
Copyright (C) Open Source Press

Corinna „Elektra“ Aichele

# Mesh

## Drahtlose Ad-hoc-Netze

Copyright (C) Open Source Press

Alle in diesem Buch enthaltenen Programme, Darstellungen und Informationen wurden nach bestem Wissen erstellt. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grunde sind die in dem vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor(en), Herausgeber, Übersetzer und Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht, auch nicht für die Verletzung von Patentrechten, die daraus resultieren können. Ebenso wenig übernehmen Autor(en) und Verlag die Gewähr dafür, dass die beschriebenen Verfahren usw. frei von Schutzrechten Dritter sind.
Die in diesem Werk wiedergegebenen Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. werden ohne Gewährleistung der freien Verwendbarkeit benutzt und können auch ohne besondere Kennzeichnung eingetragene Marken oder Warenzeichen sein und als solche den gesetzlichen Bestimmungen unterliegen.
Dieses Werk ist urheberrechtlich geschützt. Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches – oder Teilen daraus – vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlags in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

---

**Bibliografische Information Der Deutschen Nationalbibliothek**

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

---

© 2007 Open Source Press, München
Gesamtlektorat: Ulrich Wolf
Satz: Open Source Press (LaTeX)
Umschlaggestaltung: www.fritzdesign.de
Gesamtherstellung: Kösel, Krugzell

ISBN 978-3-937514-39-0 http://www.opensourcepress.de

Copyright (C) Open Source Press

Kapitel 4

# B.A.T.M.A.N. – Better Approach To Mobile Ad-Hoc Networking

Copyright (C) Open Source Press

Ausgehend von den Erfahrungen mit Freifunk-OLSR begannen die Entwickler aus der Freifunk-Community im März 2006 in Berlin damit, ein neues Routingprotokoll für drahtlose Meshnetzwerke zu entwickeln.

Alle bisher bekannten Routingalgorithmen versuchen, Routen entweder zu berechnen (proaktive Verfahren) oder sie dann zu suchen, wenn sie gebraucht werden (reaktive Verfahren). Das neue Protokoll B.A.T.M.A.N. berechnet oder sucht im Gegensatz zu diesen Protokollen keine Routen – es erfasst lediglich, ob Routen zu anderen Knoten existieren und überwacht ihre Qualität. Dabei interessiert es sich nicht dafür, wie eine Route verläuft, sondern ermittelt lediglich, über welchen direkten Nachbarn ein bestimmter Netzwerkknoten am besten zu erreichen ist, und trägt diese Information proaktiv in die Routingtabelle ein.

In der IP-Routingtabelle im Betriebssystem eines Computers ist grundsätzlich nur der nächstgelegene Router (oder ein Gateway) zu einem Ziel ein-

getragen. Ein Netzwerkknoten muss nur den jeweils nächsten direkt erreichbaren Zugangspunkt kennen, der mit dem Weiterleiten von IP-Paketen zu einem Ziel beauftragt werden kann. Kenntnisse über Zwischenstationen sind für den Absender irrelevant. Dieser hat üblicherweise auch keine Möglichkeit, den Weg seiner IP-Pakete zu beeinflussen.

Wichtig ist lediglich, dass auch der jeweils nächste Router wieder das richtige Gateway zum Ziel kennt. Solange diese Kette nicht unterbrochen wird (oder das IP-Paket im Kreis herumschwirrt, bis die Time-To-Live abgelaufen ist) erreichen die Daten ihr Ziel. Bei einem verbindungslosen Protokoll wie UDP, welches Daten nur in eine Richtung schickt, ohne eine Bestätigung (Acknowledgement) von der Empfängerseite zu erwarten, reicht es aus, wenn eine Route nur in einer Richtung funktionsfähig ist. Protokolle wie TCP dagegen erwarten eine Antwort von der Gegenseite – deshalb muss das Weiterreichen der IP-Pakete auch in der Gegenrichtung funktionieren. Es muss also nicht nur eine Route in eine Richtung, sondern auch eine Route in der Gegenrichtung existieren – was Anfänger oft vergessen, wenn sie von Hand Routingtabellen editieren. Üblicherweise nehmen IP-Pakete für den Hin- und Rückweg die gleiche Route, das ist aber nicht unbedingt notwendig.

OLSR – oder Link-State-Routing im Allgemeinen – wendet ein aufwendiges Verfahren an, um diese Vorgaben umzusetzen. Die einzige Entscheidung, die ein Link-State-Router beim Versenden oder Weiterleiten von Daten treffen kann, betrifft ebenfalls lediglich die Auswahl des nächsten Routers, der über eine direkte Verbindung per Funk oder per Kabel mit dem Weiterreichen des IP-Pakets beauftragt werden kann. Aber um diese einfache Entscheidung treffen zu können, treiben Link-State-Verfahren großen Aufwand: Das gesamte Netz wird mit Topologieinformationen geflutet, alle Routen von jedem Endpunkt des Graphen zu jedem anderen Endpunkt werden berechnet. Am Ende führt gerade die Tatsache, dass sich jeder einzelne Knoten ein eigenes Gesamtbild des Netzwerks errechnet, zu Unstimmigkeiten und in der Folge zu Kreisrouten. Die Topologiedatenbanken in einem drahtlosen Mesh mit einer nennenswerten Anzahl von Routern sind praktisch nie synchron.

Link-State-Protokolle in drahtgebundenen Netzen wie das verbreitete Protokoll OSPF vermeiden Kreisrouten, indem sie sicherstellen, dass die Topologiedatenbanken in allen Routern wirklich identisch sind. Die dazu verwendeten Verfahren versagen in einem drahtlosen Netzwerk. Bevor die Datenbanken in allen Routern tatsächlich synchronisiert sind, sind sie hoffnungslos veraltet und damit nutzlos.

Die Kenntnis über die Zwischenstationen einer Route ist nur dann relevant, wenn der Absender die Route vorgibt. In IP-basierten Netzen ist das ein Sonderfall; dieses Verfahren wird als Sourcerouting bezeichnet – die Informationsquelle (Source) gibt die Route vor. Da die Übersicht über die Netzwerktopologie in einem Mesh mit zunehmender Entfernung immer

Copyright © Open Source Press

unschärfer wird, ist eine solche Vorgehensweise in der Praxis jedoch wenig sinnvoll. Der Absender geht häufig von einer Route aus, die ungünstig ist oder bereits nicht mehr funktioniert. Beliebt ist Sourcerouting in Meshnetzwerken deshalb, weil sich Kreisrouten leicht entdecken lassen: Wenn in einer Sourceroutingtabelle ein Knoten mehr als einmal vorkommt, ist offensichtlich etwas faul.

## 4.1 Funktionsweise des B.A.T.M.A.N.-Algorithmus

Wird der B.A.T.M.A.N.-Routingdaemon in einem Router gestartet, sendet er in einem definierten Intervall per UDP-Broadcast sogenannte *Originatornachrichten* (*Originator-Messages*). Diese sind prinzipiell dasselbe wie Hello-Nachrichten bei Link-State-Protokollen. Der Unterschied zwischen Hellos und Originatornachrichten besteht darin, dass Hello-Nachrichten bei Linkstate-Protokollen nur lokal versendet werden, um Nachbarn in direkter Reichweite zu erkennen. B.A.T.M.A.N. flutet dagegen das ganze Mesh mit Originatornachrichten.

### 4.1.1 Die Originatornachricht

Eine Originatornachricht enthält die IP-Adresse des Urhebers (Originator), außerdem eine Sequenznummer, mit der die einzelnen Nachrichten vom Urheber fortlaufend nummeriert werden und eine TTL. Der Inhalt der ersten Originatornachricht eines B.A.T.M.A.N.-Nodes mit der Adresse A lautet sinngemäß: „Nachricht von Knoten A, Sequenznummer 0, TTL 50.“ Andere B.A.T.M.A.N.-Knoten erzeugen und senden ebenfalls eigene Originatornachrichten. Eine Originatornachricht von B.A.T.M.A.N. ist sehr klein, die typische Größe ist 10 Byte. Einschließlich des Overheads, der beim Versenden eines IP-Pakets immer mit anfällt, beträgt die gesamte effektiv übertragene Datenmenge üblicherweise 52 Byte. Im IP-Header eines UDP-Pakets ist immer auch die Adresse des Netzwerkknotens enthalten, welcher das Paket gerade sendet. Diese Information ist für den Protokollablauf wichtig.

B.A.T.M.A.N.-Router wiederholen die Originatornachrichten von anderen Nodes unter der Beachtung bestimmter Regeln – das Netz wird mit Originatornachrichten geflutet. Auf ihrer Reise durch das Mesh gehen die Originatornachrichten dann verloren oder verbreiten sich nur langsam, wenn sie auf ungünstige Bedingungen stoßen (schlechte Funkverbindungen, überlastete Router, viele Interferenzen durch ausgelastete Funkverbindungen). Auf guten, wenig belasteten Funkstrecken kommen sie schnell und mit weniger Verlusten voran.

Empfängt ein B.A.T.M.A.N.-Router ein weitergereichtes Originatorpaket von einem entfernten Knoten, sieht er, von welchem direkten Nachbarn er die

Copyright © Open Source Press

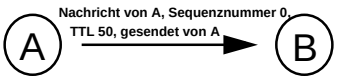
Originatornachricht bekommt. Damit erfährt er zum einen von der Existenz des anderen Knotens und sieht dabei außerdem, hinter welchem Nachbarn sich die Route befindet. Da die Originatornachricht einen Weg gefunden hat, existiert auch eine Route. Gibt es mehrere Routen zu dem entfernten Knoten, kommen die Originatornachrichten auf der besten Route schneller und/oder häufiger an. Liegen die verschiedenen Routen hinter verschiedenen Nachbarn, kann sich der Empfänger einfach für den besten Nachbarn als Gateway entscheiden, wenn er mit dem entfernten Knoten kommunizieren will. Der Empfänger muss sich lediglich merken, von welchem Nachbarn die Originatornachrichten des Senders am schnellsten und am zuverlässigsten eintreffen.

Zu jedem entdeckten Originator führt ein B.A.T.M.A.N.-Router eine Statistik über eine vordefinierte Anzahl der zuletzt empfangenen/erwarteten Originatornachrichten. Ein Node weiß somit, wie viele Originatornachrichten über welchen direkten Nachbarn tatsächlich eingetroffen sind und kann die Qualität der Verbindung beurteilen.

### 4.1.2 Bidirectional Link Check – Linkprüfung auf Bidirektionalität

Beim Weiterreichen von Originatornachrichten müssen die B.A.T.M.A.N.-Knoten einige einfache Regeln beachten. Dazu gehört, dass nur Originatornachrichten von Nachbarn weitergereicht werden, zu denen eine bidirektionale Kommunikationsverbindung besteht. Eine Route, die in beide Richtungen – auf dem Hin- und Rückweg – funktioniert, kann nur aus einer Kette von Funkverbindungen bestehen, die jeweils bidirektional arbeiten. Außerdem erwartet der Absender beim Transport von Daten, die nicht verbindungslos per Broadcast, sondern per Unicast übertragen werden, eine Empfangsbestätigung (Acknowledgement). Bleibt sie aus, wird die Nachricht mehrfach verschickt. Unidirektionale Funkstrecken sind deshalb für den Transport von Unicast-Paketen unbrauchbar. Da alle Protokolldaten von B.A.T.M.A.N. per Broadcast verschickt werden, muss nachgewiesen werden, dass die Linkstrecken, über die Originatornachrichten empfangen werden, bidirektional sind.

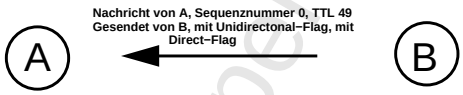
Angenommen, die Knoten A und B sind in einem Mesh direkte Nachbarn, die gegenseitig Daten empfangen können. Knoten B empfängt zum ersten Mal eine Originatornachricht von A. Da die Absenderadresse im IP-Header mit der Originatoradresse bei der empfangenen Originatornachricht identisch ist, weiß B, dass er das Paket direkt von seinem Originator bekommen hat:




Copyright © Open Source Press

B weiß nun, dass A existiert und dass A ein direkter Nachbar ist, dessen Pakete er empfängt. B weiß aber noch nicht, ob A seine Wiederholungen auch empfangen kann. Solange ein Knoten nicht weiß, ob die Verbindung zu einem direkten Nachbarn in beide Richtungen funktioniert (also bidirektional ist), muss er in die Originatornachrichten, die er von diesem Nachbarn wiederholt, eine Warnung hinzufügen, dass die Originatornachricht möglicherweise (oder erwiesenermaßen, falls der Link immer oder meistens unidirektional ist) aus einer einseitigen Kommunikationsbeziehung stammt. Anderenfalls könnten andere Knoten durch den Empfang dieser Aussendung glauben, dass hier eine benutzbare Verbindung besteht, und versuchen, Datenverkehr über die nur in eine Richtung funktionierende Verbindung zu routen.

Deshalb fügt B seiner Wiederholung der Originatornachricht von A ein sogenanntes *Unidirectional-Flag* hinzu. B wiederholt nun die Originatornachricht von A, nachdem er die TTL um 1 reduziert hat. Zusätzlich fügt B seinem Antwortpaket noch die Information `Is-Direct-Link` als Flag hinzu, damit A sicher sein kann, dass B auf ein Paket antwortet, dass er direkt von A empfangen hat:

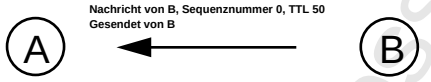



Wenn A die Wiederholung seiner eigenen Nachricht durch B empfängt, weiß er, dass es B gibt und dass B eine direkte Verbindung zu ihm hat. Anhand des gesetzten Flags `Is-Direct-Link` im Originatorpaket erkennt A, dass B seine Originatornachricht direkt von ihm selbst empfangen hat. A weiß nun durch den Empfang seiner eigenen Originatornachricht sicher, dass er einen direkten Nachbarn mit der Adresse B hat, der seine eigenen Originatornachrichten auf direktem Weg empfängt und wieder auf direktem Weg antworten kann. Damit ist aus der Sicht von A die direkte bidirektionale Verbindung zu B nachgewiesen.

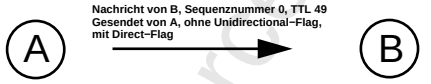
Hätte B das Originatorpaket von A nicht empfangen, hätte B es schließlich nicht wiederholt. Hätte B die Nachricht von A über einen Umweg über einen anderen B.A.T.M.A.N.-Knoten empfangen, wäre das `Is-Direct-Flag` nicht gesetzt. Wäre die Originatornachricht auf dem Rückweg verlorengegangen, wüsste A nichts davon, dass B seine Pakete empfängt. A trägt also eine Hostroute zu B in seine Routingtabelle ein. Empfängt jetzt A über B Originatornachrichten von entfernten Originatoren, wiederholt A diese, solange B das beste Gateway zu dem entfernten Originator ist.

A beteiligt sich am Fluten von Nachrichten, die er über B empfängt. Irgendwann schickt B nun ebenfalls aufgrund des festgelegten Intervalls für das Erzeugen von Originatornachrichten eine eigene Originatornachricht auf die Reise:

Copyright © Open Source Press




A weiß beim Empfang dieser Nachricht bereits, dass eine bidirektionale Verbindung zu B besteht und wiederholt diese Nachricht ohne die Unidirectional-Warnung, nachdem er die TTL um 1 reduziert hat. A setzt bei seiner Antwort ebenfalls das `Is-Direct-Link`-Flag:




B empfängt daraufhin die Wiederholung seiner eigenen Originatornachricht von A. Endlich wissen A und B, dass sie direkte Nachbarn sind, die sich gegenseitig sehen und in beide Richtungen miteinander kommunizieren können. Auch B trägt nun eine Hostroute zu A in seine Routingtabelle ein. Empfängt B Originatornachrichten von entfernten Originatoren, die von A weitergereicht werden, wiederholt B diese nun auch.

Angenommen, hinter Knoten B existiert noch der Knoten C. B und C haben eine bidirektionale Verbindung und beide wissen das auch schon, da sie den Bidirectional Link Check bereits erfolgreich durchlaufen haben. Wenn B die Originatornachrichten von A wiederholt, hört C die Übertragungen ebenfalls. So erfährt C, dass sich A hinter B befindet. C trägt B als Gateway zu A in seine Routingtabelle ein. Wiederholt B aus der anderen Richtung die Originatornachrichten von C, erfährt A von der Existenz von C. A trägt B als Gateway zu C ein.

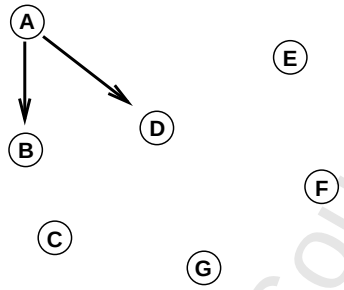
Da beide Funkverbindungen A–B und B–C daraufhin geprüft wurden, dass sie in beide Richtungen funktionieren, gibt es nun eine funktionierende Route A–B–C und eine funktionierende Route C–B–A. Über den Verlauf der beiden Routen wissen A und C jedoch nichts – sie wissen lediglich, dass B für sie das nächste Gateway ist. Wüsste A, dass C seine Originatornachrichten mit einer TTL von 50 versieht, könnte A sich wegen der TTL von 49 in den Originatornachrichten, die er von B empfängt, ausrechnen, dass C sich unmittelbar hinter B befindet. Eine Festlegung auf eine bestimmte TTL existiert in B.A.T.M.A.N. jedoch nicht. Damit kann jeder Knoten entscheiden, wie weit (im Sinne von Hops) er seine Anwesenheit dem Netz mitteilen möchte.

### 4.1.3 Eine Originatornachricht reist durch das Mesh

Die folgenden Grafiken verfolgen das Fluten eines kleinen Meshnetzes mit einer Originatornachricht der Station A. Die Qualität der Funkstrecken ist durch unterschiedlich kräftige Pfeile angedeutet.

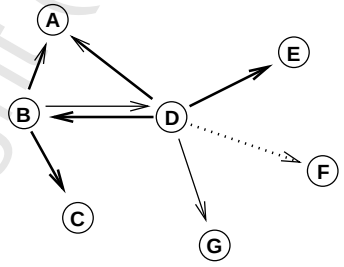
Copyright © 2009 Sure Press

Im ersten Schritt sendet A eine selbst erzeugte Originatornachricht, diese wird empfangen von B und D. Diese prüfen anhand der Urheberadresse und der Sequenznummer, ob sie diese Nachricht schon einmal gesehen haben. Wird die Nachricht zum ersten Mal empfangen, wiederholen sie die Ausstrahlung, nachdem sie die TTL reduziert haben:




B und D sehen die Nachricht zum ersten Mal und wiederholen sie. A sieht, dass B und D seine eigene Nachricht wiederholen. In diesem Beispiel ist die Bidirektionalitätsprüfung zwischen A, B und D bereits erfolgreich abgeschlossen. Daher wiederholen B und D die Originatornachricht von A ohne das Unidirectional-Flag. Nun sieht auch C die von B wiederholte Nachricht. Die Funkverbindung von B zu D funktioniert schlechter als von D zu B, sie ist gelegentlich unidirektional.

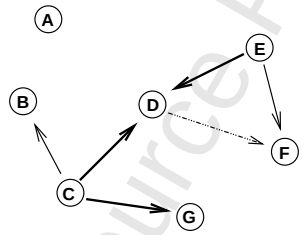
In unserem Beispiel geht das Paket auf dem Weg von B zu D verloren. Die Wiederholung durch D wird auch von E und B gesehen. Anhand der Sequenznummer erkennt B, dass die von D wiederholte Nachricht bereits bekannt ist und ignoriert sie. Die Übertragung von D zu G gelingt manchmal, in diesem Augenblick ist sie jedoch nicht erfolgreich:



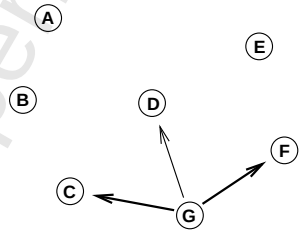

Nun wiederholen C und E die Nachricht. Die Verbindung von E zu F ist unzuverlässig, diesmal geht die Originatornachricht auf dem Weg von E zu F verloren. D sieht die Wiederholung der Nachricht von E, die bereits bekannt ist und somit ignoriert wird. Auch B und D ignorieren die Wiederholung von C. Sporadisch kommen Nachrichten von D bei F an, in der Beispielsituati-

Copyright © O'Reilly Source Press

on ist das der Fall. Der Link funktioniert aber nur von D nach F. Da F von D die Nachricht über einen unidirektionalen Link bekommt, wird sie von F ignoriert. Weitergereichte Originatornachrichten werden von B.A.T.M.A.N. nur beachtet, wenn sie von einem Nachbarn empfangen werden, der über eine bidirektionale Verbindung erreicht werden kann:




Bei der nächsten Übertragung kommt die Originatornachricht von A über G bei F an. C und D ignorieren die Wiederholung der bekannten Originatornachricht:




Nachrichten von A können über mehrere Wege bei F ankommen. Möglich sind unter anderem die Routen A–B–C–G–F, A–D–E–F, A–B–D–E–F. Wichtig für F ist nur, dass die Originatornachrichten von A entweder über G oder E zuerst bei ihm ankommen. Wenn Node F nun mit A kommunizieren möchte, muss er sich lediglich entscheiden, ob er G oder E mit dem Weiterreichen der IP-Pakete beauftragt. Diese Entscheidung kann F treffen, indem er eine Statistik führt, über welchen direkten Nachbarn wieviele Originatornachrichten von A zuerst eingetroffen sind (Tabelle 4.1).

Tabelle 4.1: Ranking-Tabelle

| **Stationen** | **Über welchen Nachbarn gesehen?** | |
|---|---|---|
| | **G** | **E** |
| **A** | 14 | 7 |
| **B** | 16 | 4 |
| **C** | 18 | 8 |
| **D** | 14 | 9 |

Copyright (C) Open Source Press

Fortsetzung:

| Stationen | Über welchen Nachbarn gesehen? | |
|---|---|---|
| **E** | 22 | 19 |
| **G** | 59 | 0 |

In der Beispielsituation ist die Route A–B–C–G–F zuverlässiger aber langsamer als beispielsweise A–D–E–F. Die kürzere Route kann zwar schneller sein, aber wenn die Mehrzahl der Pakete auf dieser Route nicht durchkommt, werden die über G eintreffenden Pakete häufiger das Rennen gewinnen.

## 4.2 Praxis mit B.A.T.M.A.N.

Der B.A.T.M.A.N.-Daemon 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.-Daemon `batmand` gibt es für Debian, OpenZaurus und OpenWRT.[1] 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 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`.

B.A.T.M.A.N. handelt automatisch UDP-Tunnel zu Internet-Gateways aus. IP-Pakete, die ins Internet gesendet werden, werden in UDP-Pakete verpackt und vom Gatewayclient an das Gateway geschickt. Auch wenn die IP-Pakete im Tunnel über mehrere Hops im Mesh transportiert werden, sieht der Tunnel für sie wie eine direkte Verbindung (Single-Hop) zum Gateway aus. Anfangs- und Endpunkt des Tunnels sind festgelegt und unabhängig davon, wie die Route im Mesh dazwischen gerade verläuft. Solange das Mesh in der Lage ist, eine funktionierende Hin- und Rückroute zwischen Client und Gateway bereitzustellen, ist die Anbindung an das Internet stabil, auch wenn sich der Routingpfad zwischen beiden Endpunkten häufig verändert.

[1] `http://downloads.open-mesh.net`

Copyright © Open Source Press

Durch die Verwendung eines Tunnels kann der Internet- oder Gatewaynutzer auf der Clientseite das Gateway aussuchen und bestimmen, dass das Gateway nicht umgeschaltet wird. Falls ein Gateway für Internetverkehr zum schwarzen Loch wird oder nur noch langsam reagiert, kann der Client selbst das Gateway abwählen und einen neuen bestimmen. Außerdem kann der Client entscheiden, wann, unter welchen Umständen und nach welchen Kriterien ein Gateway ausgesucht beziehungsweise gewechselt wird.

Hat der B.A.T.M.A.N.-Daemon erfolgreich einen Tunnel aufgebaut, legt er ein Tunnelinterface `tun0` an. Bei einer soliden Linuxdistribution sollten beim Einsatz des Kernelmoduls `tun` keine Probleme auftreten. Manchmal kommt es vor, dass der Devicenode `/dev/net/tun` nicht vorhanden ist oder nicht automatisch angelegt wurde. In diesem Fall kann der Deviceknoten von Hand mit Administratorrechten erzeugt werden:

```
mknod -m 600 /dev/net/tun c 10 200
```

### 4.2.1 Erster Start des Daemons

B.A.T.M.A.N. wird mit Administratorrechten von der Kommandozeile mit allen erforderlichen Parametern gestartet, eine Konfigurationsdatei gibt es nicht. Wie bei den meisten Kommandozeilenprogrammen gibt `batmand -h` einen kurzen Hilfetext aus. Ausführlichere Informationen erhält man mit der Option `-H`

Es können mehrere Interfaces angegeben werden, die natürlich funktionsfähig und konfiguriert sein müssen. Im einfachsten Fall genügt dann der Aufruf

```
batmand Interface1 Interface2 .. InterfaceN
```

Dabei wird das Default-Originatorintervall von einer Originatornachricht pro Sekunde und pro Interface verwendet. Der Routing-Algorithmus lässt sich über die Einstellung des Originatorintervalls an unterschiedliche Einsatzzwecke anpassen.

Wenn in einem Mesh schnelle Reaktionen auf Veränderungen der Topologie gewünscht sind und dafür etwas mehr Protokolloverhead in Kauf genommen werden kann, kann das Originatorintervall relativ klein gesetzt werden. Ein typischer Anwendungsfall sind kleinere Netze mit vielen mobilen Knoten.

Wenn geringer Protokolloverhead im Vordergrund steht, kann der Wert vergrößert werden. Ein sehr großes Mesh mit vielen hundert Knoten würde unter einem zu kleinen Originatorintervall leiden, hier fordert die große Originatoranzahl Abstriche bei der Reaktionsgeschwindigkeit.

Copyright (C) Open Source Press

Ist ein kürzeres oder längeres Originatorintervall erforderlich, kann das mit der Option `-o` angegeben werden, die Eingabe erfolgt in Millisekunden:

```
batmand -o 2000 Interface1 Interface2 ... InterfaceN
```

### 4.2.2 Den Netzwerkstatus im Debugging-Modus beobachten

Der B.A.T.M.A.N.-Daemon kann auch im Debugging-Modus mit der Option `-d` und einem Debuglevel von 0 bis 4 gestartet werden, um den Status des Netzwerks zu beobachten:

```
batmand -d Debuglevel Interface
```

Ohne die Debugging-Option oder mit der Angabe des Debuglevels 0 geht `batmand` beim Start sofort in den Hintergrund und routet auf den angegebenen Interfaces.

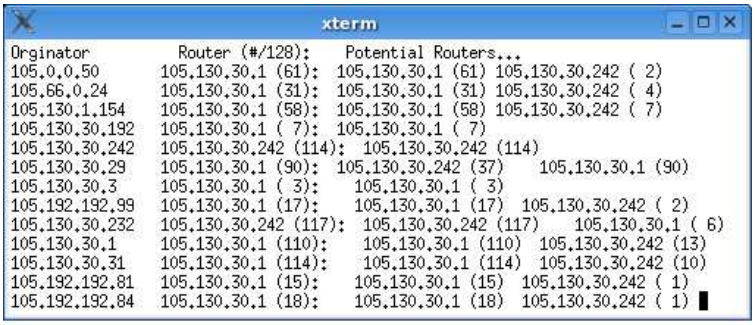



**Abbildung 4.9: Ausgabe im Debuglevel 1**

Wird das Programm in einem Debuglevel größer als Null aufgerufen, bleibt der Daemon im Vordergrund und gibt auf dem Terminal die somit angeforderten Informationen aus. Für Anwender sind nur die Debuglevel 1 und 2 interessant. Debuglevel 1 zeigt eine Liste aller erreichbaren Originatoren im Mesh an, außerdem die Nachbarn, über die zu den Originatoren geroutet werden kann und die Anzahl der jeweils empfangenen Originatornachrichten. Existieren Routen über mehrere Nachbarn zu einem Originator, werden diese in der Liste der `Potential Routers` angezeigt (Abbildung 4.9).

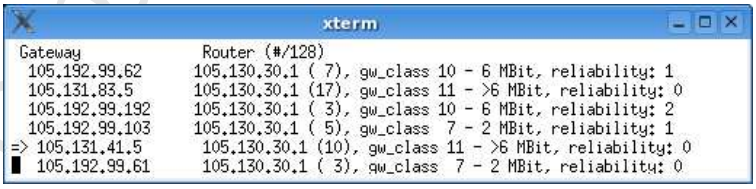



**Abbildung 4.10: Ausgabe von Debuglevel 2**

Copyright (c) Source Press

Mit der Debug-Option `-d 2` werden im Connect-Mode die vorhandenen Gateways aufgelistet. Abbildung 4.10 zeigt ein Beispiel, bei dem das Gateway mit der IP 105.131.41.5 ausgewählt wurde. Es offeriert dem Mesh einen Internet-Uplink mit mehr als 6 MBit. Von den 128 Originatornachrichten, die das Gateway auf die Reise geschickt hat, sind allerdings nur 10 auf dem lokalen Rechner angekommen, weshalb der erzielbare Datendurchsatz gering sein dürfte. Der Wert `Reliability` zeigt an, wie zuverlässig die Verbindung zum Gateway ist. Gateway und Gateway-Client kommunizieren regelmäßig darüber, ob der Tunnel aufrechterhalten werden soll. Schlägt diese Kommunikation fehl, erhöht sich der Reliability-Wert.

Der Verlauf der Route zu diesem Gateway kann unter Linux und BSD mit dem Befehl `ping -R` *`Ziel-IP`* überprüft werden. Dabei wird der Hin- und Rückweg angezeigt:

```
linux:~ #  ping -R 105.131.41.5
PING 105.131.41.5 (105.131.41.5) 56(124) bytes of data.
64 bytes from 105.131.41.5: icmp_seq=1 ttl=60 time=37.7 ms
RR:     105.130.30.0
        105.130.30.1
        105.130.1.67
        105.131.83.3
        105.131.41.5
        105.131.41.5
        105.131.41.3
        105.130.1.67
        105.130.30.1

64 bytes from 105.131.41.5: icmp_seq=2 ttl=60 time=10.2 ms  (same route)
```

Die Anzeige der Ping-Ergebnisse erfolgt fortlaufend. Ändert sich die Route, ist das sofort zu erkennen. Das `RR` in der Ausgabe steht für *Record Route.* Zu jedem erreichbaren Knoten kann so die Route geprüft werden, sofern Ping nicht durch Firewalleinstellungen blockiert wird.

### Debugging im Connect-Modus

Hat man den B.A.T.M.A.N.-Daemon bereits ohne das gewünschte Debuglevel gestartet oder möchte gleichzeitig oder abwechselnd die Ausgaben der Debuglevels 1 und 2 sehen, hat man die Möglichkeit, jederzeit beliebig viele Instanzen des B.A.T.M.A.N.-Daemon im Connect-Modus zu starten. Dazu gibt man den Schalter `-c` zusammen mit dem Schalter `-d` *`1-4`* für das Debuglevel an. Läuft ein mit dem Routing beschäftigter B.A.T.M.A.N.-Daemon im Hintergrund, kann man sich mit

```
batmand -c -d 1
```

Copyright (C) Open Source Press

den Routingstatus der routenden Instanz des `batmand` ansehen. Dabei wird die Anzeige fortlaufend aktualisiert. Soll der `batmand` nur einmalig eine Ausgabe erzeugen und sich dann beenden, kann man ihn mit der Option `-b` im Batch-Mode starten. Das ist interessant für Skripte, welche die Ausgabe des Daemons verarbeiten:

```
batmand -c -b -d 1
```

### 4.2.3 Routingklassen und Gatewayklassen

Der Daemon kennt unterschiedliche Gatewayklassen für Internet-Gateways, das heißt, die Gateways geben im Mesh bekannt, wieviel Bandbreite sie brutto zum Internet anbieten.

#### Gateway-Konfiguration

Ein B.A.T.M.A.N.-Gateway, das einen Internetzugang mit 2 MBit Bandbreite (Klasse 7) anbietet, startet man wie folgt:

```
batmand -g 7 eth1 wlan0
```

Auch das Annoncieren einer Route in ein anderes Netzwerk, Subnetz oder Hostroute analog zu den HNA-Announcements von OLSR ist möglich:

```
batmand -a 10.145.145.0/24 eth1 wlan0
```

Selbstverständlich kann man beide Optionen kombinieren und auch mehrere Host Network Announcements gleichzeitig vornehmen:

```
batmand -g 7 -a 10.11.12.13/32 10.22.33.0/22 eth1 wlan0
```

#### Gateway-Clients

Wird ein B.A.T.M.A.N.-Daemon gestartet, sucht er nicht automatisch eine Route zum Internet. Will man auf Clientseite angeben, dass er ins Internet routen soll, kann man mit der Option `-p` ein bevorzugtes Gateway angeben. Zusätzlich muss mit `-r` eine Routingklasse angegeben werden. Falls das bevorzugte Gateway nicht erreichbar ist, wird sich der B.A.T.M.A.N.-Daemon ein alternatives Gateway nach den vorgegebenen Kriterien der Routingklasse aussuchen. Drei Routingklassen werden angeboten.

Klasse 1 ist auf maximale Bandbreite zum Internet optimiert. Um das zu erreichen, sucht sich der B.A.T.M.A.N.-Daemon den besten Kompromiss

Copyright (C) Open Source Press

aus der vom Gateway angegebenen Gatewayklasse und der Verbindungsqualität über das Mesh zum Internet-Gateway. Bei Klasse 1 gibt er einem schnelleren Gateway den Vorzug, auch wenn dann die Verbindung weniger zuverlässig sein sollte:

```
batmand -r 1  ath0
```

Klasse 2 ist auf maximale Verbindungsqualität zum Internet-Gateway ausgerichtet. Der Routing-Daemon wird das am besten erreichbare Gateway wählen, auch wenn dieses nur langsamen Internetzugang anbietet. Das ist dann interessant, wenn eine langsame, dafür jedoch zuverlässige Internetverbindung Priorität haben soll, z. B. beim Chat, bei SSH-Verbindungen und VOIP-Anwendungen:

```
batmand -p 105.131.41.5 -r 2  ath0
```

Klasse 3 wählt automatisch das jeweils nächstgelegene Internet-Gateway. Diese Einstellung ist für mobile Knoten interessant, die sich viel im Mesh bewegen. Eine solche Verbindung ist schnell hergestellt, bedingt aber häufiges Umschalten des Gateways und führt deshalb oft zu Verbindungsabbrüchen bei Anwendungen, die nicht verbindungslos sind. Klasse 3 entspricht der Gatewayfunktionalität von OLSR mit den bekannten Problemen (Gateway-Switching, Verbindungsabbrüche) und Vorteilen (schneller Verbindungsaufbau).

Hat der Tunnelaufbau geklappt, findet man bei einem Blick in die Routingtabelle eine Defaultroute (Ziel: 0.0.0.0) über ein Interface mit dem Namen `tun`*`Nummer`*.

Die im Mesh verfügbaren Knoten sind jeweils mit einer Hostroute eingetragen. Hostrouten sind erkennbar an der Netzmaske 255.255.255.255. Die IP-Routingtabelle kann entweder mit dem Befehl `netstat -rn` oder mit `ip r` abgefragt werden. Auf Embedded-Systemen (OpenWRT) findet man meist das modernere `iproute`-Paket vor, auf Arbeitsplatzrechnern steht üblicherweise `netstat` zur Verfügung. Der Befehl `ip r` funktioniert nur auf Systemen, die das Paket `iproute` (oder `iproute2`) installiert haben. Die Ausgaben beider Programme sehen unterschiedlich aus, sind aber inhaltlich gleich:

```
linux:~ #  netstat -rn

Kernel IP Routentabelle
Ziel           Router        Genmask        Flags  MSS Fenster irtt Iface
105.130.77.80  105.130.30.1  255.255.255.255 UGH      0 0          0 eth2
105.130.1.67   105.130.30.1  255.255.255.255 UGH      0 0          0 eth2
(...)
105.130.30.1   0.0.0.0       255.255.255.255 UH       0 0          0 eth2
```

Copyright (C) Open Source Press

```
105.192.99.192 105.130.30.1  255.255.255.255 UGH      0 0          0 eth2
105.192.99.0   105.130.30.1  255.255.255.192 UG       0 0          0 eth2
105.0.0.0      0.0.0.0       255.0.0.0       U        0 0          0 eth2
0.0.0.0        0.0.0.0       0.0.0.0         U        0 0          0 tun0
```

```
linux:~ #  ip  r

105.130.77.80 via 105.130.30.1 dev eth2
105.130.1.67 via 105.130.30.1 dev eth2
105.131.41.1 via 105.130.30.1 dev eth2
(...)
105.130.30.1 dev eth2  scope link
105.192.99.192 via 105.130.30.1 dev eth2
105.192.99.0/26 via 105.130.30.1 dev eth2
105.0.0.0/8 dev eth2  proto kernel  scope link  src 105.130.30.38
default dev tun0  scope link
```

Eine Installation des Kernelmoduls `tun` ist nur dann unnötig, wenn ein B.A.T.M.A.N.-Knoten lediglich als Meshrelais arbeitet, selbst also keine Verbindung zu einem Internet-Gateway aufbaut und auch kein Internet-Gateway anbietet.

### 4.2.4 Visualisierungsserver

Visualisierungen von Meshnetzwerken sind beliebt und in der Praxis wegen der schnellen Übersicht nützlich. Doch bedingt durch das Konzept des B.A.T.M.A.N.-Algorithmus kennen die einzelnen Meshknoten die Gesamttopologie des Meshnetzwerks nicht. Deshalb haben sich die Entwickler eine Lösung ausgedacht, die es ermöglicht, Topologieinformationen über die Gesamttopologie des B.A.T.M.A.N.-Mesh zu ermitteln. Das Fluten des gesamten Netzwerks mit Topologieinformationen zum Zwecke der Visualisierung und das Erstellen einer Topologiedatenbank in jedem einzelnen Meshknoten wie bei einem Link-State-Protokoll würde das Gesamtkonzept von B.A.T.M.A.N. jedoch ad absurdum führen und den proaktiven Overhead wieder durch die Hintertür hereinbringen.

Statt die Topologieinformationen über das Mesh zu fluten und in jedem Knoten zu sammeln, kann ein leistungsfähiger Visualisierungsserver bei jedem routenden Daemon (egal ob Gateway-Client oder Gateway) mit der Option `-s` angegeben werden. Die einzelnen Knoten schicken ihre Informationen an den Visualisierungsserver. Der Server sammelt die Topologieinformationen und gibt sie in einem zu Graphviz kompatiblen Format aus, so dass die für Olsrd zur Verfügung stehenden Visualisierungslösungen auch mit B.A.T.M.A.N. funktionieren.

Das folgende Beispiel ruft `batmand` als Gateway-Client von Gateway 105.131 .41.5 mit der Routingklasse 2, aus, sorgt dafür, dass die erreichbaren Ori-

Copyright © Open Source Press

ginatoren und Router per Debuggingausgabe sichtbar werden und wählt 105.192.192.230 als Visualisierungsserver aus:

```
batmand -d 1 -p 105.131.41.5 -r 2 -s 105.192.192.230 ath0
```

Es muss kein Gateway angegeben werden, wenn dieses nicht explizit festgelegt werden soll. Die Angabe einer Routingklasse reicht aus. Umgekehrt muss aber bei der Angabe eines bevorzugten Gateways eine Routingklasse angegeben werden. Ist dieses Gateway nicht erreichbar, wählt der Daemon entsprechend der Routingklasse eine Alternative.

### 4.2.5 Firewalleinstellungen anpassen

`batmand` verwendet die Ports 1966 (UDP) für Protokollnachrichten und Port 1967 (UDP und TCP) für den Transport der IP-Pakete im UDP-Tunnel. Diese beiden Ports müssen für eingehenden und ausgehenden Datenverkehr durch die Firewall geöffnet sein. Die Portnummern können sich nach Erscheinen dieses Buches im Zuge der Standardisierung des B.A.T.M.A.N.-Protokolls und der offiziellen Vergabe durch die IANA (Internet Assigned Numbers Authority) noch ändern.

In Computern, auf denen der B.A.T.M.A.N.-Visualisierungsserver installiert wird, muss der UDP-Port 1968 geöffnet sein, um Topologie-Nachrichten empfangen zu können.

Wenn über einen B.A.T.M.A.N.-Gateway-Client lokale Computer (die beispielsweise am LAN-Port des Routers angeschlossen sind) ins Internet geroutet werden sollen, muss zum UDP-Tunnel hin NAT eingerichtet werden. Das folgende Beispiel ist betont einfach gehalten, und löscht im ersten Schritt mit dem Befehl `iptables -F` alle eventuell bereits vorhandenen Firewall-Regeln. 105.130.30.16 ist die IP-Adresse des Tunnelinterfaces am B.A.T.M.A.N.-Gateway-Client:

```
#!/bin/sh
iptables -F; iptables -t nat -F; iptables -t mangle -F
iptables -t nat -A POSTROUTING -o tun0 -j SNAT --to 105.130.30.16
```

Auf dem B.A.T.M.A.N.-Gateway muss ebenfalls zum Internet hin NAT eingerichtet werden, wie im Abschnitt 3.5.2 ab Seite 69 beschrieben.

### 4.2.6 Webinterface für B.A.T.M.A.N.

Für die Freifunk-Firmware gibt es ein Webinterface von Ludger Schmudde, mit dem der `batmand` mittels Webbrowser komfortabel konfiguriert und seine Funktion beobachtet werden kann. Die Installation wird im Abschnitt 6.5.1 ab Seite 156 beschrieben.

Copyright © Open Source Press

Copyright (C) Open Source Press

Kapitel 6

# Mesh-Betrieb auf SoHO-Routern

Bei einigen Routern aus dem SoHO-Bereich kann die Originalfirmware des Herstellers durch eine alternative Firmware ersetzt werden. Praktisch alle im Handel erhältlichen Router stellen eine Update-Funktion für das Einspielen einer neuen Firmware zur Verfügung. Bekannt sind hier vor allem die für diese Hardware entwickelten Linux-Distributionen OpenWRT und DDWRT. Auf der Basis von OpenWRT, Version „White Russian“, hat Sven-Ola Tücke eine für den Einsatz im Freifunk-Mesh spezialisierte Firmware-Distribution namens Freifunk-Firmware entwickelt.

Durch die Freifunk-Firmware wird die Installation eines für das Meshing vorbereiteten Betriebssystems in einem geeigneten Router so einfach wie ein Firmwareupdate des Herstellers. Die Freifunk-Firmware kann bei dem beliebten Router Linksys WRT54GL und ähnlichen Modellen über das Webinterface installiert werden.

## 6.1 Freifunk-kompatible Router

Voraussetzung für eine erfolgreiche Installation ist ein zur Freifunk-Firmware kompatibles Modell mit geeigneter Hardwarerevision. Viele Firmen verkaufen Geräte aus asiatischer OEM-Produktion unter ihrem Namen und mit firmentypischem Design, ohne diese selbst herzustellen oder zu entwickeln. Es gibt daher Geräte, deren Innenleben zu den Produkten anderer Firmen baugleich ist, da sie vom gleichen OEM-Hersteller stammen.

Leider wechselt der Inhalt der Gehäuse oft schneller als es für die Anwender wünschenswert wäre. Dabei kann sich sowohl das mitgelieferte Betriebssystem als auch das Hardwaredesign mit jeder Revisionsnummer vollständig ändern, ohne dass es äußerlich erkennbar ist, solange man nicht auf das Typenschild schaut. So war der Linksys WRT54GS bis zur Hardwarerevision 3.0 mit einem Flashspeicher von 8 MByte und einem RAM von 32 MByte vergleichsweise üppig ausgestattet. Die Hardware-Revision 4.0 enthält dagegen nur noch 4 MByte Flash und 16 MByte RAM – damit ist das Gerät zum Preis von rund 85 Euro nicht besser ausgestattet als der wesentlich billigere WRT54GL zum Preis von etwa 65 Euro. Die nachfolgenden Revisionen 5.0, 5.1 und 6.0 enthalten nur noch 2 MByte Flash und 16 MByte RAM – sie sind vollkommen uninteressant und funktionieren auch nicht mehr mit der Freifunk-Firmware.

Inzwischen scheinen einige Firmen realisiert zu haben, dass sich mit Open-Source-getriebener Hardware Geld verdienen lässt, weil das für einige Konsumenten kaufentscheidend ist. So stellte Linksys dem WRT54G nach dessen Umstellung von Linux (Hardwarerevisionen 1.0 bis 4.0) auf das proprietäre Betriebssystem VxWorks (Hardwarerevisionen 5.0 und höher) das Modell WRT54GL mit Linux zur Seite.

Eine Liste der von der Freifunk-Firmware unterstützten Router zeigt Tabelle 6.1. Dabei sind nicht alle der aufgeführten Geräte noch im Handel oder werden mit einer geeigneten Hardwarerevision angeboten.

Tabelle 6.1: Von der Freifunk-Firmware unterstützte Router

| Gerät | Bemerkung |
|---|---|
| Allnet 0277 | nicht mehr im Handel |
| Asus WL-500G | kaum noch anzutreffen |
| Asus WL-500G Deluxe | Mit USB 2.0, leider nicht mehr im Angebot |
| Asus WL-500G Premium | Mit USB 2.0, gut ausgestattet |
| Buffalo WHR-G54S | sehr empfehlenswertes und preiswertes Modell; Freifunk-Firmware kann nur per TFTP installiert werden |

Copyright © O'Reilly Verlag

Fortsetzung:

| Gerät | Bemerkung |
|---|---|
| Linksys WRT54G | alle Modelle mit Revisionsnummer 1.0 bis 4.0, neuere WRT54G-Modelle (höhere Versionsnummern) werden nicht mehr unterstützt. |
| Linksys WRT54GL | das empfohlene Gerät |
| Linksys WRT54GS | nur ältere Versionen bis Revisionsnummer 3.0 sind interessant |
| Linksys WAP54G | Accesspoint ohne Switch, nur 2 MByte Flash und 8 MByte RAM |
| Linksys WRT54G3G | eigentlich ein WRT54G mit UMTS-Card in einem Cardbus-Steckplatz |
| Motorola WR850G | in Deutschland selten im Handel |
| Siemens SE505 | nicht mehr im Handel, über Ebay zu bekommen |

Alle für den Betrieb mit der Freifunk-Firmware geeigneten Router basieren auf integrierten Chipsätzen der Firma Broadcom. Auch in einer Großstadt auf einem sehr hoch gelegenen Standort mit vielen Nachbarstationen läuft die WLAN-Hardware von Broadcom im Ad-Hoc-Modus stabil, wenn die IBSSID festgelegt wurde (siehe Abschnitt 5.4.2 auf Seite 115).

Üblicherweise verfügen die Geräte über eine MIPS-CPU mit 200 MHz, einen Flashspeicher von 4 MByte und 16 MByte RAM. Das sind dann auch die empfohlenen Systemanforderungen für die Freifunk-Firmware. Ein paar Geräte sind etwas üppiger mit Flashspeicher und/oder RAM und einer etwas schnelleren CPU ausgestattet. Manche Router haben auch nur 2 MByte Flash und 8 MByte RAM. Das ist eindeutig zu knapp bemessen und reicht höchstens für ein sehr minimalistisches System.

Gemeinsam ist allen Geräten, dass die WLAN-Interfaces sehr gut funktionieren und auch eine ordentliche Reichweite haben. Die Modelle, auf die jetzt näher eingegangen wird, sind aktuell im Handel erhältlich und haben mindestens 4 MByte Flash und 16 MByte RAM.

Über ein Paketmanagementsystem (iPKG, siehe Abschnitt 6.6) können in den Routern mit der Freifunk-Firmware komfortabel zusätzliche Programme installiert werden. Da sie auf der gleichen Hardwareplattform aufsetzen, ist das Angebot an Softwarepaketen für alle Modelle gleich. Nach der Installation der Freifunk-Firmware verbleiben noch knapp 2 MByte freier Speicherplatz bei Geräten mit 4 MByte Flash, die für die Installation weiterer Programme oder eigener Erweiterungen genutzt werden können.

Nicht für alle prinzipiell geeigneten und preiswerten WLAN-Geräte, die mit einer Linux-Firmware ausgestattet sind und in einem Mesh verwendet wer-

Copyright © Open Source Press

den können, gibt es ein schlüsselfertiges Firmware-Image von Freifunk. Dazu ist das Angebot einfach zu groß und verändert sich zu schnell. Hier kann die Linux-Distribution OpenWRT weiterhelfen, auf deren Grundlage die Freifunk-Firmware aufgebaut ist.

### 6.1.1 Linksys WRT54GL und WRT54G

Der WRT54GL ist zum Preis von rund 65 Euro das zur Zeit beliebteste Gerät für Community-Netzwerke und im Augenblick der Freifunk-Router schlechthin. Von den älteren Hardwarerevisionen des WRT54G unterscheidet sich der GL lediglich in einem weiter gesteigerten Integrationsgrad des verbauten Chipsatzes und dem Zusatzbuchstaben L für Linux auf dem Gehäuse. Bis zur Hardwarerevision 4.0 verwendete Linksys im Modell WRT54G Linux als Betriebssystem – danach wurde ein proprietäres Betriebssystem der Firma VxWorks und ein anderer Bootloader verwendet. Gleichzeitig halbierte Linksys die Größe des Flashspeichers von 4 MByte auf 2 MByte. Auch das RAM schrumpfte dabei von 16 MByte auf 8 MByte.

Nach einiger Zeit waren engagierte Bastler in der Lage, auch in diesem Gerät Linux zu installieren – doch die aktuellen Revisionen des WRT54G sind wegen der mageren Ausstattung die Mühe kaum wert. Zudem wird die Sparversion des WRT54G im Handel mitunter zu einem höheren Preis angeboten als der in jeder Hinsicht bessere WRT54GL.

Die Installation der Freifunk-Firmware lässt sich ganz einfach über das Webinterface bewerkstelligen. Der Router verfügt über einen 4-Port-Switch und einen Uplink-Port für den Anschluss eines DSL-Modems oder die Verbindung zum bestehenden Ethernet. Auf der Funkseite kann der Router mit zwei Antennen und einem eingebauten Diversity-Chip aufwarten, der beim Empfang von Signalen überprüft, über welche Antenne eine andere Station am besten zu empfangen ist. Dabei beherrscht der WRT54GL auf seinen Antennen so genanntes True-Diversity – das Gerät kann sich nicht nur beim Empfang, sondern auch beim Senden entscheiden, über welche Antenne es eine bestimmte Station am besten erreichen kann.

Die Sendeleistung lässt sich auf dem Webinterface der Freifunk-Firmware zwischen 1 und 84 mW einstellen. Die Empfangsleistung dieses Gerätetyps ist sehr gut, selbst mit den mitgelieferten Stummelantennen sind zwei Kilometer bei freier Sicht und freier Fresnel-Zone bei niedriger Datenrate problemlos möglich (siehe Seite 184 im Kapitel 7.6.3). Das Gerät verträgt einen weiten Eingangsspannungsbereich von rund 8 bis 20 Volt an seiner Kleinspannungsbuchse. Einfache Power-over-Ethernetlösungen lassen sich damit leicht realisieren, wie im Kapitel 5 beschrieben.

Bis vor kurzem war das mitgelieferte ungeregelte Netzteil von der billigen und stromfressenden Sorte. Inzwischen wird der WRT54GL mit einem kleinen und sparsamen, getakteten Schaltnetzteil geliefert. Beide Netzteilvari-

Copyright © Open Source Press

anten sind mit einer Abgabeleistung von 12 Watt für den Betrieb des kleinen Routers reichlich überdimensioniert. Der Accesspoint verbraucht bei 12 Volt Eingangsspannung typischerweise nur etwa 0,3 Ampere Strom.

Auf der Webseite von OpenWRT finden sich auch zahlreiche Vorschläge, wie der Router modifiziert werden kann. Engagierte Bastler löten sogar einen SD-Card-Slot ein und übertakten das kleine Gerät, wie das bei den PCs von Gamern üblich ist.

Unter `http://212.222.128.68/sven-ola/ipkg/_g+gl/` findet man das zu diesem Routermodell und alten Versionen des WRT54G passende Firmware-Image.

### 6.1.2 Buffalo WHR-G54S

Der Buffalo WHR-G54S ist zu einem Preis von etwa 40 Euro bedeutend preiswerter als der Linksys WRT54GL – dabei steht er ihm nicht unbedingt nach. Prozessorleistung, RAM-Ausstattung und Flashspeicher sind gleich. Wie der WRT54GL verfügt das Gerät über einen eingebauten 4-Port-Switch und einen sogenannten WAN-Port für den Uplink (üblicherweise über ein DSL-Modem oder Ethernet).

Etwas schwieriger als beim Linksys ist der Installationsprozess der Freifunk-Firmware. Versucht man es über das Webinterface der Original-Firmware, lehnt der Router das Freifunk-Image als ungültig ab. In diesem Fall kann man auf das Einspielen des Firmwareimages via TFTP ausweichen. Ist die Freifunk-Firmware erst einmal installiert, können weitere Firmware-Updates problemlos über das Webinterface vorgenommen werden.

An einigen Stellen hat Buffalo an der Hardware gespart: Es gibt nur eine externe Antenne und eine billige Behelfsantenne im Gehäuse. Das ist aber beim Betrieb mit einer leistungsfähigen Außenantenne nicht unbedingt ein Nachteil. Unter manchen Gesichtspunkten ist der Buffalo WHR-G54S sogar besser als der WRT54GL. Bei gleicher Funktionalität ist das Gerät wesentlich kleiner und sieht zumindest subjektiv besser aus. Ebenso wie der WRT54GL arbeitet der Buffalo stabil im Ad-Hoc-Modus und hat eine sehr gute Reichweite.

Der WHR-G54S verbraucht wenig Strom aus der Steckdose, da er mit einem effizienten und modernen, getakteten Steckernetzteil geliefert wird. Auf eine aufwendige Spannungsaufbereitung im Gehäuse des Routers wie beim WRT54GL wurde bei diesem Modell verzichtet. Der Gleichspannungseingang des WHR-G54S braucht eine gut geregelte Kleinspannung von 3,3 Volt. Da die Spannung sehr viel kleiner ist, ist die Stromstärke höher als beispielsweise beim WRT54GL. Ein einfaches Versorgen des Routers per selbstgebasteltem PoE ohne zusätzlichen getakteten Spannungswandler ist deshalb beim WHR-G54S nicht möglich. Für den Betrieb eines Buffalo sollte man deshalb eine Steckdose in der Nähe haben.

Copyright © Open Source Press

Damit eignet sich das Gerät nicht so gut für den Betrieb auf einem Mast im Freien. Da Kabel im Freien verwittern, ist Netzstrom auf einem Mast ein Risikofaktor. Für den Betrieb in der Wohnung oder am Fensterbrett ist der „Büffel“ dagegen sehr gut geeignet. Wenn man eine PoE-Lösung für weniger als 20 Euro Materialkosten bauen kann und will, ist das kleine Gerät aber auch für Außeneinsätze interessant.

Unter `http://212.222.128.68/sven-ola/ipkg/_trx/` findet man die passende Firmware.

### 6.1.3 Asus WL500G Premium

Der Asus WL500G Premium ist ein mit zusätzlichen Funktionen ausgestatteter WLAN-Router zum Preis von etwa 90 Euro. Das Gerät besitzt einen Prozessor mit 266 MHz, 8 MByte Flash und 32 MByte RAM. Das hervorstechendste Merkmal ist das Vorhandensein von zwei schnellen USB-2.0-Anschlüssen. Damit kann man an das Gerät externe Festplatten, Speichersticks, zusätzliche WLAN-Karten, Drucker oder eine Soundkarte anschließen. Der WL500G Premium ist also ein Router mit vielen Möglichkeiten, der jedoch nur über einen externen Antennenanschluss verfügt. Das Gehäuse ist für einen SoHO-Router relativ groß.

Der Router eignet sich für den Betrieb über Power over Ethernet nur, wenn ein zusätzlicher externer Spannungswandler eingesetzt wird, da er mit 5 Volt Eingangsspannung arbeitet. Zusätzliche Komponenten, die ihre Energie über einen der beiden USB-Anschlüsse beziehen, treiben den Stromverbrauch des Routers natürlich in die Höhe. Für die Montage an einem Mast ist das Gerät eigentlich viel zu schade.

Dieser Router arbeitet mit dem gleichen Firmware-Image wie der Buffalo WHR-G54S. Die Installation der Firmware kann ebenso wie beim Buffalo WHR-G54S nicht über das Webinterface erfolgen. Durch Drücken des Reset-Knopfes kann man den TFTP-Server des Bootloaders aktivieren, der unter der IP 192.168.1.1 zu erreichen ist. Nun kann per TFTP-Upload das Firmware-Image mit einem TFTP-Client installiert werden. Ist die Installation abgeschlossen, startet der Asus nicht automatisch neu. Nachdem man fünf Minuten gewartet hat, kann man das Gerät vom Strom trennen. Beim nächsten Start bootet das Gerät dann in die Freifunk-Firmware.

## 6.2 Installation der Freifunk-Firmware

Selbstverständlich erlischt bei allen im Folgenden beschriebenen Eingriffen die Herstellergarantie und man handelt auf eigene Gefahr.

Beim Installieren eines Firmware-Image muss dafür Sorge getragen werden, dass der Flashprozess nicht vorzeitig unterbrochen wird (jemand zieht aus

Copyright © O'Reilly Open Source Press

Versehen den Mehrfachstecker aus der Dose, eine Steckdose hat einen Wackelkontakt, es wird nicht gewartet, bis der Flashprozess abgeschlossen ist). Dabei ist es gleichgültig, ob die Firmware per Webinterface oder per TFTP-Upload installiert wird. Schlägt die Installation der Firmware wegen einer vorzeitigen Unterbrechung fehl, hat man das Gerät möglicherweise in einen hilflos blinkenden Briefbeschwerer aus thermoplastischem Kunststoff verwandelt – „gebrickt“ (von engl. Brick = Ziegelstein) oder „zerflasht“, wie es im Jargon heißt. Bei einem Stromausfall während des Flashvorgangs kann es passieren, dass das Gerät auch den Bootloader beschädigt, während die Spannung zusammenbricht[1]. Außerdem sollte man überprüfen, ob die zu installierende Firmwaredatei vollständig und unbeschädigt ist und tatsächlich zu dem Router passt.

Im Downloadverzeichnis der Freifunk-Firmware befindet sich eine Textdatei mit der Bezeichnung `md5sum.asc`, diese enthält die Prüfsummen der Firmware-Images, die einen Test auf Unversehrheit ermöglichen. Um die Prüfsumme der heruntergeladenen Dateien zu ermitteln, steht unter Linux das Programm `md5sum` zur Verfügung. Windows-Anwender können das Programm `md5sum.exe` verwenden, das ebenfalls auf dem Server liegt.

Beim ersten Flashen der Firmware sollte man den PC unbedingt per Ethernetkabel mit einem Ethernetport im Switch des Routers verbinden und nicht über WLAN flashen.

### 6.2.1 Firmware-Installation auf Linksys WRT54GL

Die Installation ist bei diesem Gerät sehr leicht und erfolgt am einfachsten über das Webinterface des Routers. Router und Arbeitsplatzrechner werden mit einem Patchkabel verbunden. Der WRT54G/GL hat fünf Ethernetschnittstellen, eine WAN-Buchse und vier LAN-Buchsen. In eine der vier Ethernetbuchsen (Ports) wird das Patchkabel gesteckt.

Im Neuzustand verwendet der Router die IP-Adresse 192.168.1.1 und weist einem per Ethernetkabel angeschlossenen Computer per DHCP eine IP-Adresse zu, wenn der PC entsprechend eingerichtet ist. Ansonsten kann man auch der Netzwerkkarte manuell eine IP-Adresse zwischen 192.168.1.2 und 192.168.1.254 statisch zuweisen. Eine eventuell vorhandene WLAN-Karte im PC sollte für die Firmwareinstallation sicherheitshalber deaktiviert werden, damit man nicht aus Versehen den Router per WLAN zu flashen versucht – auch hier gibt der Router per DHCP Adressen aus.

Jetzt kann man mit einem Webbrowser die URL `http://192.168.1.1` öffnen. War die Netzwerkkonfiguration erfolgreich, befindet man sich nun im Webinterface des Routers. Sollte die Verbindung partout nicht klappen, hilft

[1] Führt man das Flashen in einem Schwellen- oder Entwicklungsland aus, tut man also gut daran, das betreffende Gerät direkt aus einer geladenen Batterie mit der passenden Spannung zu speisen, da man nie weiß, wann das nächste Mal der Strom ausfällt.

Copyright © Open Source Press