sensor Networks (WSNs)

Unsere Fachstudie konzentriert sich nur auf WMNs. Deswegen werden diese Netzwerke im folgenden detaillierter erläutert.

1.1.3 WMN

Ein drahtloses vermaschtes Netzwerk ist ein drahtloses Ad-Hoc Netzwerk, in dem zumindest zwei unterschiedliche Kommunikationspfade zu jedem Knoten in diesem Netzwerk existieren. Dieser Typ von drahtlosen Ad-Hoc Netzwerken ist sehr zuverlässig und hat einen sehr hohen Grad an Redundanz.

WMNs bestehen aus Mesh-Routern und Mesh-Clients. Mesh-Router haben zusätzliche Routing-Funktionalität, um das Mesh-Netzwerk zu realisieren. Um die Flexibilität des WMNs zu erhöhen, werden Mesh-Router meistens mit mehreren drahtlosen Kommunikationsgeräten ausgestattet. Man unterscheidet zwei Arten von Mesh-Clients: Typ 1 und Typ 2. Mesh-Clients von Typ 1 sind mit einer äquivalenten Funktionalität ausgestattet wie auch Mesh-Router. Mesh-Clients von Typ 2 sind Knoten mit einer einfacheren Funktionalität als Mesh-Router ausgestattet [2].

Abhängig von unterschiedlicher Funktionalität von Mesh-Routern und Mesh-Clients, kann man folgende Architekturen von WMNs unterscheiden [2]:

- flache
- hierarchische
- hybride

Flache WMNs unterstützen nur Mesh-Clients von Typ 1, so dass die Funktionen des WMN wie Routing-, Verwaltung- und Sicherheit-Protokolle für beide Mesh-Router und Mesh-Clients gleich sind.

Hierarchische WMNs unterstützen nur Mesh-Clients von Typ 2. Deswegen sind die Funktionen von Mesh-Routern und Mesh-Clients sehr unterschiedlich. Da Mesh-Clients von Typ 2 einige notwendige Funktionen von Mesh-Routern fehlen, müssen Mesh-Router um bestimmte Funktionalität erweitert werden, damit Mesh-Router die Mesh-Clients von Typ 2 unterstützen können.

Hybride WMNs sind verallgemeinerte WMNs und unterstützen Mesh-Clients sowohl von Typ 1 als auch von Typ 2. Und wieder müssen Mesh-Router um bestimmte Funktionen erweitert werden, damit sie die Mesh-Clients von Typ 2 unterstützen können und ihnen einen Zugriffsmechanismus auf das WMN zur Verfügung stellen können.

Die wichtigsten Eigenschaften von WMNs [2]:

- Multi-Hop drahtloses Netzwerk
- Sehr zuverlässig
- Sie haben einen hohen Grad an Redundanz
- Sie sind fähig, sich selbst aufzubauen, zu reparieren und zu verwalten
- kompatibel zu anderen drahtlosen Netzwerken und können mit ihnen interagieren

1.1.4 IEEE 802.11

IEEE 802.11 ist eine Familie von Standards für drahtlose Netzwerke.

Im folgenden werden einige für die Fachstudie wichtige Standards erläutert.

1.1.4.1 IEEE 802.11a

IEEE 802.11a spezifiziert eine Variante der physikalischen Schicht, die im 5 GHz Frequenzband arbeitet und Übertragungsraten bis zu 54 MBit/s ermöglicht.

Durch die Verwendung des 5 GHz Frequenzbereiches kann es zu Störeinflüssen durch andere Geräte, die diese Frequenz benutzen kommen, z.B. Radargeräte und Satellitenübertragungsgeräte. Deswegen wird der 5 GHz Frequenzbereich in Europa stärker als der 2.4 GHz Frequenzbereich reguliert.

Es werden 8 nicht überlappende Kanäle im 5 GHz Frequenzbereich zur Verfügung gestellt. Nicht überlappenden Funkkanälen in einem 802.11a WLAN lassen ca. 3 mal so viele Access-Points auf engen Raum zu wie der 802.11b Standard.

Die zugelassen Sendeleistung durch die ETSI für den 802.11a Standard in Europa beträgt 30mW. Dieser Frequenzbereich ist von der Regtp für Lokale Funknetze am 13. November 2002 gebührenfrei freigegeben worden. Die in Deutschland ist die Nutzung dieser Norm nur auf den Einsatz im Indoorbereich beschränkt.

1.1.4.2 IEEE 802.11b

IEEE 802.11b ist ebenfalls eine alternative Spezifikation der physikalischen Schicht, die mit dem bisher genutzten 2.4 GHz Frequenzband auskommt und Übertragungsraten bis zu 11 MBit/s ermöglicht.

Eine Kompatibilität zwischen dem 802.11a und 802.11b besteht aus dem unterschiedlichen Übertragungsfrequenzen nicht.

Es werden nur 3 nicht überlagernde Kanäle zur Verfügung gestellt, wodurch es bei einer größeren Flächendeckung mit 802.11b Access-Points zu Interferenzen mit auf den selben Kanälen senden Access-Points aus der Nachbarschaft kommt.

Das 2.4 GHz Frequenzband teilen sich unter anderem z. B. Mikrowellenherde und Bluetooth-Geräte welches ebenfalls eventuelle Störungen hervorrufen könnte. Geräte mit dieser Norm dürfen in Deutschland für den Indoor- sowie Outdoorbereich genutzt werden.

1.1.4.3 IEEE 802.11g

IEEE 802.11g spezifiziert eine Variante der physikalischen Schicht, die im 2.4 GHz Frequenzband arbeitet und Übertragungsraten bis zu 54 MBit/s ermöglicht.

Zwischen dem 802.11b und 802.11g Standard besteht völlige Abwärtskompatibilität durch Verwendung der CCK-Modulation (Complementary Code Keying).

Die Nutzung von Geräten mit dieser Norm ist für den Indoor- und Outdoorbereich von

der Regtp genehmigt worden.

1.1.4.4 IEEE 802.11n

IEEE 802.11g spezifiziert eine Variante der physikalischen Schicht, die vorzugsweise im 5 GHz Frequenzband arbeitet, MIMO (Multiple Input Multiple Output) einsetzt und Übertragungsraten bis zu 500 MBit/s ermöglicht.

MIMO sendet mit mehreren Antennen in verschiedene Richtungen und Wege. Der MIMO Empfänger besitzt ebenfalls mehrere Antennen, um die Signale wieder zu empfangen und durch eine digitale Signalaufbereitung wieder zu entschlüsseln. Bei MIMO steigt der Datendurchsatz linear zur verwendeten Antennenanzahl an.

IEEE 802.11n befindet sich noch in Entwicklung.

1.1.4.5 IEEE 802.11s

IEEE 802.11s ist ein Standard für drahtlose vermaschte Netzwerke. Der Standard spezifiziert WMNs auf der Sicherungsschicht (Layer 2).

IEEE 802.11s Standard basiert auf IEEE 802.11a/b/g Standards und kompatibel mit diesen Standards.

1.1.5 Ad-Hoc Routing-Protokolle

Ein Ad-Hoc Netzwerk, beispielsweise ein Sensornetzwerk, besteht aus einer großen Anzahl von Knoten, die sich typischerweise spontan vernetzen müssen. Über Funk können diese Knoten miteinander kommunizieren. Knoten können aber wegen der Leistungsfähigkeit, des Energieverbrauches und auch der Mobilität nicht mit allen anderen Knoten kommunizieren. Routing-Verfahren bieten die Möglichkeit, Kommunikation zwischen zwei Knoten mit Hilfe der anderen Knoten auszuführen, wenn beide Knoten miteinander nicht direkt kommunizieren können. Mit anderen Worten, erlaubt Routing indirekte Kommunikation zwischen Knoten in einem Netzwerk [2].

Es gibt mehr als 70 konkurrierende Entwürfe für das Routing der Pakete durch ein Mobiles Ad-Hoc/Maschennetzwerk. Eine Klassifikation der Routingprotokolle kann durch die Anzahl der Empfänger getroffen werden:

- unicast Routing - Ziel der Datenübertragung ist ein einzelner Knoten
- multicast Routing - Ziel sind mehrere Knoten
- geocast Routing - Ziel sind alle Knoten in einem bestimmten geografischen Bereich

- broadcast Forwarding - Ziel sind alle Knoten in der Reichweite des Senders

Eine andere Möglichkeit der Klassifikation besteht in der Einteilung der Protokolle hinsichtlich des grundsätzlichen Ansatzes. Diese Ansätze werden im Folgenden vorgestellt.

1.1.5.1 Positionsbasierte Routingverfahren

Positionsbasierte Routingverfahren nutzen geodätische Informationen über die genauen Positionen der Knoten. Diese Informationen werden z. B. über GPS-Empfänger gewonnen. Anhand dieser Ortsinformationen lässt sich der kürzeste oder der beste Pfad zwischen Quell- und Zielknoten bestimmen. Ein Beispiel für ein positionsbasiertes Routingprotokoll ist LAR [2].

1.1.5.2 Topologiebasierte Routingverfahren

Die topologiebasierten Routingverfahren kommen ohne geodätische Informationen über die Positionen der Knoten des mobilen Ad-hoc-Netzes aus. Ihnen genügen logische Informationen über die Nachbarschaftsbeziehungen der Knoten, also welche Knoten eine direkte Verbindung haben oder über einen oder mehrere Zwischenknoten (hop) in Verbindung treten können. Diese Nachbarknoten können miteinander kommunizieren. Die topologischen Informationen werden meistens durch den Versand so genannter HELLO-Pakete gewonnen. Je nach Zeitpunkt des Aufbaus der Topologiedatenbasis handelt es sich um proaktives oder reaktives Routing. Ein Beispiel für ein Protokoll aus dieser Klasse ist das Neighbourhood Discovery Protocol (NHDP), das Elemente des Optimized Link State Routing Protocol (OLSR) verwendet [2].

1.1.5.3 Proaktive Verfahren

Proaktive Routingverfahren bestimmen die zu verwendenden Pfade zwischen zwei Knoten bereits, bevor diese für die Übertragung von Nutzdaten benötigt werden. Sollen dann Nutzdaten verschickt werden, so muss nicht auf die Bestimmung des Pfads zum Zielknoten gewartet werden. Nachteilig ist dafür jedoch, dass diese Verfahren auch ohne Verkehr von Nutzdaten viele Kontrollpakete verschicken, um Pfade zu bestimmen, die womöglich später nicht benötigt werden. Ein Beispiel für ein Protokoll aus dieser Klasse ist das Optimized Link State Routing Protocol (OLSR) [2].

1.1.5.4 Reaktive Verfahren

Im Gegensatz zu den proaktiven Verfahren bestimmen reaktive Routingverfahren die benötigten Pfade zwischen zwei Knoten erst, wenn Nutzdaten übertragen werden sollen. Daraus ergibt sich, dass das erste Datenpaket einer Verbindung erst mit Verzögerung

versendet werden kann, da zunächst auf den Abschluss der Routenbestimmung gewartet werden muss. Dafür werden allerdings auch nur Kontrollpakete versendet, wenn Nutzdaten verschickt werden und dies zur Routenbestimmung notwendig ist. Dies schlägt sich positiv im Energieverbrauch der Knoten nieder. Das Protokoll Ad hoc On-Demand Distance Vector (AODV) ist ein Beispiel für ein Protokoll dieser Kategorie [2].

1.1.5.5 Hybride Verfahren

Hybride Verfahren kombinieren proaktive und reaktive Routingverfahren. Dabei soll das Ziel erreicht werden, die Vorteile der beiden Ansätze in einem neuen Routingprotokoll zusammenzufassen. Beispielsweise kann in einem lokal beschränkten Bereich ein proaktives Verfahren eingesetzt werden, während für weiter entfernte Ziele ein reaktives Verfahren eingesetzt wird. Dies vermindert die Belastung des Netzes durch Kontrollpakete, die bei einem rein proaktiven Verfahren über das gesamte Netz versendet würden. Trotzdem stehen für lokale Ziele sofort Pfade zur Verfügung, ohne dass auf deren Bestimmung wie bei einem rein reaktiven Verfahren gewartet werden müsste. ZRP ist ein Routingprotokoll, das diesen Ansatz umsetzt [2].

1.1.5.6 OLSR



Optimized Link State Routing, kurz OLSR, ist ein Routingprotokoll für mobile Ad-hoc-Netze, das eine an die Anforderungen eines mobilen drahtlosen LANs angepasste Version des Link State Routing darstellt. Es wurde von der IETF mit dem RFC 3626 standardisiert. Bei diesem verteilten flexiblen Routingverfahren ist allen Routern die vollständige Netztopologie bekannt, sodass sie von Fall zu Fall den kürzesten Weg zum Ziel festlegen können. Als proaktives Routingprotokoll hält es die dafür benötigten Informationen jederzeit bereit. Ein in Mesh-Netzwerken bekannter Vertreter von LSR ist OLSR von olsr.org. Inzwischen existieren für OLSR spezielle Erweiterungen. Mit der ETX-Erweiterung wird dem Umstand Rechnung getragen, dass Links asymmetrisch sein können. Mit dem Fisheye-Algorithmus ist OLSR auch für größere Netzwerke brauchbar geworden, da Routen zu weiter entfernten Knoten weniger häufig neu berechnet werden. Der entscheidende Nachteil ist aber der trotz Fisheye-Algorithmus noch recht hohe Rechenaufwand von OLSR Routing-Daemon, sobald die Anzahl an Knoten ein gewisses Maß übersteigt.

1.1.5.7 B.A.T.M.A.N.



B.A.T.M.A.N., Better Approach To Mobile Adhoc Networking, ist ein Routing-Protokoll für mobile Ad-Hoc Netzwerke. Dieses Protokoll wurde erfunden und wird entwickelt vom FreiFunk-Projekt. B.A.T.M.A.N. soll das vom FreiFunk-Projekt verwendete OLSR Routing-Protokoll ablösen. Das schnelle Wachstum der Netzwerke vom FreiFunk-Projekt führte schnell zu Skalierbarkeitsproblemen wegen der Schwächen im Design des OLSR Routing-Protokolls. Im Gegensatz zu OLSR Routing-Protokoll versucht B.A.T.M.A.N. nicht, alle Routen voraus zu berechnen. Jeder Knoten kennt nur seine direkte Nachbarn und berechnet keine kompletten Routen. Es werden keine Routing-Informationen im Netzwerk geflutet wie bei OLSR. B.A.T.M.A.N. soll das Skalierbarkeitsproblem von OLSR lösen. [3, 4].

1.2 Existierende Lösungen und Projekte

FreiFunk hat zum Ziel, freie, unabhängige und nichtkommerzielle Computer-Funknetze zu etablieren. Es bildet eine Plattform für Menschen, die an einer offenen Netzwerk-Infrastruktur interessiert sind.

<http://wiki.freifunk.net/Hauptseite>

OpenNet hat sich zur Aufgabe gemacht, freie und offene Kommunikationsinfrastrukturen zu fördern. Dabei setzen die Vereinsmitglieder auf WLAN-Technik und die Vernetzung von Dach zu Dach und Haus zu Haus.

<http://wiki.opennet-initiative.de/index.php/Hauptseite>

UMIC-Mesh ist ein hybrides Testbed für WMNs. Das Projekt verfolgt 2 Ziele, einerseits ein großes und skalierbares Ad-Hoc Mesh-Netzwerk für Forschung bereitzustellen und andererseits allen Studenten und Mitarbeiter der Computer Science Abteilung einen breitbandigen Zugang zum Netzwerk der Abteilung zur Verfügung zu stellen.

<http://umic-mesh.net/>

Google WiFi ist ein freies WMN, das von Google finanziert wird und zur Zeit in Mountain View in Kalifornien eingesetzt wird.

<http://wifi.google.com/>

2 Aufgabenstellung

Für Forschungszwecke soll innerhalb des SFB Nexus [1] ein WMN installiert werden.

Dieses WMN dient

- einerseits Nexus-Anwendungen, insbesondere Anwendungen auf mobilen Geräten, als *Kommunikationsmedium*.
- andererseits auch als *Testbed* zur Erforschung verschiedener Erweiterungen von WMNs.

Dieses WMN soll beispielsweise der Untersuchung neuartiger kontextbezogener Kommunikationsmechanismen, der Erforschung von Publish/Subscribe-Diensten für WMNs oder der Verwaltung von Umgebungsmodellen innerhalb eines hybriden Systems wie es ein WMN darstellt, dienen.

Ziel dieser Fachstudie ist die Ausarbeitung einer Empfehlung für die Beschaffung entsprechender Geräte (*Hardwareplattformen und Systemsoftware*) für den Aufbau eines WMN.

Das Vorgehen umfasst im einzelnen:

- Einarbeitung in grundlegende WMN-Technologien
- Analyse der Anforderungen des Nexus-Projektes an ein WMN
- Erstellung einer Übersicht über aktuelle verfügbare Hardwareplattformen und Systemsoftware für WMN
- Bewertung der analysierten Systeme hinsichtlich der ermittelten Anforderungen
- Ausarbeitung einer Empfehlung für eines geeigneten WMN hinsichtlich Hardwareplattform und Systemsoftware

3 Anforderungen

Nach der Einarbeitung in WMN-Technologien und Analyse der Anforderungen des Nexus-Projektes wurde folgendes festgehalten:

TODO: †anforderungen ausformulieren, begruendung, erkläerung, usw.. TODO: † warum 5GHz - da 2,4 belegt..

TODO
TODO

3.1 Anforderungen an Hardware

- IEEE 802.11a kompatibel (5GHz Frequenzen)
- Ad-hoc Modus (auch bei 5GHz Frequenzen)
- Treiber für Linux und Windows
- Open-Source Firmware für Router
- Zusätzlich zu Wireless Mesh Network auch weitere (Netzwerk-)Schnittstellen zur Verwaltung vorsehen (LAN)
- Nach Möglichkeit keine selber gebastelten Lösungen
- Unterstützung von IEEE 802.11n wäre vorteilhaft
- 2 Mini PCI-Slots bei Routern, damit auf verschiedenen Frequenzbändern gesendet und empfangen werden kann (Performanceverbesserung)

3.2 Anforderungen an Software

- Betriebssystem ist nicht festgelegt, soll Ergebnis der Fachstudie sein
- Freiheit bei Routingprotokollen (Austauschbarkeit)
- Software soll Konfigurierung und Instrumentierung von WMN ermöglichen
- Topologie verändern bzw. erfassen soll möglich sein
- Visualisierung von Anfragen und Topologie

3.3 Zusätzliche Anforderungen

TODO: † Aufwandschätzung in aufgaben!

TODO

- Abdeckung des Gebäudes Universitätsstraße 38
- Verbindung mit Informatik-Netz nur über Gateway mit strikter Filterung (nur eine Richtung (UNI->Mesh) für die Verwaltung)
- Aufwandschätzung (wie viele Knoten usw.)
- Budget max. 25.000 Euro (evtl. mehr in Zukunft)
- 4-5 Mesh-Knoten pro Quadrat
- Speicherkapazität wichtig bei Routern

- PC + WLAN-Karten sind wegen Flexibilität vorzuziehen
- PDAs (bzw. andere kleine Clients) mit 802.11a

4 Hardware-Lösungen für den Aufbau eines Mesh-Netzwerkes

Es gibt verschiedene Möglichkeiten ein Meshnetzwerk aufzubauen. Im Weiteren werden einige davon im Detail beschrieben. Die Vorteile und Nachteile von einzelnen Möglichkeiten werden ebenfalls erläutert.

4.1 PCs + WLAN-Karten

Die einfachste Möglichkeit wäre die herkömmlichen PCs mit WLAN-Karten zu einem Mesh-Router einzurichten. Man nimmt dabei einfach die WLAN-Karten (PCI, Mini-PCI oder PCMCIA) und baut diese in PCs oder in Laptops ein. Man installiert dann auf diesen Rechnern entsprechende Treiber, die WLAN-Karten im Ad-Hoc Modus betreiben können und Routing-Software, z.B. OLSR-Daemon. **TODO:** †ueberarbeiten.. *Generelles Problem dabei ist, dass Ad-Hoc Modus bei Karten im 5Ghz Bereich von unausgereift bis nicht vorhanden ist (siehe 1.1.4).*

TODO

Hersteller haben gespart an der Entwicklung, da Ad-Hoc Modus einigermaßen kompliziert ist, und alle meist nur Infrastrukturmodus benutzt haben. Fehler liegen in Firmware von Chipsatz und im Treiber.

TODO: †ueberarbeiten.. Es gibt einen MadWiFi-Treiber, der für eine Vielzahl von Chipsätzen entwickelt wurde und *mit dem sollte es einigermaßen* funktionieren, sobald dieser noch zusätzlich gepatcht ist, und Firmware der Karte Ad-Hoc zulässt. Generell wegen der geringen Verbreitung von 802.11a in Europa, sind nur wenige Karten erhältlich. Z.B konnten Karten mit Atheros Chipsatz, z.B AR5004X, *uns* weiterhelfen.

TODO

TODO: †Installation ist relativ einfach - Hardware oder Software.. Stromversorgung brauchen alle)) Stromnebel..? **Vorteile:**

TODO

- Sehr Flexibel (Hardware kann noch nützlich sein)
- Installation ist relativ einfach
- Software Unterstützung ist vorhanden
- Mehrere WLAN und Ethernet Interfaces möglich

Nachteile:

- Groß und stationär
- Stromversorgung
- Schlechte Sende- und Empfangqualität, da sich die Antenne im elektromagnetischen Stromnebel des PCs befindet

4.1.1 PCI-WLAN-Karten

TODO: †Ziel des Abschnittes! Anfang ist schlecht..

TODO

Peripheral Component Interconnect, meist PCI abgekürzt, ist ein Bus-Standard zur Verbindung von Peripheriegeräten mit dem Chipsatz eines Prozessors. Da der PCI-Bus vom Prozessor relativ unabhängig ist, wird er nicht nur im PC benutzt, sondern z.B. auch im Alpha-PC oder im Macintosh. Über den PCI-Bus kann der Prozessor die wichtigsten Ein- und Ausgabekomponenten (z.B. Controller, Grafikkarte) „lokal“ und damit schneller ansprechen. Ursprünglich sollte der PCI-Bus die Anforderungen in PCs für Grafik-, Netzwerk- und andere Schnittstellenkarten über längere Zeit erfüllen. Allerdings wurde er schon nach kurzer Zeit zu langsam für schnelle Grafikkarten, so dass für diese 1997 ein zusätzlicher Steckplatz, der Accelerated Graphics Port (AGP), eingeführt wurde. Für so gut wie alle anderen Steckkarten-Typen blieb PCI dagegen bis heute Standard, soll aber ab **2005** schrittweise von PCI-Express ersetzt werden. TODO: †heute ist 2005))

TODO

PCI-WLAN-Karten werden in einen freien PCI-Steckplatz des Mainboards gesteckt. Ein Vorteil von PCI-WLAN-Karten ist die bessere Stabilität TODO: †gegenüber was..im Betrieb. Weiterhin besitzen die meisten PCI-WLAN-Karten die Möglichkeit die mitgelieferte Antenne gegen eine andere zu tauschen. Zu beachten ist, dass die Antenne üblicherweise direkt hinten an der Karte angebracht ist und somit in unmittelbarer Nähe zum PC-Gehäuse ist. Dies kann jedoch negative Auswirkungen auf die Reichweite oder den Datendurchsatz haben. Deshalb kann es für eine bessere Verbindung notwendig sein, die Antenne mit einem Koaxialkabel vom Rechnergehäuse zu entfernen.

TODO

Vorteile:

- Bessere Stabilität im Betrieb
- Meistens abschraubbare Antenne
- Verschwinden im Gehäuse, Platz wird nicht verschwendet

Nachteile:

- Oft recht schlechte Empfangs/Sendeleistung, weil die kleine Antenne direkt hinten am Rechner sitzt (Lösung: zusätzliche Antenne)

TODO: †Allgemein: Welche chipsätze in Frage kommen (atheros, intel..), Diskussion der Karten.. TODO: †Sicherheit - erlauterung!

TODO

TODO

4.1.1.1 Linksys WMP55AG



Abbildung 1: Linksys WMP55AG

Chipsatz:

- Atheros AR5213A

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)
- WPA
- LEAP

Treiber:

- Sehr gute Linux-Unterstützung, MadWifi-Treiber funktioniert mit dieser WLAN PCI-Karte ohne Probleme. Windows-Treiber werden von Linksys bereitgestellt.

Preis:

- ca. 90 Euro

Installation:

- Lasst sich leicht sowohl unter Windows als auch unter Linux (MadWifi-Treiber) installieren.

<http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Links:

- <http://madwifi.org/wiki/Compatibility/Linksys>
- <http://forums.fedoraforum.org/showthread.php?t=91165>
- http://www.pcworld.com/product/specs/prtprdid,704176/wireless_ag_54mbps_pci_adptr_80211a80211b80211g_compatible.html
- http://www.linksys.com/servlet/Satellite?c=L_CASupport_C2&childpagename=US%2FLayout&cid=1169671168007&pagename=Linksys%2FCommon%2FVisitorWrapper&lid=6800768007N09

4.1.1.2 Netgear WAG311



Abbildung 2: Netgear WAG311

Chipsatz:

- Atheros AR5212

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)
- WPA, WPA-PSK

- PPTP, P2TP, IPSec VPN pass-through

Treiber:

- Sehr gute Linux-Unterstützung, MadWifi-Treiber funktioniert mit dieser WLAN PCI-Karte ohne Probleme.

Preis:

- ca. 50-60 Euro

Installation:

- <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>
http://www.packetpro.com/~peterson/linux-netgear_wg311t_pci.html

Weitere Informationen:

- Externe Antenne, die mit der WLAN-PCI-Karte durch langes Kabel verbunden ist. Das Kabel lässt sich nicht von der PCI-Karte trennen.

Links:

- <http://www.netgear.com/Products/Adapters/AGDualBandWirelessAdapters/WAG311.aspx>
- <http://madwifi.org/wiki/Compatibility/Netgear>
- <http://www.linuxquestions.org/questions/mandriva-30/using-netgear-wag311-via-ma>
- http://www.packetpro.com/~peterson/linux-netgear_wg311t_pci.html
- http://www.netgear.com/upload/product/wag311/enus_ds_wag311.pdf

4.1.1.3 D-Link DWL-A520



Abbildung 3: D-Link DWL-A520

Chipsatz:

- Atheros AR5210

IEEE Standards:

- 802.11a

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)

Treiber:

- Von D-Link werden nur Treiber für Windows bereitgestellt. Sehr gute Linux-Unterstützung, MadWifi-Treiber funktioniert mit dieser WLAN PCI-Karte ohne Probleme.

Preis:

- ca. 70-80 Euro

Installation:

- <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Weitere Informationen:

- Antenne ist nicht abschraubbar.

Links:

- <http://support.dlink.com/products/print.asp?productid=DWL-A520>
- <http://madwifi.org/wiki/Compatibility/D-Link>

4.1.1.4 Gigabyte GN-WPEAG

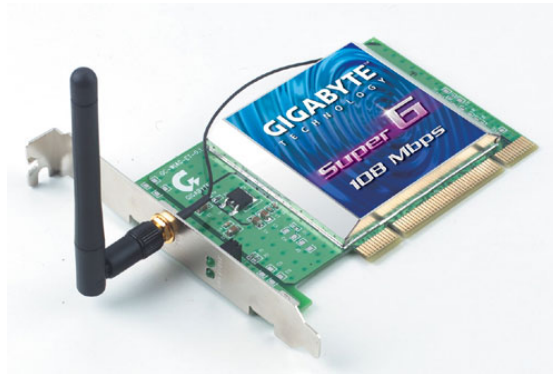


Abbildung 4: Gigabyte GN-WPEAG

Chipsatz:

- Atheros AR5212

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)
- WPA
- WPA2

Treiber:

- Von Gigabyte werden nur Treiber für Windows bereitgestellt.
http://www.gigabyte.com.tw/Support/Communication/Driver_Model.aspx?ProductID=952
Sehr gute Linux-Unterstützung, MadWifi-Treiber funktioniert mit dieser WLAN PCI-Karte ohne Probleme.
<http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Preis:

- ca. 70-80 Euro

Installation:

- <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Weitere Informationen:

- Abschraubbare Antenne mit reversed SMA. Eigentlich ist das eine Mini-PCI-Karte mit PCI-Adapter.

Links:

- http://www.gigabyte.com.tw/Products/Communication/Products_Spec.aspx?ProductID=952
- http://www.gigabyte.com.tw/Support/Communication/Driver_Model.aspx?ProductID=952
- <http://madwifi.org/wiki/Compatibility/Gigabyte>

4.1.1.5 Andere PCI-WLAN-Karten

- Intel PRO/Wireless 5000

Chipsatz Intel, 802.11a WLAN PCI-Karte, unterstützt Ad-Hoc- und Infrastruktur-Modus, Treiber von Intel nur für Windows vorhanden, für Linux werden keine Treiber entwickelt, kostet ca. 200 Euro

<http://support.intel.com/support/wireless/wlan/pro5000/pciadapter>
ftp://download.intel.com/support/wireless/wlan/pro5000/PR05000_INFO.pdf

- D-Link DWL-AG530

Chipsatz Atheros AR5212 oder AR5213, 802.11a/b/g WLAN-Karte, MadWifi-Treiber Unterstützung, Externe abschraubbare Antenne, kostet ca 80 Euro

<http://www.dlink.com/products/?pid=306>
<http://madwifi.org/wiki/Compatibility/D-Link>

- D-Link DWL-G550

Chipsatz Atheros AR5212, 802.11a/b/g WLAN-Karte, MadWifi-Treiber Unterstützung, Externe abschraubbare Antenne, kostet ca 60 Euro

<http://www.dlink.com/products/?pid=414>
<http://madwifi.org/wiki/Compatibility/D-Link>

TODO: †Allgemein: Tabellenübersicht + Freitext, Andere PCI-WLAN-Karten einheitlich (ausführlich?) beschreiben oder erklären warum werden sie nur kurz erwähnt und anderen ausführlich beschrieben..

TODO

4.1.2 Mini-PCI WLAN-Karten

Mini-PCI ist eine vor allem für die Nutzung in Notebooks und Laptops miniaturisierte Version des PCI Steckplatzes, wie er in allen Desktop PCs vorkommt. PCI steht dabei für Peripheral Component Interconnect. Die Abmessungen einer Mini-PCI Card betragen 6,0 x 4,6 x 0,5 cm.

TODO: †PCI in abkuerzungsverzeichniss und hier erklaerung entwernen..

TODO

Mini-PCI WLAN-Karten sind ursprünglich für Laptops gedacht, sind aber mit entsprechenden Adaptern (PCI-zu-MiniPCI) und externen Antennen auch im normalen PCs zu verwenden. Als Vorteil ist dabei die Flexibilität zu nennen. Als Nachteil, die Zusätzlichen Kosten und Aufwand der Installation. Meist sind Mini-PCI Cards für Wireless LAN bereits vom Hersteller eingebaut. Der Vorteil der Ausführung als standardisiertes Modul liegt darin, daß eine Mini-PCI Karte in aller Regel einfach gegen eine andere Karte, auch eines anderen Herstellers, ausgetauscht werden kann. Im Falle der WLAN Mini-PCI Module kann z.B. problemlos vom langsameren 802.11b Standard auf ein schnelleres WLAN Modul nach 802.11g gewechselt werden.

TODO: †Installationen? **Vorteile:**

TODO

- Können mit Hilfe eines Adapters zu einer PCI-WLAN-Karte umgebaut werden
- Können leicht ausgetauscht werden

Nachteile:

- Meistens kostenintensiv
- Installationen

4.1.2.1 Wistron CM9 Atheros AR5213A



Abbildung 5: Wistron CM9 Atheros AR5213A

Chipsatz:

- Atheros AR5213A

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)
- WPA
- WPA2

Treiber:

- Hervorragende Unterstützung von MadWifi-Treiber [5], auch Ad-Hoc-Modus.

Preis:

- ca. 40 Euro

Installation:

- <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Links:

- <http://www.alix-board.de/produkte/wistroncm9.html>
- <http://www.pcengines.ch/cm9.htm>
- <http://forum.openwrt.org/viewtopic.php?pid=10213>
- <http://madwifi.org/ticket/1209>

4.1.2.2 Intel PRO/Wireless 3945



Abbildung 6: Intel PRO/Wireless 3945

Chipsatz:

- Intel

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-bit)
- WPA
- WPA2

Treiber:

- Es werden von Intel Treiber sowohl für Windows als auch für Linux bereitgestellt.

http://downloadcenter.intel.com/Product_Filter.aspx?ProductID=2259

Von Intel wurde ein Projekt für die Unterstützung von Intel PRO/Wireless 3945 erstellt.

<http://ipw3945.sourceforge.net>

Der ipw3945-Treiber funktioniert auch im Ad-Hoc-Modus, aber nicht sehr stabil, es kommt oft zu Verbindungsabbrüchen.

Preis:

- ca. 20-30 Euro

Installation:

- Im Gegensatz zu den „klassischen“ Intel Wireless-Chipsätzen 2100- und 2200BG-Chipsätzen ist der Treiber für den 3945ABG noch nicht im Kernel verfügbar. Um auch damit kabellos ins Internet zu gehen, sind ein paar Handgriffe notwendig.

<http://ipw3945.sourceforge.net/README.ipw3945>

<http://ipw3945.sourceforge.net/INSTALL>

Links:

- http://www.intel.com/network/connectivity/products/wireless/prowireless_mobile.htm
- http://downloadcenter.intel.com/Product_Filter.aspx?ProductID=2259
- <http://ipw3945.sourceforge.net/>
- <http://ipw3945.sourceforge.net/README.ipw3945>
- <http://ipw3945.sourceforge.net/INSTALL>

4.1.2.3 Intel PRO/Wireless 2915



Abbildung 7: Intel PRO/Wireless 2915

Chipsatz:

- Intel

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-bit)
- WPA
- WPA2

Treiber:

- Es werden von Intel Treiber sowohl für Windows als auch für Linux bereitgestellt.

http://downloadcenter.intel.com/Product_Filter.aspx?ProductID=1847

Von Intel wurde ein Projekt für die Unterstützung von Intel PRO/Wireless 2915 erstellt.

<http://ipw2200.sourceforge.net>

Der ipw2200-Treiber funktioniert auch im Ad-Hoc-Modus, aber nicht sehr stabil, es kommt oft zu Verbindungsabbrüchen. Der ipw2200-Treiber ist im

Kernel 2.6 enthalten, kann aber auch separat als Modul kompiliert werden.
Der im Kernel enthaltene Treiber unterstützt den Monitor-Modus nicht.

Preis:

- ca. 30 Euro

Installation:

- <http://ipw2200.sourceforge.net/README.ipw2200>
<http://ipw2200.sourceforge.net/INSTALL>

Links:

- <http://support.intel.com/support/wireless/wlan/pro2915abg>
- http://download.intel.com/support/wireless/wlan/pro2915abg/sb/303330002us_channel.pdf
- <http://ipw2200.sourceforge.net/>
- <http://www.intel.com/cd/personal/computing/emea/deu/234998.htm>
- http://downloadcenter.intel.com/Product_Filter.aspx?ProductID=1847

4.1.2.4 Intel Wireless WiFi Link 4965AGN



Abbildung 8: Intel Wireless WiFi Link 4965AGN

Chipsatz:

- Intel

IEEE Standards:

- 802.11a/b/g/n(draft)

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-bit)
- WPA
- WPA2

Treiber:

- <http://www.intellinuxwireless.org/>

Preis:

- ca. 30 Euro

Installation:

- <http://www.intellinuxwireless.org/>

Links:

- http://www.intel.com/network/connectivity/products/wireless/wireless_n/overview.htm
- <http://www.intellinuxwireless.org/>
- <http://www.wifi-info.de/intel-kuendigt-11n-chipsatz-fuer-centrino-notebooks-an/01/2007/>
- http://downloadcenter.intel.com/filter_results.aspx?strTypes=all&ProductID=2753&OSFullName=Linux*&lang=eng&strOSs=39&submit=Go%21

4.1.3 PCMCIA WLAN-Karten

Diese WLAN-Karten sind für Notebooks gedacht. Heutzutage ist es jedoch üblich, dass die Notebooks schon ein integriertes WLAN-Modul (Mini-PCI oder Mini-PCIe) eingebaut haben. Damit ist die Notwendigkeit dieser Module nur noch für Notebooks älterer Generationen notwendig. Die PCMCIA WLAN-Karten spielen für die Zwecke der Fachstudie keine besonders große Rolle und werden hier nur wegen eines möglichen Einsatzes in mobilen Clients des Mesh-Netzwerkes betrachtet. Im folgenden werden die Vorteile und Nachteile der PCMCIA WLAN-Karten aufgelistet.

Vorteile:

- Leichte Installation
- Leicht austauschbar
- Sehr gute Unterstützung durch Treiber sowohl unter Linux als auch Windows
- Hervorragend geeignet für mobile Clients des Mesh-Netzwerkes

Nachteile:

- Sie sind veraltet und deswegen schwer zu finden
- Selten ein Anschluss für eine externe Antenne vorhanden
- Relativ teuer im Vergleich zu Mini-PCI und Mini-PCIe Karten

4.1.3.1 Proxim Orinoco Gold 8480-WD



Abbildung 9: Proxim Orinoco Gold 8480-WD

Chipsatz:

- Atheros AR5212

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-bit)
- WPA
- WPA2

Treiber:

- Unter Linux hervorragende Unterstützung von MadWifi-Treiber [5], auch Ad-Hoc-Modus.

Preis:

- ca. 80 Euro

Installation:

- <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Links:

- <http://www.proxim.com/products/wifi/client/abgcard/index.html>
- <http://madwifi.org/wiki/Compatibility/Proxim>

4.1.3.2 Netgear WAG511



Abbildung 10: Netgear WAG511

Chipsatz:

- Atheros AR5001X+

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)
- WPA
- WPA2
- PTP, P2TP, IPSec, VPN pass-through

Treiber:

- Von Netgear werden nur Windows Treiber angeboten.
<http://www.netgear.de/de/Support/download.html?func=Detail&id=10676>
Unter Linux hervorragende Unterstützung von MadWifi-Treiber, auch Ad-Hoc-Modus.
<http://madwifi.org/>

Preis:

- ca. 50-60 Euro

Installation:

- http://www.lrz-muenchen.de/services/netz/mobil/funklan-installation/installation/windowsxp_wg511/flan-instwpx.html
<http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Links:

- <http://www.netgear.com/Products/Adapters/AGDualBandWirelessAdapters/WAG511.aspx>
- <http://www.netgear.de/de/Support/download.html?func=Detail&id=10676>
- <http://www.netgear.de/Produkte/Wireless/DualBand/WAG511/index.html>
- <http://madwifi.org/wiki/Compatibility/Netgear>

4.1.3.3 SMC 2536W-AG



Abbildung 11: SMC 2536W-AG

Chipsatz:

- Atheros AR5001

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)
- WPA
- WPA2

Treiber:

- Von SMC werden nur Treiber für Windows angeboten.
http://www.smc.com/index.cfm?event=downloads.searchResultsDetail&localeCode=EN_USA&productCategory=9&partNumber=2916&modelName=348&knowsPartNumber=false&userPartNumber=&docId=3103

Unter Linux hervorragende Unterstützung von MadWifi-Treiber [5], auch Ad-Hoc-Modus.

Preis:

- ca. 80 Euro

Installation:

- <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Links:

- <http://www.smc.com/index.cfm?event=viewProduct&cid=9&scid=49&localeCode=EN%5FUSA&pid=348>
- http://www.smc.com/files/AC/2536Wag_Ds_ww.pdf
- <http://madwifi.org/wiki/Compatibility/SMC>
- <http://forums.fedoraforum.org/archive/index.php/t-101517.html>

4.1.3.4 Linksys WPC55AG



Abbildung 12: Linksys WPC55AG

Chipsatz:

- Atheros AR5212 oder AR5006X

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Sicherheit:

- WEP (40-, 104-, 128-bit)
- WPA
- WPA2

Treiber:

- Von Linksys werden nur Treiber für Windows bereitgestellt.
http://www.linksys.com/servlet/Satellite?c=L_Product_C2&childpagename=US%2FLayout&cid=1115416827328&pagename=Linksys%2FCommon%2FVisitorWrapper
 Auch hier kann man Treiber für Windows finden:
<http://www.phoenixnetworks.net/atheros.php>
 Unter Linux hervorragende Unterstützung von MadWifi-Treiber [5], auch Ad-Hoc-Modus.

Preis:

- ca. 50-60 Euro

Installation:

- <http://madwifi.org/wiki/UserDocs/FirstTimeHowTo>

Links:

- http://www.linksys.com/servlet/Satellite?c=L_Product_C2&childpagename=US%2FLayout&cid=1115416827328&pagename=Linksys%2FCommon%2FVisitorWrapper
- <http://www.phoenixnetworks.net/atheros.php>
- <http://madwifi.org/>
- <http://madwifi.org/wiki/Compatibility/Linksys>
- http://reviews.cnet.com/adapters-nics/linksys-wpc55ag-dual-band/4505-3380_7-21128291.html
- <http://www.google.de/search?q=Linksys+WPC55AG>
- <http://lists.funkfeuer.at/pipermail/discuss/2006-September/001592.html>
- <http://www.uk-surplus.com/manuals/brochures/linksyswpc55duser.pdf>

4.1.3.5 Andere PCMCIA-WLAN-Karten

- Intel PRO/Wireless 5000
Chipsatz Intel, 802.11a WLAN PCMCIA-Karte, unterstützt Ad-Hoc- und Infrastruktur-Modus, Treiber von Intel nur für Windows vorhanden, für Linux werden keine Treiber entwickelt, kostet ca. 150 Euro
<http://www.intel.com/support/wireless/wlan/pro5000/lancardbus>
- Netgear WAB501 Chipsatz Atheros AR5211, 802.11a/b WLAN-Karte, MadWifi-Treiber Unterstützung
<http://kbserver.netgear.com/products/WAB501.asp>
<http://madwifi.org/wiki/Compatibility/Netgear>
- Netgear WG511U
Chipsatz Atheros AR5004X, 802.11a/g WLAN-Karte, MadWifi-Treiber Unterstützung
<http://www.netgear.com/Products/Adapters/AGDualBandWirelessAdapters/WG511U.aspx>
<http://madwifi.org/wiki/Compatibility/Netgear>

- Proxim Orinoco Silver 8481-WD
Chipsatz Atheros AR5001X+, 802.11a/b/g WLAN-Karte, MadWifi-Treiber Unterstützung, kostet ca 80-90 Euro
<http://www.proxim.com/products/wifi/client/abgcard/index.html>
<http://madwifi.org/wiki/Compatibility/Proxim>
- Cisco Aironet CB21AG
Chipsatz Atheros 5212, 802.11a/b/g WLAN-Karte, Madwifi-Treiber Unterstützung, kostet ca 100 Euro
<http://madwifi.org/wiki/Compatibility/Cisco>

4.2 WLAN-Router

TODO: †nicht nur WLAN und WAN -> auch zwieschen switch und Wlan(WRT54 mit DD-WRT)

TODO

Die Kombination aus Access Point und Router wird häufig als WLAN-Router bezeichnet. Das ist solange korrekt, soweit es einen WAN-Port gibt. Das Routing findet dann zwischen WLAN und WAN (und falls vorhanden auch zwischen LAN und WAN) statt. Fehlt dieser WAN-Port, handelt es sich hier lediglich um Marketing-Begriffe, da reine Access Points auf OSI-Ebene 2 arbeiten und somit Bridges und keine Router sind. Oft sind das aber keine vollständigen Router, da diese Geräte ausschließlich als Internetzugangs-Systeme dienen und nur mit aktiviertem PPPoE (oder PPPoA) sowie NAT-Routing (oder IP-Masquerading) eingesetzt werden können.

In diesem Abschnitt werden so genannte *stand-alone* Lösungen für eine WMN vorgestellt.

4.2.1 SoHo-Router

Man kann herkömmliche WLAN-Router für Heimanwender (SoHO-Router - small or home office) kaufen, die sich mit alternativer Firmware (spezielle Linux-Software mit OLSR-daemon) zu einem Mesh-Router umrüsten lassen. Ein WLAN-Router ist die Kombination eines normalen Router (Kabelrouter) mit einem Accesspoint. Es gibt solche mit eingebauten Modem und andere mit einem Anschluss (WAN-Port) dafür (für Modems mit LAN-Anschluss). Ein Nachteil ist, dass es viele Modelle gibt, die eine fix verbaute Antenne haben, die nicht gewechselt werden kann.

Vorteile:

- Klein und handlich
- Mobil und flexibel
- Sehr günstig
- Gute Reichweite
- Wenig Stromverbrauch
- Leichte Konfiguration und Installation

Nachteile:

- Eingeschränkte Software-Unterstützung

- Open-Source Firmware schwer zu finden
- Durch das Öffnen von Geräten und das Einspielen von fremder Firmware erlischt die Garantie des Herstellers
- Eingeschränkter Funktionsumfang

4.2.1.1 Linksys WRT54G v1.0



Abbildung 13: Linksys WRT54G v1.0

IEEE Standards:

- 802.11b/g
- 802.11a/b/g (wenn man die mitgelieferte Mini-PCI WLAN-Karte durch z.B. Atheros 802.11a/b/g WLAN-Karte austauscht)

Betriebsart:

- Ad-Hoc
- Infrastruktur

TODO: †Erläuterung warum diese benötigt wird, was ist damit möglich usw. TODO
Beschreibung oder/und Verweis auf OpenWRT abschnitt..

Firmware:

- Es gibt mehrere fremde frei verfügbare Firmware für dieses Gerät. Alle unten aufgeführten Firmware sind Open-Source Projekte:
OpenWRT
<http://wiki.openwrt.org/OpenWrtDocs/Hardware/Linksys/WRT54G>
DD-WRT
http://www.dd-wrt.com/wiki/index.php/Linksys_WRT54G/GL/GS/GX

Preis:

- ca. 40-50 Euro

Installation:

- Die mitgelieferte Mini-PCI WLAN-Karte durch z.B. Atheros 802.11a Mini-PCI austauschen und oben erwähnte frei verfügbare Firmware installieren (siehe oben Firmware).

Weitere Informationen:

- Ein Mini-PCI Slot ist für eine WLAN-Karte vorhanden.

Links:

- <http://wiki.openwrt.org/OpenWrtDocs/Hardware/Linksys/WRT54G>
- http://www.dd-wrt.com/wiki/index.php/Linksys_WRT54G/GL/GS/GX
- <http://forum.opennet-initiative.de/thread.php?threadid=505&sid=56c53647db6353a4>
- <http://www.linksysinfo.org/forums/showthread.php?t=47124>

4.2.1.2 Linksys WRT55AG



Abbildung 14: Linksys WRT55AG

IEEE Standards:

- 802.11a/b/g

Betriebsart:

- Ad-Hoc
- Infrastruktur

Firmware:

- Open-Source Firmware befindet sich noch in Entwicklung:
Modifizierte Version von OpenWRT Kamikaze
<http://legacy.not404.com/cgi-bin/trac.fcgi/wiki/OpenWRT/Atheros/Linksys/WRT55AGv2#KamikazeKernelonWRT55AGv2>
OpenWRT
<http://wiki.openwrt.org/OpenWrtDocs/Hardware/Linksys/WRT55AG>

Preis:

- ca. 70-80 Euro

Weitere Informationen:

- 2 Slots sind für Mini-PCI WLAN-Karten vorhanden.

Links:

- <http://wiki.openwrt.org/OpenWrtDocs/Hardware/Linksys/WRT55AG>
- http://www.tomsnetworking.de/content/tests/j2003a/test_linksys_wrt55ag/index.html
- http://reviews.cnet.com/routers/linksys-wrt55ag-wireless-a/4505-3319_7-21131921.html
- <http://legacy.not404.com/cgi-bin/trac.fcgi/wiki/OpenWRT/Atheros/Linksys/WRT55AGv2>

4.2.1.3 Asus WL500G/GP



Abbildung 15: Asus WL500G/GP

IEEE Standards:

- 802.11b/g
- 802.11a/b/g (wenn man die mitgelieferte Mini-PCI WLAN-Karte durch z.B. Atheros 802.11a/b/g WLAN-Karte austauscht)

Betriebsart:

- Ad-Hoc
- Infrastruktur

Firmware:

- Es sind mehrere fremde frei verfügbare Firmware für dieses Gerät. Alle unten aufgeführten Firmware sind Open-Source Projekte:
 - OpenWRT
 - <http://wiki.openwrt.org/OpenWrtDocs/Hardware/Asus/WL500G>
 - <http://wiki.openwrt.org/OpenWrtDocs/Hardware/Asus/WL500GP>
 - FreeWRT
 - <http://www.freewrt.org/trac/wiki/Documentation/Hardware/AsusWL500G>
 - <http://www.freewrt.org/trac/wiki/Documentation/Hardware/AsusWL500GP>
 - Olegs custom firmware
 - <http://oleg.wl500g.info>

Preis:

- ca. 70-80 Euro

Installation:

- Die mitgelieferte Mini-PCI WLAN-Karte durch z.B. Atheros 802.11a Mini-PCI austauschen und oben erwähnte frei verfügbare Firmware installieren (siehe oben Firmware).

http://wiki.opennet-initiative.de/index.php/Mini-PCI_Umbau

Weitere Informationen:

- Ein Mini-PCI Slot ist für eine WLAN-Karte vorhanden.

Links:

- <http://wiki.opennet-initiative.de/index.php/AP9>
- <http://wiki.openwrt.org/OpenWrtDocs/Hardware/Asus/WL500G>
- <http://wiki.openwrt.org/OpenWrtDocs/Hardware/Asus/WL500GP>
- <http://www.freewrt.org/trac/wiki/Documentation/Hardware/AsusWL500G>
- <http://www.freewrt.org/trac/wiki/Documentation/Hardware/AsusWL500GP>
- <http://wl500g.dyndns.org/>
- <http://oleg.wl500g.info/>
- <http://au.asus.com/products.aspx?l1=12&l2=43>
- http://www.freifunk-bno.de/component/option,com_smf/Itemid,88/topic,910.msg10357/
- http://www.cyber-wulf.de/a_wl500g.html
- <http://wiki.openwrt.org/OpenWrtDocs/Hardware/Asus/WL500G>
- <http://forum.opennet-initiative.de/print.php?threadid=505&page=6&sid=460903353d70c65fad4960105ab76cdd>
- <http://forum.openwrt.org/viewtopic.php?pid=41756>
- <http://www.familie-prokop.de/asus-wl500gp/index.html>

4.2.1.4 Andere WLAN-Router

- Netgear HR314

802.11a WLAN-Router, unterstützt Ad-Hoc- und Infrastruktur-Modus, keine Open-Source Firmware vorhanden, kostet ca. 30 Euro

<http://www.wi-fiplanet.com/reviews/article.php/1559091>

4.2.2 Professionelle Router

In diesem Abschnitt werden sogenannte professionelle Mesh-Router betrachtet. Die Begriffe, die dafür oft als Synonyme verwendet werden, sind dabei:

- Routerboards
- Stand-alone Mesh-Router
- Minicomputers
- Single-Board-Computers (SBC)
- Access Points (AP)

TODO: †ueberarbeiten und SEPARATE diskussion in eigenem Abschnitt

TODO

Im Projekt (<http://umic-mesh.net>) wurden professionelle Router eingesetzt, das sind spezielle Router-Boards mit Steckplätzen für MiniPCI WLAN-karten. Boards kosten etwa 100-200 Euro, dazu muss man allerdings noch passende WLAN-Karten kaufen + Antennen + Kabel + Netzteil + Gehäuse, also keine billige Lösung.

TODO: †identisch zur bereits diskutierten PC-Loesung? *Man könnte aber nur diese Karten kaufen + Adapter PCI-MiniPCI und in Rechner einbauen (Das wäre dann die platzsparende Version von PCs + WLAN-Karte). WLAN-Karten z.B Wistron Neweb CM9 Atheros 802.11a/b/g Mini-PCI, hier (<http://www.pcengines.ch/cm9.htm>), und Boards sind hier (<http://www.pcengines.ch/wrap.htm>, <http://www.pcengines.ch/alix.htm>).*

TODO

Es gibt noch diese kleine Mesh-Router, wie von Meraki. Die haben wohl ihre eigene Firmware drin und eigene Routingprotokolle oder eigene Implementierungen davon besser gesagt. Hier ein Paar, die 802.11a unterstützen, sind aber outdoor, haben also große Reichweiten. Ob es sinnvoll ist, sie im Gebäude einzusetzen:

- Aphelion 3300AG Outdoor Wireless Access Point - 802.11a/b/g
- Aphelion 600AG/605AG Intelligenter sequentieller Wireless Access Point für den Außenbereich mit den Standards 802.11a/b/g (http://www.abcddata.de/abcdataneu/WLAN_MESH_Aphelion.php)
- PLANET MAP-2100 - indoor - sind aber zum Teil sehr teuer (1200 Euro)

Vorteile:

- Outdoor (in unserem Fall irrelevant)
- Große Reichweiten

Nachteile:

- Sehr teuer
- Closed source

Links:

- <http://wiki.opennet-initiative.de/index.php/WRAP>
- http://www.abcddata.de/abcdataneu/WLAN_MESH_Aphelion.php
- http://www.aerial.net/shop/product_info.php?cPath=33&products_id=351
- <http://forum.openwrt.org/viewtopic.php?id=9655>

4.2.3 Access Points

Ein WLAN-Accesspoint ist der Verbindungspunkt eines kabelbasierten Netzwerkes zu einem WLAN. Der Accesspoint ist Basisstation für alle WLAN-Clients, zu der sie eine drahtlose Verbindung aufbauen. Sendet ein WLAN-Client Daten, die für einen Empfänger im kabelbasierten Netzwerkteil bestimmt sind, so *reicht* der Accesspoint diese Daten über das Kabelnetz an den Empfänger weiter. Weiterhin kann ein Accesspoint auch mehrere WLAN-Clients untereinander verbinden. Somit ist der Accesspoint quasi ein kabelloser Switch.

Dieser hat (je nach Ausstattung) einige der folgenden Optionen:

- Ein oder mehrere integrierte WLAN-Module
- Einen integrierten DHCP-Server
- Umfangreiche Sicherheits- und Verschlüsselungsmöglichkeiten (WEP, WPA und WPA2 dienen der Verschlüsselung der zu ubetragenden Daten ; MAC-Filter und SSID Optionen ; Einstellungen bezüglich des Remotezugriffs)
- Verschiedene Arbeitsmodi (Accesspoint (AP), Bridge (Point-to-Point oder Point-to-Multipoint), Repeater, MESSID)

Einige Access Points:

- Intel PRO/Wireless 5000
<http://support.intel.com/support/wireless/wlan/pro5000/accesspoint>
<http://www.pcmag.com/article2/0,1759,5524,00.asp>
- Linksys WAP55AG
http://www.tomsnetworking.de/content/aktuelles/news_beitrag/news/851/6/index.html
- NETGEAR WAB102
<http://kbserver.netgear.com/products/WAB102.asp>
http://reviews.cnet.com/wireless-access-points/netgear-wab102-802-11a/4505-3265_7-20708150.html
<http://archive.cert.uni-stuttgart.de/bugtraq/2003/12/msg00159.html>

4.3 PDAs und Handys

Hier sind einige WLAN-fähige PDAs und Handys :

- Apple iPhone Smartphone
IEEE 802.11b/g
<http://pocketpccentral.net/smartphone/apple/iphone.htm>
<http://www.tomshardware.com/de/test-apple-iphone-vs-ipaq-hw6910-blackberry-pearl-ht-testberichte-239720.html>
- RIM BlackBerry 8820 Smartphone
IEEE 802.11a/b/g
<http://www.blackberry8800series.com>
http://www.reghardware.co.uk/2007/08/14/review_blackberry_8820/
http://eu.blackberry.com/eng/devices/device-detail.jsp?navId=H0,C201,P563#tab_tab_overview
- Motorola Symbol MC70 Smartphone
IEEE 802.11a/b/g
http://www.handheld-loesungen.com/symbol_mc70.htm
- i-mate K-jam Smartphone
IEEE 802.11b/g
http://www.lordpercy.com/imate_kjam_review.htm
http://www.mobiletechreview.com/i-mate_K-JAM.htm

- Sony Ericsson G900 Smartphone
IEEE 802.11b/g
<http://www.sonyericsson.com/cws/corporate/press/pressreleases/pressreleasedetails/g700andg900global-20080210>
- OQO Model 02 UMPC
IEEE 802.11a/b/g
<http://www.worldofppc.com/HWTests/oqo02.htm>

5 Systemsoftware für Mesh-Netzwerk

In Kapitel 4 haben wir mehrere Hardware-Lösungen für ein Mesh-Netzwerk vorgestellt. Für alle im Kapitel 4 vorgestellten PCI-, MiniPCI- und PCMCIA-Karten werden vom Hersteller dieser WLAN-Karten Windows-Treiber bereitgestellt. Und nur wenige Hersteller (z.B. Intel) haben auch Linux-Treiber für ihre Karten implementiert. Glücklicherweise existiert für WLAN-Karten mit Atheros-Chipsatz der Open-Source Linux-Treiber MadWifi, der alle im Kapitel 4 aufgelisteten WLAN-Karten mit Atheros-Chipsatz unterstützt. Bei SoHo-Routern sieht die Situation etwas anders aus. Hier gibt es nicht so viele Möglichkeiten bei der Wahl nach einer Software. Die meisten Hersteller von SoHo-Routern stellen den Source-Code des Betriebssystems für SoHo-Router nicht bereit. Deswegen wird eine Open-Source Firmware gebraucht, mit der SoHo-Router im Ad-Hoc Modus betrieben werden können, denn die meisten SoHo-Router arbeiten nur im Infrastruktur Modus und den Ad-Hoc Modus nicht realisieren.

In diesem Kapitel wird verschiedene Treiber- und Routing-Software vorgestellt, die zusammen mit der Hardware aus Kapitel 4 die Realisierung von Ad-Hoc Mesh-Netzwerken ermöglicht.

5.1 Linux MadWiFi-Treiber

Linux MadWifi-Treiber [5] ist Linux Kernel Treiber für WLAN-Karten mit Atheros Chip-satz. Linux MadWifi-Treiber ist heutzutage einer der fortgeschrittensten Linux Treiber für WLAN-Karten. Der Treiber ist stabil und hat eine große Benutzergemeinschaft. Der MadWifi-Treiber selbst ist Open-Source, verwendet aber eine proprietäre Softwareschicht Hardware Abstraction Layer (HAL), die nur in binärer Form vorhanden ist.

Das Hardware Abstraction Layer (HAL) wird vom MadWifi-Treiber gebraucht, um die Atheros-Chips ansprechen zu können. Dafür wurde bisher ein Closed-Source-Modul verwendet. Dies hat unter anderem damit zu tun, dass die Atheros-Chipsätze prinzipiell auf Frequenzen funken könnten, für die sie nicht zugelassen sind - beispielsweise weil diese vom Militär zur Kommunikation verwendet werden.

Durch das proprietäre Modul war der Madwifi-Treiber bisher jedoch von einer Aufnahme in den Linux-Kernel ausgeschlossen. Die Entwickler hatten außerdem das Problem, dass sie Fehler unter Umständen nicht beheben konnten, da sie nicht nachvollziehen konnten, wie der HAL-Baustein arbeitet.

MadWifi selbst wird daher ab sofort nicht weiterentwickelt. Stattdessen setzen die Programmierer auf OpenHAL, eine Linux-Portierung des HAL-Modules des in OpenBSD verfügbaren freien Atheros-Treibers. In der Vergangenheit wurde vom Software Freedom Law Center (SFLC) bestätigt, dass die durch Reverse Engineering entstandene Software keine Copyrights verletzt. Solche Behauptungen hatten die Entwicklung lange

ausgebremst.

Der neue Treiber „Ath5k“ wird MadWifi nun ersetzen und soll nicht nur die freie Komponente OpenHAL einsetzen, sondern auch mit dem neuen Linux-WLAN-System Mac80211 zusammenarbeiten, so dass der Treiber in den offiziellen Linux-Kernel gelangen kann. MadWifi soll jedoch weiter mit Fehlerkorrekturen und HAL-Updates versorgt werden.

Links:

- <http://madwifi.org/>
- <http://madwifi.org/wiki/About/ar5k>
- <http://madwifi.org/wiki/About/OpenHAL>
- <http://madwifi.org/wiki/UserDocs/GettingMadwifi>
- <http://madwifi.org/wiki/Compatibility>
- <http://www.intellinuxwireless.org/?p=mac80211>

5.2 OpenWRT

OpenWRT [6] ist eine GNU/Linux-Distribution für WLAN-Router. Anstatt einer statischen Firmware setzt OpenWRT auf ein voll beschreibbares Dateisystem sowie einen Paketmanager. OpenWRT läuft unter anderem auf Geräten der Firmen Linksys, ALLNET, ASUS, Belkin, Buffalo, Microsoft und Siemens.

Vorteile:

- Flexibilität
- Erweiterbarkeit
- Individualisierbarkeit
- Sicherheit
- Gewohnte Linux-Flexibilität und Funktionsumfang!

Nachteile:

- Standardmäßig sind nur die nötigsten Unix-Tools vorhanden

Links:

- <http://openwrt.org/>
- <http://toh.openwrt.org/>

5.3 Mesh-Routing Software

In diesem Abschnitt werden Routing-Daemonen vorgestellt, die ein bestimmtes Routing-Protokoll implementieren und auf jedem Knoten in einem Mesh-Netzwerk ausgeführt werden. Diese Daemonen tauschen Routing-Informationen aus und machen es möglich, Nachrichten von einem Knoten zu einem anderen Knoten im Mesh-Netzwerk zu transportieren. Routing-Daemonen ermöglichen die Kommunikation zwischen 2 Knoten in einem Ad-Hoc Mesh-Netzwerk, zwischen denen mehr als 1 Hop liegt.

5.3.1 olsr.org OLSR daemon



Der olsr.org OLSR daemon [7] ist eine Implementierung des Optimized Link State Routing Protokolls. OLSR ist ein Routing-Protokoll für mobile Ad-Hoc Netzwerke. Der Protokoll ist pro-aktiv, tabellengesteuert und nutzt die Technik Multipoint Relaying (MPR) zum Fluten von Nachrichten. olsrd implementiert ausserdem auch eine populäre Erweiterung Link Quality Extension. Zur Zeit ist die Implementierung von olsrd verfügbar für GNU/Linux, Windows, OS X, FreeBSD, OpenBSD and NetBSD. olsrd ist eine gut strukturierte und kodierte Implementierung, die leicht zu warten, zu erweitern und auf andere Plattformen zu portieren sein soll. Die Implementierung ist konform zu RFC3626 in Bezug auf die Kernfunktionalität und die zusätzlichen Funktionen. olsrd unterstützt das Konzept von ladbaren Plug-Ins. Mit diesen Plug-Ins kann man benutzerdefinierte Pakete mit Hilfe des OLSR MPR Flutens versenden und behandeln oder irgendeine andere zusätzliche Funktionalität bereitstellen.

Links:

- <http://ietf.org/rfc/rfc3626.txt>
- http://wiki.freifunk.net/OLSR_mit_Windows
- http://wireless.subsignal.org/index.php?title=Laptop_mit_OLSR
- <http://wiki.opennet-initiative.de/index.php/OLSR>

5.3.2 Open-Mesh B.A.T.M.A.N. daemon



Der B.A.T.M.A.N.-Daemon [4] steht bislang für Linux, FreeBSD und Macintosh OS-X zur Verfügung. Die Entwicklungsarbeit konzentriert sich jedoch in erster Linie auf Linux, weshalb es vorkommen kann, dass erweiterte Funktionen unter anderen Betriebssystemen erst mit einer gewissen Verzögerung zur Verfügung stehen.

Linux-Installationspakete des B.A.T.M.A.N.-Daemons batmand gibt es für Debian, OpenZaurus und OpenWRT. Zum Kompilieren aus dem Quelltext genügt ein einfaches make und make install im Sourcecodeverzeichnis. Als einzige Abhängigkeit wird die Bibliothek libpthread vorausgesetzt, die auf einem Linux-System, üblicherweise bereits installiert sein sollte.

Um über ein B.A.T.M.A.N.-Mesh ins Internet gehen zu können, muss außerdem unter Linux das Kernelmodul tun installiert sein. Es ist im Standardkernel der meisten Linuxdistributionen enthalten und wird beim ersten Start des B.A.T.M.A.N.-Daemons automatisch geladen. Wer einen selbstkompilierten Kernel einsetzt, findet es zum Beispiel in xconfig in der Abteilung Network Device Support unter der Bezeichnung Universal TUN/TAP device driver support.

Links:

- <https://www.open-mesh.net/Members/adagio/batman-install-howto-stichworte>
- http://open-mesh.net/batman/doc/batmand_howto.pdf
- https://www.opensourcepress.de/fileadmin/osp/pdf/mesh_leseprobe.pdf

5.3.3 Meshcom Driver



Der MeshDriver klemmt sich zwischen den WLAN-Treiber und den TCP/IP-Stack. Anders als 802.11s funkt er im Ad-hoc-Modus von 802.11 und bezieht eine Ethernet-Schnittstelle automatisch in das vermaschte Netz ein, etwa für den Internet-Zugang. Meshcom stellt

eine Beta-Version für privaten Einsatz und Forschungszwecke kostenlos zur Verfügung, die unter Windows XP oder Linux mit Kernel 2.6 läuft. Sie beherrscht allerdings einige wesentliche Features noch nicht, beispielsweise Authentifizierung und Verschlüsselung. Auch reaktives Routing fehlt noch, also das fallweise Bestimmen der Route an Stelle von vorab festgelegten Routing-Tabellen.

Links:

- <http://www.meshcom.com/>

6 Test

Der Test, der hier detailliert beschrieben wird, wurde während der Fachstudie im Nexus-Labor mit der Test-Hardware durchgeführt.

6.1 Hardware

Zuerst wurde folgende Hardware für den Test im Nexus-Labor eingerichtet:

- Zwei Rechner (x86) mit jeweils einer **Wistron CM9 Atheros AR5213A** (siehe [4.1.2.1](#)) WLAN-Karte
- Ein Laptop (x86) mit einer **Intel Wireless WiFi Link 4965AGN** (siehe [4.1.2.4](#)) WLAN-Karte

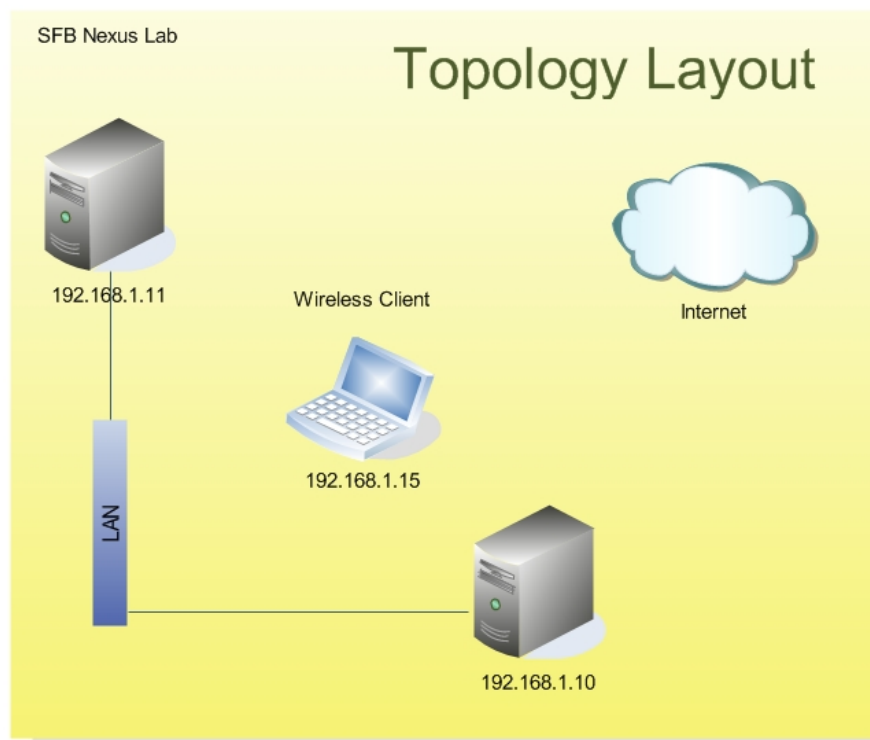


Abbildung 16: Topologie

6.2 Software

6.2.1 Betriebssystem

Auf den Rechnern mit der Wistron CM9 Atheros AR5213A WLAN-Karte wurde als Betriebssystem Fedora 6 mit Linux Kernel 2.6.18 und auf dem Laptop Windows XP verwendet.

6.2.2 Treiber für WLAN-Karten

In diesem Abschnitt wird beschrieben, wie man Treiber für WLAN-Karten auf den beiden Rechnern und Laptop installiert und die WLAN-Karten konfiguriert werden können.

Auf den Rechnern mit der Wistron CM9 Atheros AR5213A WLAN-Karte wurde die letzte Version des MadWifi-Treibers zuerst kompiliert und dann installiert.

Kompilieren des MadWifi-Treibers:

```
svn checkout http://svn.madwifi.org/madwifi/trunk madwifi
cd madwifi
make
```

Installieren des kompilierten MadWifi-Treibers (**root**-Rechte benötigt):

```
make install
```

Manuelles Laden des MadWifi-Treibers (**root**-Rechte benötigt):

```
modprobe ath_pci
```

Automatische Laden des MadWifi-Treibers beim Booten (**root**-Rechte benötigt):

```
mkdir /etc/modules.autoload.d/
echo ath_pci >> /etc/modules.autoload.d/kernel-2.6
```

Nachdem der MadWifi-Treiber geladen wurde (manuell oder automatisch), kann die WLAN-Karte konfiguriert werden. Die WLAN-Karte kann entweder manuell oder automatisch beim Booten konfiguriert werden.

Manuelles Konfigurieren der WLAN-Karte (**root**-Rechte benötigt):

```
ifconfig ath1 inet 192.168.2.1/24 # IP-Adresse
iwconfig ath1 essid mesh          # SSID="mesh"
iwconfig ath1 mode ad-hoc         # Ad-Hoc Modus einschalten
iwconfig ath1 channel 36          # 802.11a Kanal 36
iwconfig ath1 enc s:1234567890abc # 108 bit WEP-Passwort
```

```
# (13 Zeichen)
```

Damit die WLAN-Karte beim Booten von Fedora 6 automatisch konfiguriert werden kann, muss eine Konfigurationsdatei mit dem Namen **ifcfg-ath1** im Verzeichnis **/etc/sysconfig/network-scripts** angelegt werden.

Listing der Datei **/etc/sysconfig/network-scripts/ifcfg-ath1**:

```
cat /etc/sysconfig/network-scripts/ifcfg-ath1
DEVICE=ath1
ONBOOT=yes

BOOTPROTO=static
IPADDR=192.168.2.1
NETMASK=255.255.255.0

ESSID=mesh
MODE=ad-hoc
CHANNEL=36
KEY=s:1234567890abc
```

Nachdem die Datei **/etc/sysconfig/network-scripts/ifcfg-ath1** erstellt wurde, muss Init-Skript für Netzwerk-Dienste neugestartet werden (**root**-Rechte benötigt):

```
/etc/init.d/network restart
```

Den offiziellen Windows-Treiber für die Intel Wireless WiFi Link 4965AGN WLAN-Karte kann auf der Intel-Webpage heruntergeladen werden. Die Anleitung zur Installation und Konfiguration dieser WLAN-Karte findet man auch auf der selben Webpage (siehe [4.1.2.4](#)) und wird hier nicht weiter beschrieben.

6.2.3 olsrd

In diesem Abschnitt wird erklärt, wie man den olsr.org OLSR daemon kompiliert, installiert und konfiguriert. Außerdem wird hier auch gezeigt, wie man Plugins für den olsr.org OLSR daemon kompiliert und installiert.

Kompilieren des OLSR daemons:

```
cvs -d:pserver:anonymous@olsrd.cvs.sourceforge.net:\
    /cvsroot/olsrd login
cvs -z3 -d:pserver:anonymous@olsrd.cvs.sourceforge.net:\
    /cvsroot/olsrd co olsrd-current
cd olsrd-current
make
```

Installieren des OLSR daemons (**root**-Rechte benötigt):

```
make install
```

Im Folgenden wird demonstriert, wie man das HTTP Mini-Server Plugin **httpinfo** kompiliert und installiert. Das Plugin **httpinfo** ist ein kleiner und einfacher HTTP-Server und erlaubt es, z.B. die Routing-Tabelle eines Knotens in einem Mesh-Netzwerk zu erfassen.

Damit die Topology eines Mesh-Netzwerkes visualisiert werden kann, muss noch das Dot Data Generation Plugin **dot_draw** kompiliert und installiert werden. Das Plugin **dot_draw** ist auch ein kleiner Server. Wenn man eine TCP-Verbindung zu diesem Server aufbaut (z.B. mit **netcat** oder **telnet**), dann bekommt man die aktuelle Topology des Mesh-Netzwerkes in Form eines Dot-Graphes (siehe GraphViz <http://www.graphviz.org> und [de.wikipedia.org/wiki/DOT_\(GraphViz\)](http://de.wikipedia.org/wiki/DOT_(GraphViz))). Dieser Dot-Graph ist eine einfache Textdatei und kann mit Hilfe des Programms **dot** zu einem Bild konvertiert werden.

Es gibt noch andere zahlreiche Plugins für den olsr.org OLSR daemon. Sie alle können auf dieselbe Weise kompiliert und installiert werden.

Kompilieren des HTTP Mini-Server Plugins **httpinfo**:

```
cd lib/httpinfo
make
```

Installieren des HTTP Mini-Server Plugins **httpinfo** (**root**-Rechte benötigt):

```
make install
chcon -t textrel_shlib_t /usr/lib/olsrd-httpinfo.so.0.1
```

Der olsr.org OLSR daemon wird über die Datei **/etc/olsrd.conf** konfiguriert. Das vollständige Listing der Datei **/etc/olsrd.conf** ist im Anhang angegeben (siehe 8.1). Es muss vor allem das Netzwerk-Interface und Plugins konfiguriert werden.

Starten des olsr.org OLSR daemons (**root**-Rechte benötigt):

```
olsrd
```

Der olsr.org OLSR daemon kann auch automatisch beim Starten des Betriebssystems gestartet werden. Dafür muss ein Startup-Skript **/etc/init.d/olsrd** erzeugt werden. Das Listing der Datei **/etc/init.d/olsrd** kann man hier betrachten 8.2.

Nachdem die Datei erzeugt wurde, muss noch das System so konfiguriert werden, dass das Startup-Skript **/etc/init.d/olsrd** beim Booten ausgeführt werden kann (**root**-Rechte benötigt):

```
chmod 755 /etc/init.d/olsrd
chkconfig --add olsrd
```

Das Kompilieren und Konfigurieren des olsr.org OLSR daemons für Windows XP wird hier nicht erklärt, weil es ziemlich kompliziert ist. Wir haben den olsr.org OLSR daemon und die GUI zu ihm für Windows XP selbst kompiliert und konfiguriert. Um den olsr.org OLSR daemon zu kompilieren, wird cygwin mit gcc benötigt. Um die GUI für den olsr.org OLSR daemon zu kompilieren, wird Microsoft Visual C++ 2005 benötigt.

6.2.4 Visualisierung

Das Dot Data Generation Plugin **dot_draw** für den olsr.org OLSR daemon stellt die Topology eines Mesh-Netzwerkes in Form eines Dot-Graphes dar. Der Dot-Graph ist eine Textdatei und man kann aus dieser Datei die Topology nicht sofort sehen.

Um die aktuelle Topology mit einem Webbrowser online betrachten zu können, haben wir auf einem Linux-Rechner in unserem Mesh-Netzwerk einen Apache HTTP-Server installiert und ein CGI-Skript (Perl-Skript) entwickelt, das die aktuelle Topology des Mesh-Netzwerkes grafisch darstellt.

Installieren des Pache HTTP-Servers in Fedora 6 (**root**-Rechte benötigt):

```
yup install httpd
```

Das CGI-Skript, das die aktuelle Topology des Mesh-Netzwerkes vom Dot Data Generation Plugin **dot_draw** ausliest, ins Bild konvertiert und in eine Webseite integriert, finden Sie hier [8.3](#). Das CGI-Skript muss im Verzeichnis **/var/httpd/cgi-bin/** abgelegt und ausführbar gemacht werden.

Das CGI-Skript **topology.pl** benötigt noch das Paket GraphViz, konkret wird das Programm **dot** aus diesem Paket benötigt, um Dot-Graphen zu Bildren konvertieren zu können.

Installieren des GraphViz-Pakets (**root**-Rechte benötigt):

```
yup install graphviz
```

6.2.5 dhcpd

In unserem Test haben wir jedem Knoten in unsrem kleinen Mesh-Netzwerk IP-Adressen statisch vergeben. Mit 3 Knoten im Netzwerk ist der Aufwand dafür sehr gering. Wenn sich aber Knoten zum Mesh-Netzwerk dynamisch verbinden und verschwinden können oder wenn die Anzahl der Knoten im Mesh-Netzwerk sehr groß wird, dann kann man auf einem der Linux-Rechnern in unserem Mesh-Netzwerk einen DHCP-Server installieren. Dieser Knoten mit DHCP-Server muss natürlich ständig im Mesh-Netzwerk vorhanden sein.

Installieren des DHCP-Servers in Fedora 6 (**root**-Rechte benötigt):

```
yup install dhcp
```

Der DHCP-Server kann über die Datei `/etc/dhcpd.conf` konfiguriert werden.

Damit der DHCP-Server automatisch beim Booten gestartet werden kann, muss man folgendes Kommando ausführen (**root**-Rechte benötigt):

```
chkconfig dhcpd on
```

Manuelles Beziehen einer IP-Adresse vom DHCP-Server (**root**-Rechte benötigt):

```
dhclient ath1
```

6.2.6 Firewall

Damit der olsr.org OLSR daemon überhaupt korrekt funktionieren kann, müssen mehrere Ports in der Firewall von beiden Rechnern geöffnet werden.

Damit der OLSR-Protokoll funktionieren kann, muss der UDP-Port 698 für eingehende Pakete geöffnet werden (**root**-Rechte benötigt):

```
iptables -A RH-Firewall-1-INPUT -i ath1 -p udp \
    --sport 698 -j ACCEPT
```

Damit man auf den HTTP Mini-Server **httpinfo** eines Knotens zugreifen kann, muss der TCP-Port (hier 8080, kann in der `/etc/olsrd.conf` konfiguriert werden) des Servers für eingehende Pakete geöffnet werden (**root**-Rechte benötigt):

```
iptables -A RH-Firewall-1-INPUT -p tcp --dport 8080 \
    -m state --state NEW -j ACCEPT
```

Damit man auf den Dot Data Generation Server **dot_draw** eines Knotens zugreifen kann, muss der TCP-Port (hier 8081, kann in der `/etc/olsrd.conf` konfiguriert werden) des Servers für eingehende Pakete geöffnet werden (**root**-Rechte benötigt):

```
iptables -A RH-Firewall-1-INPUT -p tcp --dport 8081 \
    -m state --state NEW -j ACCEPT
```

Um die aktuelle Topology unseres Mesh-Netzwerkes betrachten zu können, muss man den TCP-Port 80 für eingehende Verbindungen öffnen (**root**-Rechte benötigt):

```
iptables -A RH-Firewall-1-INPUT -p tcp --dport http \
    -m state --state NEW -j ACCEPT
```

Wenn im Mesh-Netzwerk ein DHCP-Server verwendet werden soll, dann müssen auf jedem Knoten im Mesh-Netzwerk die UDP-Ports 67 und 68 geöffnet werden (**root**-Rechte benötigt):

```
-A RH-Firewall-1-INPUT -i ath1 -p udp \
    --sport 67:68 --dport 67:68 -j ACCEPT
```

Damit alle diese Ports automatisch beim Starten des Betriebssystems geöffnet werden können, muss in Fedora 6 die Datei `/etc/sysconfig/iptables` erweitert werden (**root**-Rechte benötigt):

```
cat >> /etc/sysconfig/iptables << EOF
-A RH-Firewall-1-INPUT -i ath1 -p udp \
    --sport 698 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 8080 \
    -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport 8081 \
    -m state --state NEW -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport http \
    -m state --state NEW -j ACCEPT
EOF

/etc/init.d/iptables restart
```

Die Konfiguration der Windows-Firewall auf dem Laptop wird hier nicht erklärt, siehe entsprechende Literatur und Artikel im Internet.

6.3 Inbetriebnahme

Nachdem alle Treiber, der OLSR daemon und Plugins kompiliert und installiert wurden und entsprechende Ports in Firewall geöffnet wurden, kann auf jedem Knoten der OLSR daemon gestartet werden.

6.4 Ergebnisse

In diesem Abschnitt werden wir die Ergebnisse von unserem Test präsentieren.

6.4.1 Topologie

Auf dem Bild 17 kann man die Topologie des Mesh-Metzwerkes betrachten. Der Knoten mit der IP-Adresse 192.168.2.15 ist der Laptop und die anderen beiden sind Linux-

Rechner. Der rechteckige Knoten auf dem Bild, ist der Knoten, der diese Topologie erzeugt hat.

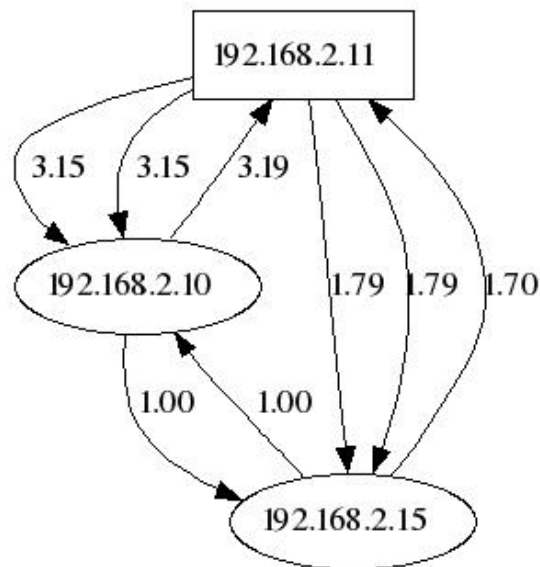


Abbildung 17: Topologie dargestellt mit dem olsr.org Plugin **dot_draw**

†Bedeutung der Zahlen?

TODO

6.4.2 Routing-Tabelle

Auf dem Bild 18 kann man die Routing-Tabelle eines Linux-Rechners betrachten



olsr.org OLSR daemon

[Configuration](#)
[Routes](#)
[Links/Topology](#)
[All](#)
[About](#)

Version: olsr.org - 0.5.5pre (built on 2007-10-29 14:42:12 on pcvs63.informatik.uni-stuttgart.de)
 OS: GNU/Linux
 System time: Mon, 19 Nov 2007 17:08:40
 Olsrd uptime: 20 day(s) 22 hours 04 minutes 43 seconds
 HTTP stats(ok/dyn/error/illegal): 748/0/0/0
 Click [here](#) to generate a configuration file for this node.

Variables

Main address: 192.168.2.11	IP version: 4	Debug level: 0	FIB Metrics: flat
Pollrate: 0.05	TC redundancy: 2	MPR coverage: 7	
Fisheye: Enabled	TOS: 0x0010	RtTable: 0x00fe/254	Willingness: 7
LQ extension: Enabled	LQ level: 2	LQ winsize: 100	

Interfaces

ath1
 IP: 192.168.2.11 MASK: 255.255.255.0 BCAST: 192.168.2.255
 MTU: 1472 WLAN: Yes STATUS: UP
 Olsrd is configured to run even if no interfaces are available

Plugins

Name	Parameters
olsrd_dot_draw.so.0.3	<input type="text" value="KEY, VALUE"/>
olsrd_httpinfo.so.0.1	<input type="text" value="KEY, VALUE"/>

Announced HNA entries

OLSR Routes in Kernel

Destination	Gateway	Metric	ETX	Interface
192.168.2.10	192.168.2.15	2	2.449	ath1
192.168.2.15	192.168.2.15	1	1.449	ath1

Links

Local IP	Remote IP	Hysteresis	LinkQuality	lost	total	NLQ	ETX
192.168.2.11	192.168.2.15	0.00	0.6832	100	1.00	1.47	
192.168.2.11	192.168.2.10	0.00	0.5149	100	0.57	3.45	

Neighbors

IP Address	SYM	MPR	MPRS	Willingness	2 Hop Neighbors
192.168.2.10	YES	YES	YES	7	<input type="text" value="IP ADDRESS"/> (1)
192.168.2.15	YES	YES	YES	3	<input type="text" value="IP ADDRESS"/> (1)

Topology Entries

Destination IP	Last Hop IP	LQ	ILQ	ETX
192.168.2.11	192.168.2.10		0.51	0.57
192.168.2.15	192.168.2.10		1.00	0.99
192.168.2.10	192.168.2.11		0.51	0.57
192.168.2.15	192.168.2.11		0.69	1.00
192.168.2.10	192.168.2.15		1.00	1.00
192.168.2.11	192.168.2.15		0.69	1.00

Abbildung 18: Routing-Tabelle eines Knotens dargestellt mit dem olsr.org Plugin **httpinfo**

7 Fazit

TODO: † Abschnitte 7.1 7.2 - detaillierte Beschreibung - Was sind die Gründe, warum genau diese Konfiguration ausgewählt wurde!!! Was sind bei den anderen Alternativen schlecht?

TODO

7.1 Übersicht

Dieser Abschnitt stellt eine Übersicht über aktuelle verfügbare Hardwareplattformen und Systemsoftware für WMN dar. Auf Abbildung 20 wird die Übersicht und unsere Bewertung der Hardware und auf Abbildung 22 wird die Übersicht und unsere Bewertung der Software dargestellt.

TODO: † Textuelle Zusammenfassung der Tabelle!

TODO

7.2 Empfehlung für ein geeignetes WMN

Um ein stabiles, für die Forschung geeignetes Mesh-Netzwerk aufzubauen, sind vor allem folgende Hardware zu empfehlen:

- PCs + WLAN Karten mit Atheros Chipsatz und externen Antennen (z.B. *Wistron CM9 Atheros AR5213A*)

Weiterhin kann das Mesh-Netzwerk zu einem heterogenen WMN erweitert werden, indem folgende WLAN-Router eingesetzt werden:

- Linksys WRT54G v1.0

Als Software-System für das aufzubauende WMN hat sich Linux als Betriebssystem und **olsrd** als Routing Software als besonders geeignet erwiesen. TODO: † Warum

TODO

Um das ganze Informatikgebäude abzudecken, würden ca. 30-45 PCs reichen. Das heißt ca. 10-15 PCs pro Stock. Kosten für die notwendige Hardware liegen damit unter der Budget-Grenze. TODO: † genauer Plan

TODO

	IEEE 802.11abgn	Ad-Hoc Modus	Treiber (Linux/Windows)	Open-Source Firmware	LAN-Anschluss	Sicherheit	Installation	Konfiguration	Mini-PCI Slot
Linksys WMP55AG	++	+	++	-	++	++	++	+	-
Netgear WAG311	++	+	++	-	++	++	++	+	-
D-Link DWL-A520	+	+	++	-	++	+	++	+	-
Gigabyte GN-WPEAG	++	+	++	-	++	+++	++	+	-
Intel PRO/Wireless 5000	+	+	+	-	++	+	++	+	-
D-Link DWL-AG530	++	+	++	-	++	+++	++	+	-
D-Link DWL-G550	++	+	++	-	++	+++	++	+	-
Wistron CM9 Atheros AR5213A	++	+	++	-	++	+++	++	+	-
Intel PRO/Wireless 3945	++	+	++	-	++	+++	+	+	-
Intel PRO/Wireless 2915	++	+	++	-	++	+++	+	+	-
Intel Wireless WiFi Link 4965AGN	+++	+	++	-	++	+++	+	+	-
Linksys WRT54G v1.0	++	+	-	++	+	+++	-	+	+
Linksys WRT55AG	++	+	-	+	+	+	-	+	++
Asus WL500G/GP	++	+	-	+++	+	+++	-	+	+
Netgear HR314	+	-	-	-	+	+	-	+	-

Abbildung 20: Übersicht und Bewertung von Hardware

	Betriebssysteme	Installation	Konfiguration	Visualisierung	Open-Source
OLSRD	+++	+	+	++	+
B.A.T.M.A.N.	+	+	+	-	+
Meshcom	+	+	-	+	-

Abbildung 22: Übersicht und Bewertung von Software

8 Anhang

8.1 olsrd.conf

Listing der Datei `/etc/olsrd.conf`:

```
#
# olsr.org OLSR daemon config file
#
# Lines starting with a # are discarded
#
# This file was shipped with olsrd 0.X.X
#

# This file is an example of a typical
# configuration for a mostly static
# network (regarding mobility) using
# the LQ extention

# Debug level (0-9)
# If set to 0 the daemon runs in the background

DebugLevel          0
LinkQualityFishEye 1
LinkQualityDijkstraLimit 2 6.0

# IP version to use (4 or 6)

IpVersion            4

# Clear the screen each time the internal state changes

ClearScreen          yes

# HNA IPv4 routes
# syntax: netaddr netmask
# Example Internet gateway:
# 0.0.0.0 0.0.0.0

Hna4
{
#   Internet gateway:
```

```

# 0.0.0.0      0.0.0.0
# more entries can be added:
# 192.168.1.0  255.255.255.0
}

# HNA IPv6 routes
# syntax: netaddr prefix
# Example Internet gateway:
Hna6
{
# Internet gateway:
# ::          0
# more entries can be added:
# fec0:2200:106:: 48
}

# Should olsrd keep on running even if there are
# no interfaces available? This is a good idea
# for a PCMCIA/USB hotswap environment.
# "yes" OR "no"

AllowNoInt      yes

# TOS(type of service) value for
# the IP header of control traffic.
# If not set it will default to 16

TosValue        16

# The fixed willingness to use(0-7)
# If not set willingness will be calculated
# dynamically based on battery/power status
# if such information is available

Willingness      7

# Allow processes like the GUI front-end
# to connect to the daemon.

IpcConnect
{
# Determines how many simultaneously
# IPC connections that will be allowed

```

```

# Setting this to 0 disables IPC

MaxConnections 2

# By default only 127.0.0.1 is allowed
# to connect. Here allowed hosts can
# be added

Host          127.0.0.1
#Host         10.0.0.5

# You can also specify entire net-ranges
# that are allowed to connect. Multiple
# entries are allowed

#Net          192.168.1.0 255.255.255.0
}

# Wether to use hysteresis or not
# Hysteresis adds more robustness to the
# link sensing but delays neighbor registration.
# Used by default. 'yes' or 'no'

UseHysteresis no

# Hysteresis parameters
# Do not alter these unless you know
# what you are doing!
# Set to auto by default. Allowed
# values are floating point values
# in the interval 0,1
# THRLow must always be lower than
# THR_High.

#HystScaling    0.50
#HystThrHigh    0.80
#HystThrLow     0.30

# Link quality level
# 0 = do not use link quality
# 1 = use link quality for MPR selection
# 2 = use link quality for MPR selection and routing
# Defaults to 0

```

```

LinkQualityLevel          2

# Link quality window size
# Defaults to 10

LinkQualityWinSize        100

# Polling rate in seconds(float).
# Default value 0.05 sec

Pollrate                   0.05

# Interval to poll network interfaces for configuration
# changes. Defaults to 2.5 seconds

NicChgsPollInt            3.0

# TC redundancy
# Specifies how much neighbor info should
# be sent in TC messages
# Possible values are:
# 0 - only send MPR selectors
# 1 - send MPR selectors and MPRs
# 2 - send all neighbors
#
# defaults to 0

TcRedundancy               2

#
# MPR coverage
# Specifies how many MPRs a node should
# try select to reach every 2 hop neighbor
#
# Can be set to any integer >0
#
# defaults to 1

MprCoverage                7

# Olsrd plugins to load

```

```

# This must be the absolute path to the file
# or the loader will use the following scheme:
# - Try the paths in the LD_LIBRARY_PATH
#   environment variable.
# - The list of libraries cached in /etc/ld.so.cache
# - /lib , followed by /usr/lib

# Example plugin entry with parameters:

#LoadPlugin "olsrd_dyn_gw.so.0.3"
#{
    # Here parameters are set to be sent to the
    # plugin. These are on the form "key" "value".
    # Parameters of cause, differs from plugin to plugin.
    # Consult the documentation of your plugin for details.

    # Example: dyn_gw params

    # how often to check for Internet connectivity
    # defaults to 5 secs
#   PlParam      "Interval"      "40"

    # if one or more IPv4 addresses are given, do a ping
    # on these in descending order to validate that there is
    # not only an entry in routing table, but also a real
    # internet connection. If any of these addresses could be
    # pinged successfully, the test was succesful, i.e.
    # if the ping on the 1st address was successful, the
    # 2nd won't be pinged
#   PlParam      "Ping"          "141.1.1.1"
#   PlParam      "Ping"          "194.25.2.129"
#}

LoadPlugin "olsrd_httpinfo.so.0.1"
{
    PlParam      "port"          "8080"
    PlParam      "Host"          "127.0.0.1"
    PlParam      "Net"           "129.69.210.0_255.255.255.0"
    PlParam      "Net"           "192.168.2.0_255.255.255.0"
}

LoadPlugin "olsrd_dot_draw.so.0.3"
{
    PlParam      "port"          "8081"

```



```

        PIParam          "accept" "127.0.0.1"
    }

# Interfaces and their rules
# Omitted options will be set to the
# default values. Multiple interfaces
# can be specified in the same block
# and multiple blocks can be set.

# !!CHANGE THE INTERFACE LABEL(s) TO MATCH YOUR INTERFACE(s)!!
# (eg. wlan0 or eth1):

Interface "ath1"
{
    # Olsrd can autodetect changes in NIC
    # configurations(IP address changes etc.).
    # This is Enabled by default and the interval
    # to poll for changes on is defined by
    # NicChgsPollInt.
    # This polling can be disabled pr. NIC by setting
    # AutoDetectChanges to no.

    AutoDetectChanges          yes

    # IPv4 broadcast address to use. The
    # one usefull example would be 255.255.255.255
    # If not defined the broadcastaddress
    # every card is configured with is used

    # Ip4Broadcast              255.255.255.255

    # IPv6 address scope to use.
    # Must be 'site-local' or 'global'

    # Ip6AddrType                site-local
    # IPv6 multicast address to use when
    # using site-local addresses.
    # If not defined, ff05::15 is used

    # Ip6MulticastSite           ff05::11

    # IPv6 multicast address to use when
    # using global addresses

```

```

# If not defined, ff0e::1 is used

# Ip6MulticastGlobal          ff0e::1

# Emission intervals.
# If not defined, RFC proposed values will
# be used in most cases.

# Hello interval in seconds(float)
HelloInterval      2.0

# HELLO validity time
HelloValidityTime  20.0

# TC interval in seconds(float)
TcInterval         5.0

# TC validity time
TcValidityTime     30.0

# MID interval in seconds(float)
MidInterval        5.0

# MID validity time
MidValidityTime    30.0

# HNA interval in seconds(float)
HnaInterval        5.0

# HNA validity time
HnaValidityTime    30.0

# When multiple links exist between hosts
# the weight of interface is used to determine
# the link to use. Normally the weight is
# automatically calculated by olsrd based
# on the characteristics of the interface,
# but here you can specify a fixed value.
# Olsrd will choose links with the lowest value.
# Note:
# Interface weight is used only when LinkQualityLevel is 0.
# For any other value of LinkQualityLevel, the interface

```

```

# ETX value is used instead.
# Weight 0

# If a certain route should be preferred
# or ignored by the mesh, the Link Quality
# value of a node can be multiplied with a factor
# entered here. In the example the route
# using 192.168.0.1 would rather be ignored.
# A multiplier of 0.5 will result in a small
# (bad) LinkQuality value and a high (bad)
# ETX value.
# Note:
# Link quality multiplier is used only when
# LinkQualityLevel is > 0.

# LinkQualityMult 192.168.0.1 0.5

# This multiplier applies to all other nodes
# LinkQualityMult default 0.8
}

```

8.2 olsrd Startup-Skript

Listing der Datei `/etc/init.d/olsrd`:

```

#!/bin/bash
#
# Startup script for the OLSR Daemon
#
# chkconfig: 235 16 84
# description: This script starts OLSRD
#              (Ad Hoc routing protocol)
#
# processname: olsrd
# config: %{_sysconfdir}/olsrd.conf
# pidfile: %{_localstatedir}/run/olsrd.pid

source /etc/init.d/functions
source /etc/sysconfig/network

# Check that networking is up.

```

```

[ ${NETWORKING} = "no" ] && exit 0

[ -x /usr/sbin/olsrd ] || exit 1
[ -r /etc/olsrd.conf ] || exit 1

RETVAL=0
prog="olsrd"
desc="Ad_Hoc_routing_protocol"

start() {
    echo -n "Starting_$desc_($prog):_"
    daemon $prog -d 0
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/$prog
    return $RETVAL
}

stop() {
    echo -n "Shutting_down_$desc_($prog):_"
    killproc $prog
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog
    return $RETVAL
}

reload() {
    echo -n "Reloading_$desc_($prog):_"
    killproc $prog -HUP
    RETVAL=$?
    echo
    return $RETVAL
}

restart() {
    stop
    start
}

case "$1" in
    start)
        start

```

```

        ;;
stop)
    stop
    ;;
restart)
    restart
    ;;
reload)
    reload
    ;;
condrestart)
    [ -e /var/lock/subsys/$prog ] && restart
    RETVAL=$?
    ;;
status)
    status olsrd
    ;;
*)
    echo $"Usage_${0}_{start|stop|restart|reload|\
condrestart|status}"
    RETVAL=1
esac

exit $RETVAL

```

8.3 topology.pl

Listing der Datei `/var/httpd/cgi-bin/topology.pl`:

```

#!/usr/bin/perl

use IO::Socket;

$TOP_PATH = "/var/www/html/topology";
$WWWPATH = "/topology";
$SERVER = "localhost";
$PORT = "8081";
$DOT_FILENAME = "topology.dot";
$IMAGE_TYPE = "png";
$IMAGE_FILENAME = "topology.$IMAGE_TYPE";
$IMAGE_BGCOLOR = "grey";
$IMAGE_SIZE = "35,20";

```

```

$remote = IO::Socket::INET->new(Proto => "tcp",
    PeerAddr => $SERVER, PeerPort => $PORT) ||
    die "couldn't connect to dot server\n";

open DOT_FILE, ">$TOP_PATH/$DOT_FILENAME" ||
    die "couldn't create $DOT_FILENAME\n";

while ($line = <$remote>) {
    print DOT_FILE $line;

    if ($line =~ /\}/i) {
        last;
    }
}

close DOT_FILE;

`dot -T$IMAGE_TYPE -Gsize=$IMAGE_SIZE -Gbgcolor=$IMAGE_BGCOLOR\
-o $TOP_PATH/$IMAGE_FILENAME $TOP_PATH/$DOT_FILENAME`;

$date = `date`;

print<<EOF
Content-type: text/html\r
\r

<HTML>
<HEAD>
<TITLE>Topology</TITLE>
</HEAD>
<BODY>

<CENTER>

<H1>Topology</H1>

<H2>$date</H2>

<IMG SRC="$WWW_PATH/$IMAGE_FILENAME">

<BR><BR>

<A HREF="http://pcvs63.informatik.uni-stuttgart.de:8080">
OLSRD-Seite von pcvs63

```

```
</A>  
</CENTER>  
</BODY>  
</HTML>  
EOF
```

Literaturverzeichnis

- [1] Nexus-Projekt. <http://www.nexus.uni-stuttgart.de>.
- [2] C.K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall, December 2001.
- [3] Corinna Elektra Aichele. *Mesh. Drahtlose Ad-hoc-Netze*. Open Source Press, Juli 2007.
- [4] B.A.T.M.A.N. <https://www.open-mesh.net/batman>.
- [5] MadWifi. <http://madwifi.org>.
- [6] OpenWRT. <http://openwrt.org>.
- [7] olsr.org. <http://www.olsr.org>.

9 Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

Stuttgart, den _____

Alexander Egorenkov

Sergey Telezhnikov

Valerij Schneider