
Free Software for Wireless Community Networks

René Ejury
FOSS.IN
Bangalore Dec 2007



preface

a **lot of text** will fill the following slides

- **don't read it, listen**
- you can find the presentation online soon after this session at <http://bangalore-wireless-mesh.absorb.it>

maybe you are familiar with wireless mesh networks...

- have a look at <http://bangalore-wireless-mesh.absorb.it> and become part of the local mesh (ssid: freifunk.foss.in)

about: me

free IT Consultant, Teacher and Free Software Developer

joined Freifunk (Free Community Network) movement in 2004

- member of Opennet Rostock, Germany
- maintainer of the Opennet Firmware since 2005

other projects:

- setting up and maintaining mesh installations
- develop Thunderbird / Seamonkey extension 'Virtual Identity'
(www.absorb.it/virtual-id)

currently in India cooperating with janastu.org to establish a wireless mesh

about: motivation

might introduce you of all details of the Opennet Firmware

- **but: won't help you**
 - current version is specialized for the Opennet Rostock
 - German language only
 - after two years learning it's time to do a rewrite, and this time do it right!

SO:

- **I'll introduce you to the field of Free Software and Mesh Networking**
 - credits will go to all the developers in the community

about: motivation

motivate you to

- establish community mesh networks in India
- join a development group regarding your interests and possibilities
- start developing your own Mesh Firmware

and: there will be surely a new version of the Opennet Firmware soon

- based on OpenWrt Kamikaze
- completely splitted in task-oriented packages
- suitable for broader usage
 - english language support, separated configuration

you might join the development at http://trac.on-i.de/on_firmware_ng

Today's Menu

Mesh Introduction

- Mesh Networks – Ad-Hoc Networks
- The Freifunk Mesh / The Opennet Mesh

Software Overview

- OpenWrt
- Freifunk Firmware
- Opennet Firmware
- OLSR
- B.A.T.M.A.N

Build your own Mesh

- build your own Firmware
- join the Free Software development

Mesh Introduction

- Infrastructure vs. Ad-Hoc
- The Freifunk Network
- The Opennet Network

Infrastructure vs. Ad-Hoc

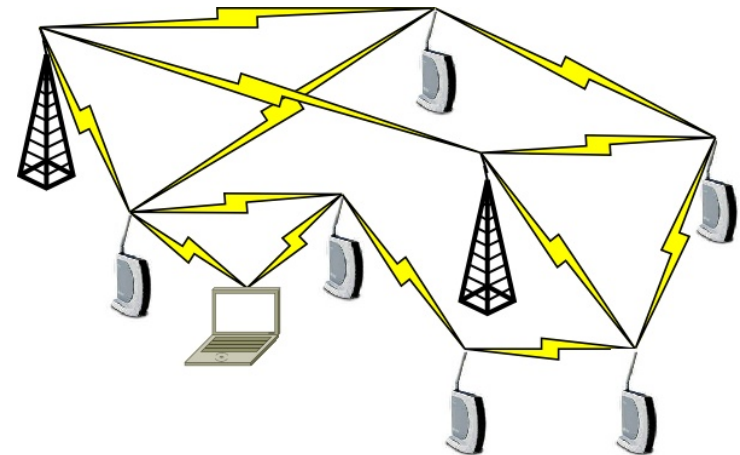
infrastructure = managed mode

- AP (master), STAs (clients)
- share same SSID (network name)
- **nothing special**



mesh networks = ad-hoc mode

- independent basic service set's (IBSS)
- BSSID – Basic Service Set Identifier uniquely identifies a BBS
- **every wireless device in range can communicate with any other** if they share the same BSSID



Problems of Ad-Hoc mode

“All Beacon and Probe Response frames carry a Timestamp field. A STA receiving such a frame from another STA in an IBSS with the same SSID shall compare the Timestamp field with its own TSF time. If the Timestamp field of the received frame is later than its own TSF time, the STA shall adopt all parameters contained in the Beacon frame.”
<http://standards.ieee.org/getieee802/802.11.html>, page 128.

broken drivers, bad implementation of ad-hoc standard
may result in **switching the channel, netsplit** etc.

most community **mesh networks use a (non-standard) fixed BSSID**
drivers are available for Broadcom and Atheros chipsets

You can't build a reliable 'ad-hoc' network with Laptops!

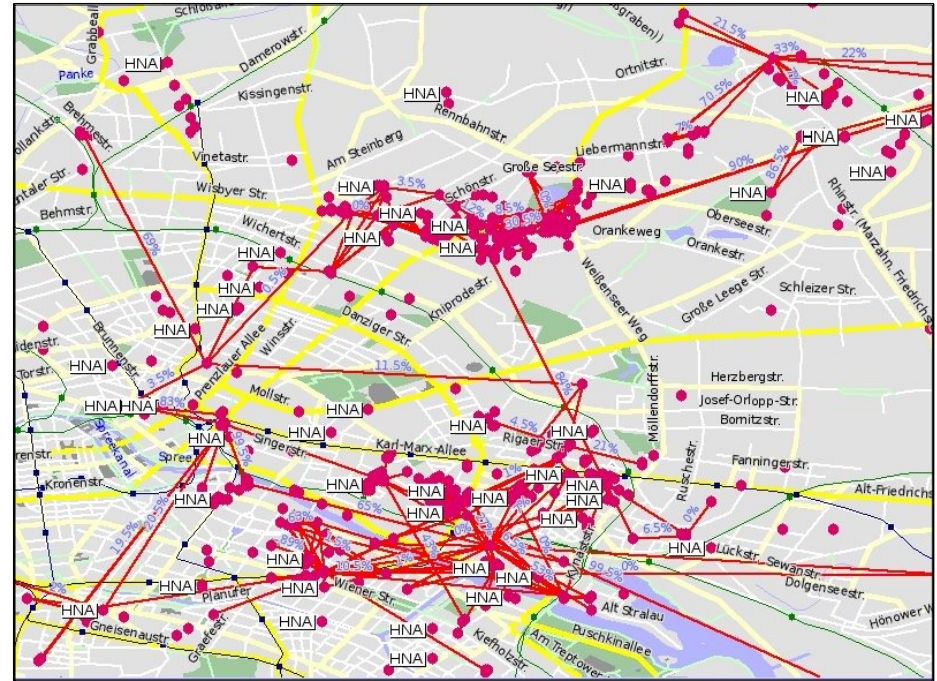
The Freifunk Network

Berlin

- Oct. 2002 BerLon conference
 - pico peering agreement
- network started around 2004
- currently ~500 nodes (i.e. households)

biggest community mesh network worldwide(?)

- similar networks in Germany
 - for instance Leipzig, with ~300 nodes
- other around the world, widespread use of the Freifunk Firmware
 - in India in Dharamsala, Mumbai, Goa and Trivandrum (test setups)



Freifunk Network Topology

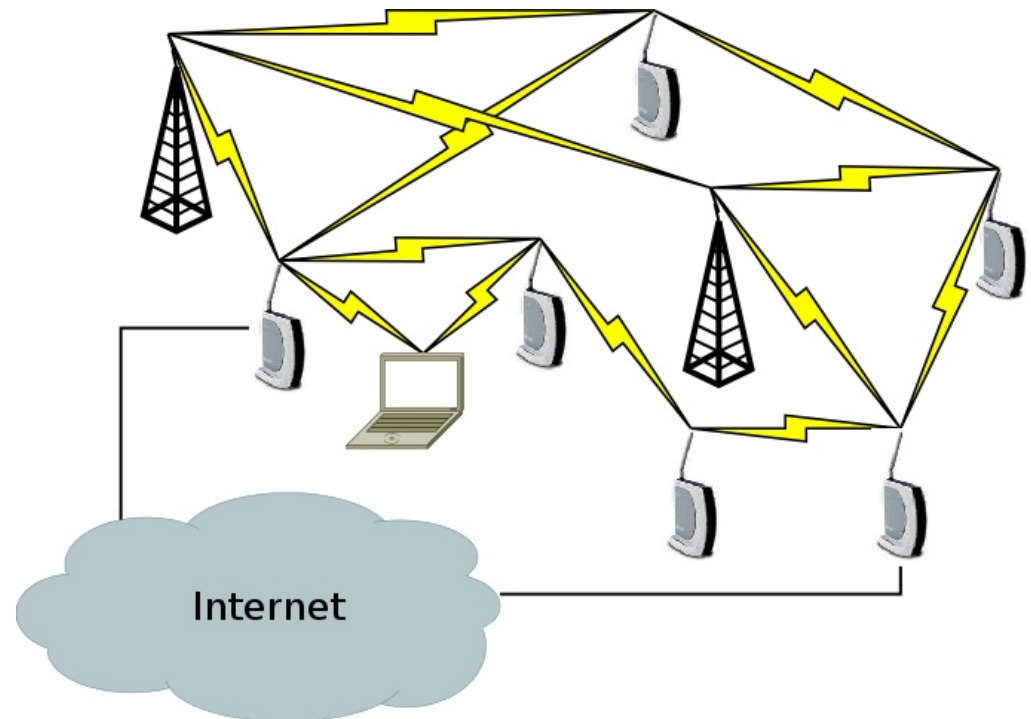
nodes are mostly **Wireless Routers** of all kind

Internet is donated, no special membership, no accounting required

routing done by OLSR
(layer 3)

'nearest' Internet Gateway is used

no encryption on layer 2



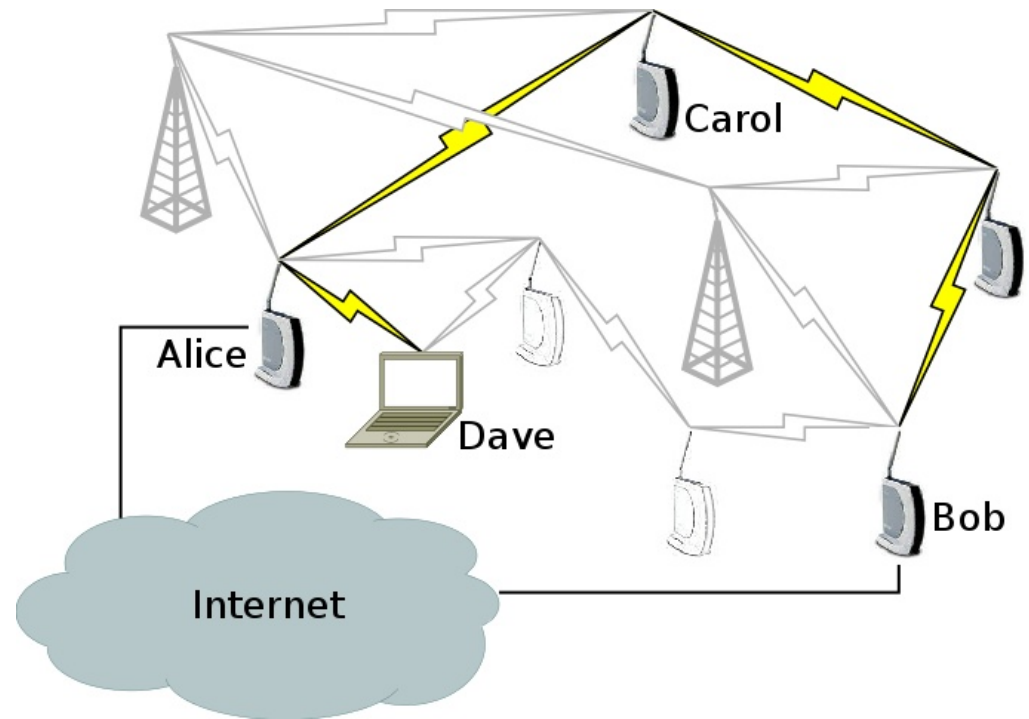
Freifunk Network Problems

no information about bandwidth / services of Gateways, **no Gateway selection**

IP of Internet-donators is used for external communication of other Freifunk users

used **Gateway might change continuously**, traffic gets interrupted

(OLSR routing loops)

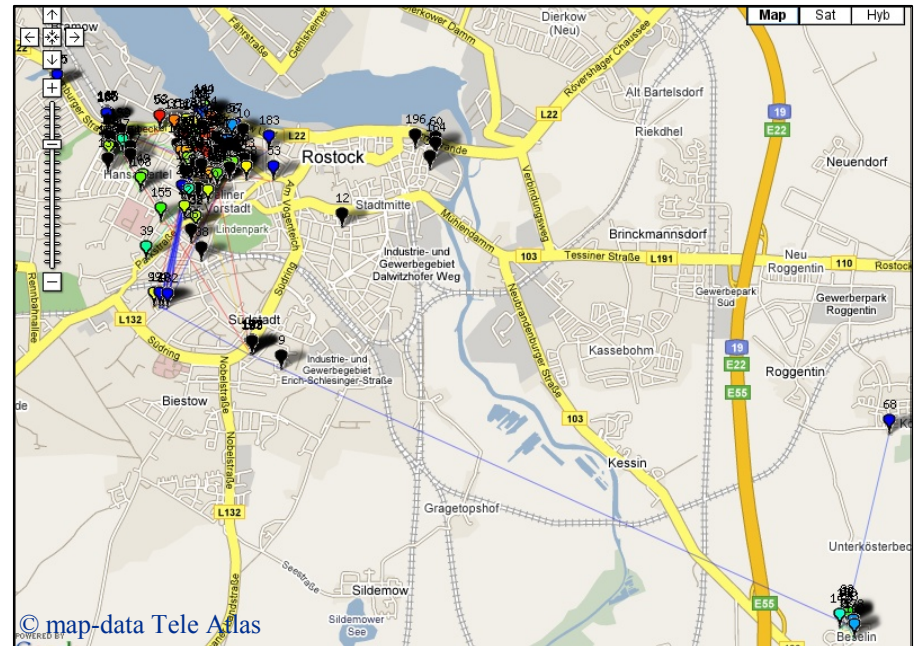


The Opennet Network

Rostock, north of Berlin, Germany

- 2004: no DSL
- started 2005
- now ~200 nodes, ~150 members
- **high density** in one area
~100 APs / km²

Internet support for village nearby



no 'Freifunk' network as by usage of the Freifunk Firmware,
but a Freifunk Network by building free information infrastructure

Opennet Topology

mostly APs, OLSR routing,
no encryption on layer 2

Internet (at the beginning)
paid by the community,
required accounting

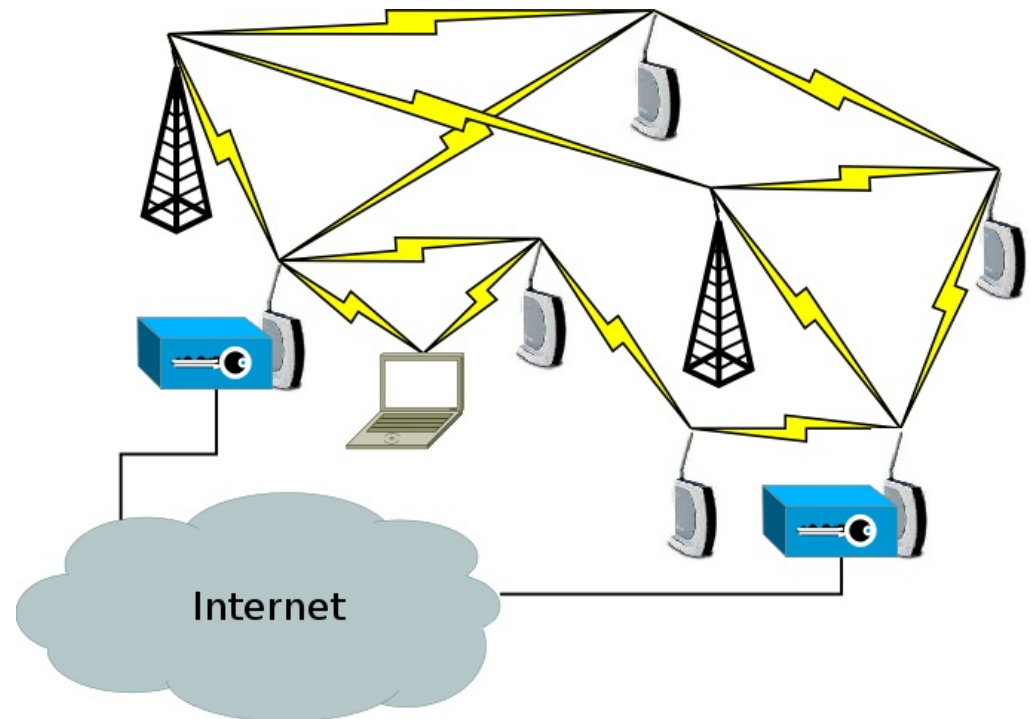
- Internet only available
through OpenVPN
over Opennet Gateways

good:

- user traffic encrypted
- selection of Gateways

bad:

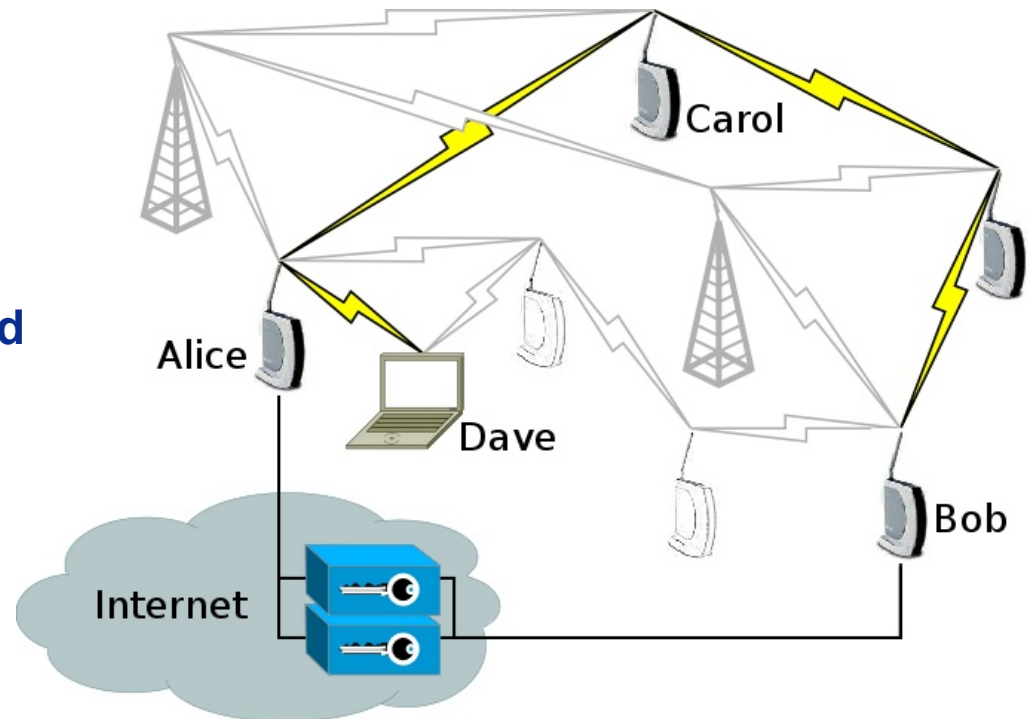
- special infrastructure required



Opennet User-Gateways

two central Gateways used for user-donated bandwidth

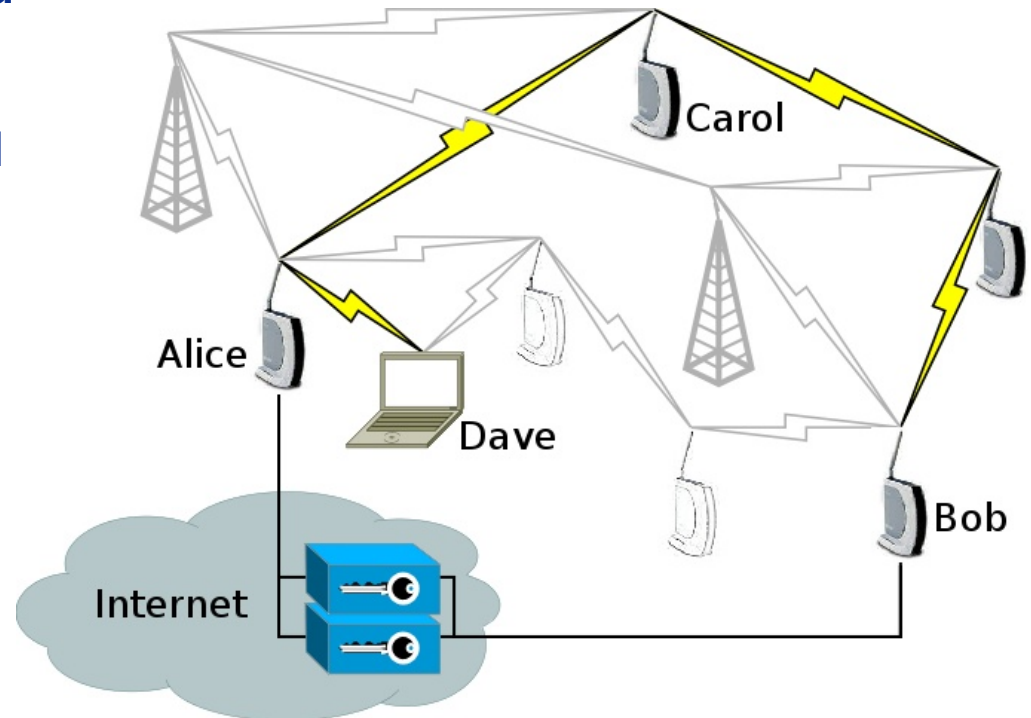
- central Gateways tunneled into our network
- enables 'roaming' over different Internet providers
- **IP of donators is not spread to the outside world**
- availability of services controlled by the community
 - no services blocked



Opennet Problems

- no information about provided bandwidth trough user-DSL available
- no selection of user-provided DSL possible

(OLSR routing loops)



Software Overview

- OpenWrt
- Freifunk Firmware
- Opennet Firmware
- OLSR
- B.A.T.M.A.N.

OpenWrt

Linux distribution for embedded devices

- provides writable filesystem
- package management



OpenWrt
Wireless Freedom

“For developer, OpenWrt is the **framework to build an application without having to build a complete firmware around it**; for users this means the ability for full customization, to use the device in ways never envisioned” (<http://openwrt.org>)

core developers

- Mike Baker <mbm>, Imre Kaloz <Kaloz>, Waldemar Brodkorb <wbx>, Nicolas Thill <Nico>, Felix Fietkau <nbd>, Florian Fainelli <florian>

both **Freifunk** and **Opennet Firmware** are build up on **OpenWrt**

OpenWrt

started as a free (Linux) firmware for Linksys OpenWrt routers in 2004

- 2003 SeattleWireless: Andrew Miklas found WRT54G running Linux
- community pressure: Linksys released the source code (GPL)
- different open firmware projects appear
 - see <http://forum.openwrt.org/viewtopic.php?id=10221>

first 'usable' version came out in 25.6.2005 (White Russian RC1)

current stable release is 'White Russian 0.9' from the 10.1.2007

heavy development on **upcoming OpenWrt 'Kamikaze', usable by now**

- version 7.09 (Sept. 07), kernel 2.6.22.1, see <https://dev.openwrt.org/>

OpenWrt Kamikaze / buildroot-ng

generate target images for Broadcom, Atheros, x86...

no NVRAM variables anymore, config files are stored on usual JFFS2

“... **OpenWrt does not contain any executables or even sources**, it is an automated system for downloading the sources, patching them to work with the given platform and compiling them correctly for that platform. What this means is that just by changing the template, you can change any step in the process.” (<http://wiki.openwrt.org/BuildRoot>)

great step forward for all who like to develop

- **easy to build** for everybody who is able to compile a kernel
- **easy to adapt** for developers and end-users
- delivers an **Image Builder** to create modified firmware images
- delivers an **SDK** to easily build additional software packages

Freifunk Firmware

“The one and only purpose of this firmware: build citywide intranets by meshing with layer-3 OLSR routing” (Readme.txt)

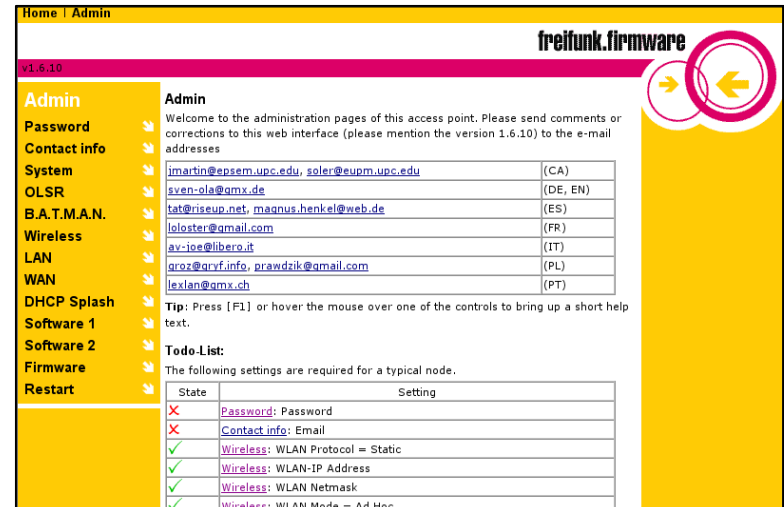
- started in 2004, developed by Sven-Ola Tuecke
- <http://ff-firmware.sourceforge.net/>

current release 1.6.22 (3.12.2007)

- based on OpenWrt White Russian RC6, Linux 2.4.32

“... the **Freifunk Firmware will stay with Linksys/Broadcom**. There are plans to migrate as ipkg on top of OpenWrt/Kamikaze in the future.” (<http://ff-firmware.sourceforge.net/>)

- (Kamikaze based Freifunk version for Fonera is available)



The screenshot shows the administration interface for freifunk.firmware version 1.6.10. The interface is divided into a left sidebar with navigation options and a main content area. The sidebar includes: Admin, Password, Contact info, System, OLSR, B.A.T.M.A.N., Wireless, LAN, WAN, DHCP Splash, Software 1, Software 2, Firmware, and Restart. The main content area is titled 'Admin' and contains a welcome message, a table of user accounts, a tip, and a 'Todo-List' table.

System	Account	Country
System	lmartin@epsem.upc.edu , sofer@eupm.upc.edu	(CA)
System	svn-ola@gmx.de	(DE, EN)
System	tat@riseup.net , magnus.henkel@web.de	(ES)
System	loloster@gmail.com	(FR)
System	av-iae@libero.it	(IT)
System	groz@orvf.info , prawdzik@gmail.com	(PL)
System	lexlan@gmx.ch	(PT)

State	Setting
X	Password: Password
X	Contact info: Email
✓	Wireless: WLAN Protocol = Static
✓	Wireless: WLAN-IP Address
✓	Wireless: WLAN Netmask
✓	Wireless: WLAN Mode = Ad Hoc

Freifunk Firmware

small footprint to support devices with only a small amount of flash

- for instance WAP54G-V2.0 (2 Mb Flash)
 - incredible efficient coding, some ‘magic’
- beautiful **out-of-the box software**
- flash and start you mesh network. no infrastructure needed
- forces not only the development of Freifunk, also of OLSR routing algorithm

good:

- **used by a lot of communities**, many packages and translations available
- easy to build a mesh without additional infrastructure

bad:

- **difficult to adapt**, no ‘cutting-edge’ OpenWrt
- interface seems difficult for newcomers

Opennet Firmware

started in 2005

- **Freifunk was not up to date**
 - on underlying OpenWrt **OpenVPN (openssl) was not running**

started as a 'whole' Firmware based on OpenWrt White Russian RC2

developed by Sebastian Hagen, Mathias Mahnke, Rene Ejury (from Opennet)

migrated into a Package in 2006

- able to run on OpenWrt White Russian 0.9
- compatible to versions from RC5 on



Opennet Firmware

good:

- availability to **extend with up-to-date OpenWrt-packages**
- easy Web-Interface for new users
- OpenVPN Interface
- **OpenVPN encryption**
- Gateway selection
- transmission power adaptation
- remote OLSR configuration
- update notification

bad:

- **to specialized for Opennet Rostock**, no external developers
- inefficient coding
- no translations available
- incompatible with lots of Freifunk developments

Schlüssel-Erstellung

Staat:	<input type="text" value="de"/>
Landesname:	<input type="text" value="Mecklenburg-Vorpommern"/>
Ort:	<input type="text" value="Rostock"/>
eigener Name:	<input type="text" value="**** bitte Namen eintragen ****"/>
Nutzergruppe:	<input type="text" value="OUN"/>
Accesspoint-Name:	<input type="text" value="46.aps.on"/>
email-adresse:	<input type="text" value="ap46@opennet-initiative.de"/>

Geheimen Schlüssel und Anfrage zur Zertifizierung (CSR) generieren:

Mesh-Routing: OLSR



pro-active routing protocol

- **Optimized Link State Routing** (OLSR, RFC3626) protocol
 - developers Andreas Tønnesen, Thomas Lopatic, Sven-Ola Tuecke, Corinna 'Elektra' Aichele, Hannes Gredler, Bernd Petrovitsch, ...
 - **RFC** was never used (?) in wireless practice and **didn't worked for wireless mesh's**. had to be adapted by the community
 - outcome: **Link Quality Extension** (calculates ETX from LQ and NLQ)
 - ETX = expected transmission count
 - fisheye
- ... not RFC conform, but works, most used routing algorithm for community mesh networks, continuously improvements

Mesh-Routing: OLSR Hello (LQ)

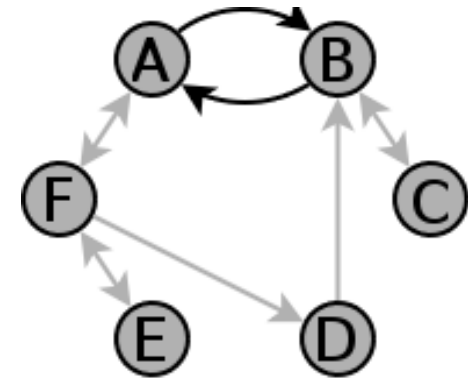
neighbor sensing: (how A and B learn to know of each other)

A -> all: see F (sym, LQ 95%).

B -> all: see C (sym, LQ 85%), D (asym, LQ 60%),
A (asym, LQ 92%).

A -> all: see F (sym, LQ 93%), B (sym, LQ 64%).

B -> all: see C (sym, LQ 72%), D (asym, LQ 63%),
A (sym, LQ 75%).



(D -> all: see F (asym, LQ 30%))

calculation of expected transmission count:

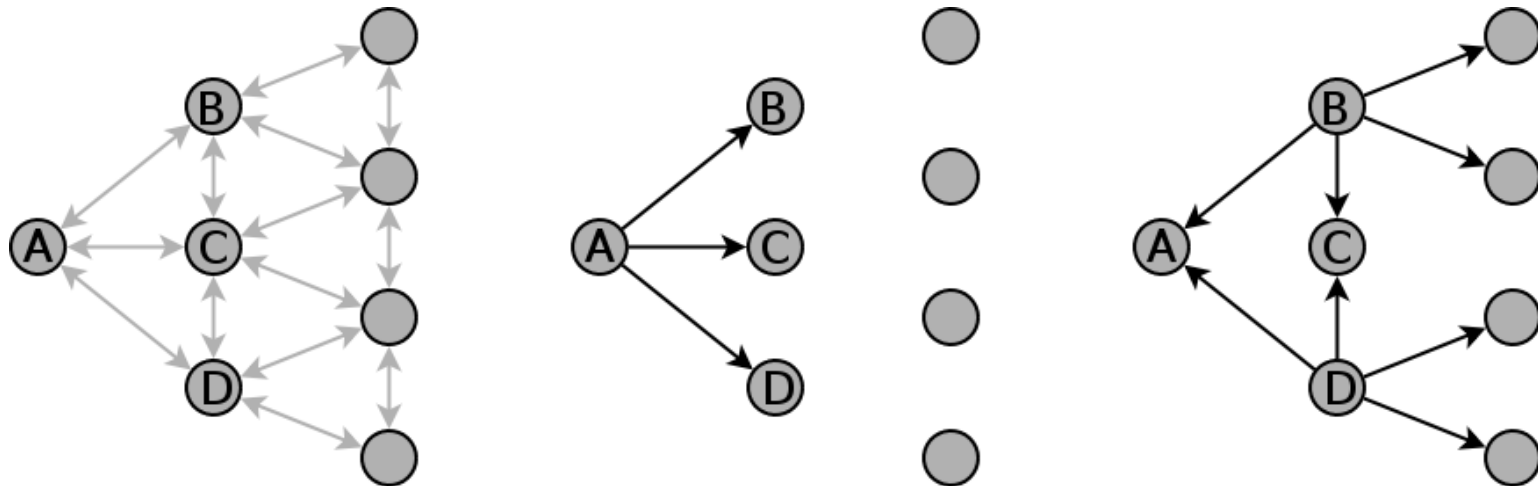
$ETX = 1/(LQ * NLQ)$ (LQ - Link Quality, NLQ - Neighbor Link Quality)

Mesh-Routing: OLSR TC Flooding

topology control (TC) information flooding

- TC-messages contain Information about MPRs (Multi Point Relais)
- only MPRs forwarding (rebroadcasting) TC messages

recalculate routes with shortest path algorithm **if change is detected**



Mesh-Routing: OLSR

every node sends regularly (UDP, Port 698)

- **HELLO** Messages for neighbor-sensing
- **TC** Messages to announce own link-set (**flooded** through the net)
- (HNA Messages, MID Messages)

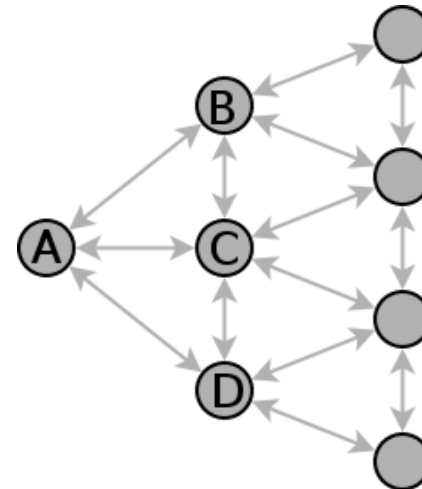
whole topology-graph is calculated, evaluated and routing table is updated

every node 'knows' the whole network

structure and decides based on that where to route a package

- if topology-views differ, routing loops may occur

knowing the whole network seems **worthless if the neighbor node decides where to send the package**

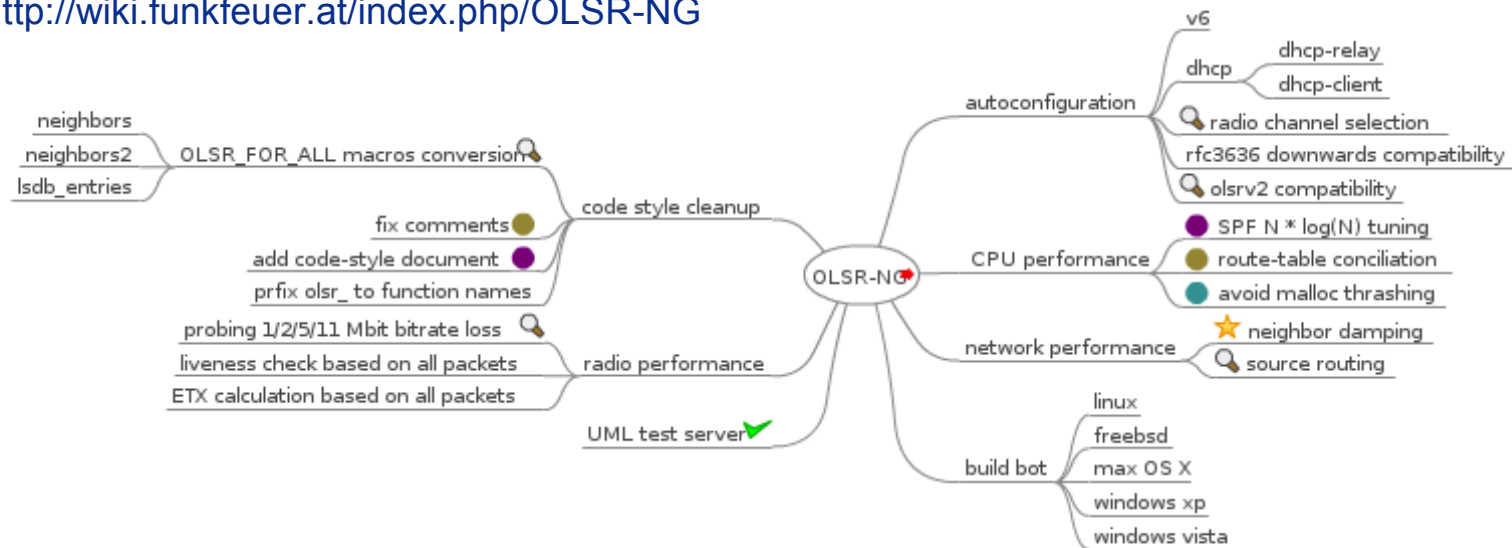


Mesh-Routing: OLSR-NG

goals of OLSR-NG

- scalability: **scale up to 1000+ nodes**
- improve and/or replace the basic algorithms (fisheye/ETX/MPR/...)
- improve the C code for the current olsr.org implementation
- **write a new RFC** from the results

<http://wiki.funkfeuer.at/index.php/OLSR-NG>



Mesh-Routing: B.A.T.M.A.N.

better approach to mobile ad-hoc networking

- <https://www.open-mesh.net/batman>



idea in 2005 by Corinna 'Elektra' Aichele and Thomas Lopatic

- algorithm **tested and developed in the Freifunk community**

developer: Andreas Langer, Axel Neumann, Corinna 'Elektra' Aichele, Ludger Schudde, Marek Lindner, Simon Wunderlich, Stefan Sperling, Wesley Tsai

each node perceives and maintains only the information about the best next hop towards all other nodes

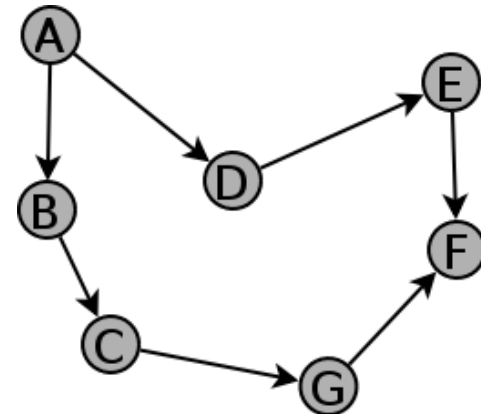
- no route calculation
- no contradicting topology information possible

Mesh-Routing: B.A.T.M.A.N.

OGMs – originator messages

- one message-type for link sensing, neighbor discovery, bidirectional-link validation, and flooding
- **each node re-broadcasts each received OGM at most once** and only those received from the best ranking neighbor, OGMs are flooded
- OGMs on bad routes will get lost or get delayed

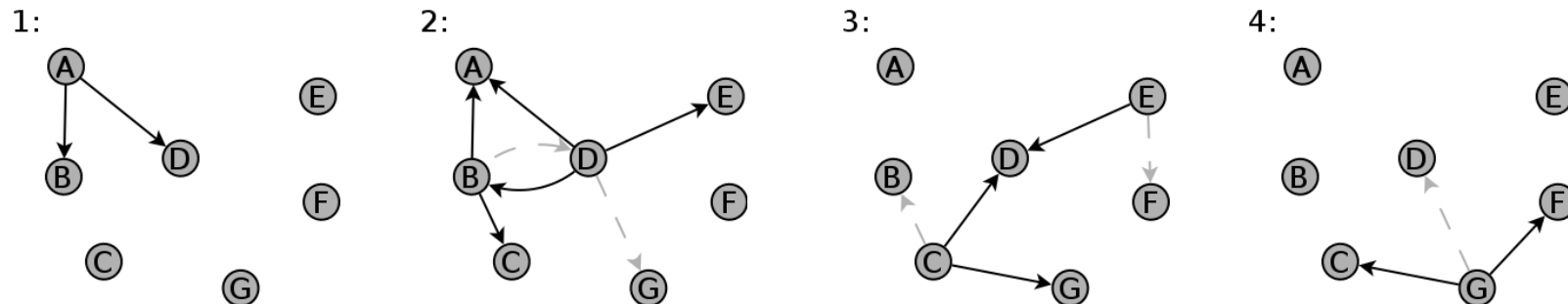
“If **node X** has more than one neighbor, it **can tell by the number of originator messages it receives quicker and more reliable** via one of its single hop neighbors, **which neighbor it has to choose to send data** to the distant node.”



Mesh-Routing: B.A.T.M.A.N.

received **OMGs** only **broadcasted** if

- **first one received** and from current best neighbor (symmetric link)
- A sends OGM(A), B + D receiving
- B is broadcasting OGM(A), A + C receiving, A ignores OGM(A)
D is broadcasting OGM(A), A + B + E receiving, A + B ignoring OGM(A)
- C is broadcasting OGM(A), D + G receiving, D ignores OGM(A)
E is broadcasting OGM(A), D receiving, D ignores OGM(A)
- G is broadcasting OGM(A), C + F receiving, C ignores OGM(A)



Mesh-Routing: B.A.T.M.A.N.

Gateway selection possible, UDP-tunnel will be established

- announcement of Gateway **bandwidth**,
- measurement of Gateway **reliability**

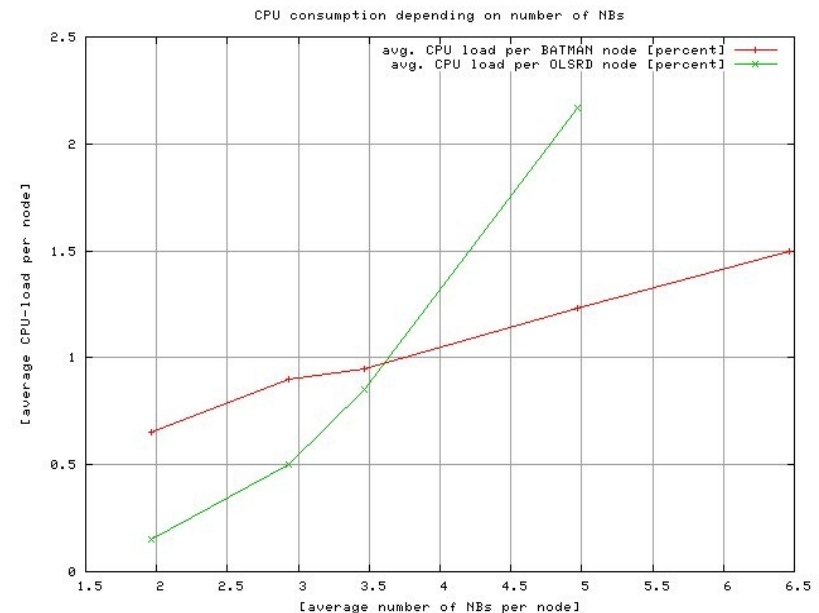
policy routing allows parallel testing of OLSR and B.A.T.M.A.N.

- package and web-frontend available for Freifunk Firmware

real-world **long-term tests missing**

- parallel-to-OLSR tests in Berlin promising
- smaller tests in different communities

if you like to build a mesh,
start using B.A.T.M.A.N.



etc.

other projects are important to enjoy your mesh, some of them...

X-Wrt

- Webinterface for OpenWrt, <http://x-wrt.org/>
- Webif for White Russian, Webif² for Kamikaze

Mapping projects

- a lot of different attempts, in Freifunk Berlin at least four

madwifi

- atheros wireless card driver, now based on OpenHAL

... **translations needed** too, there is yet no Hindi Freifunk Firmware ...

Build your own Mesh

- build your own Firmware
- join the Free Software development

how to start

- **get a OpenWrt-compatible router**
<http://wiki.openwrt.org/TableOfHardware>
 - best by price/quality is Buffalo WHR-G54S (~40€,2000rp)
 - standard one is Linksys WRT54GL (~60€,3000rp)
 - if you can use one with at least 16MB
- **install current Freifunk Firmware**
 - flashing the router might destroy your device, so use reliable power supply



Enjoy your new mesh network!

Motivate other people to join it!



how to continue

install OpenWrt Kamikaze on your router

- it's good to have a stable Freifunk on your router before you start experimenting, else you might have to learn how to de-brick your router (Freifunk enables boot_wait)
- **check** the OpenWrt-forum and the trac on dev.openwrt.org **if your device is supported** (currently problems with broadcom platforms)
- if you like to have a closer look what's going on, **build yourself a serial cable** (http://wiki.openwrt.org/OpenWrtDocs/Customizing/Hardware/Serial_Console)

Broadcom-devices:

- 2.6 kernel uses the bcm43xx driver – not stable (Oct 07)
 - 2.4 kernel used with a proprietary Broadcom module

get familiar with the router configuration,
you have an up-to-date GNU/Linux box

build your own OpenWrt Kamikaze

checkout OpenWrt Kamikaze

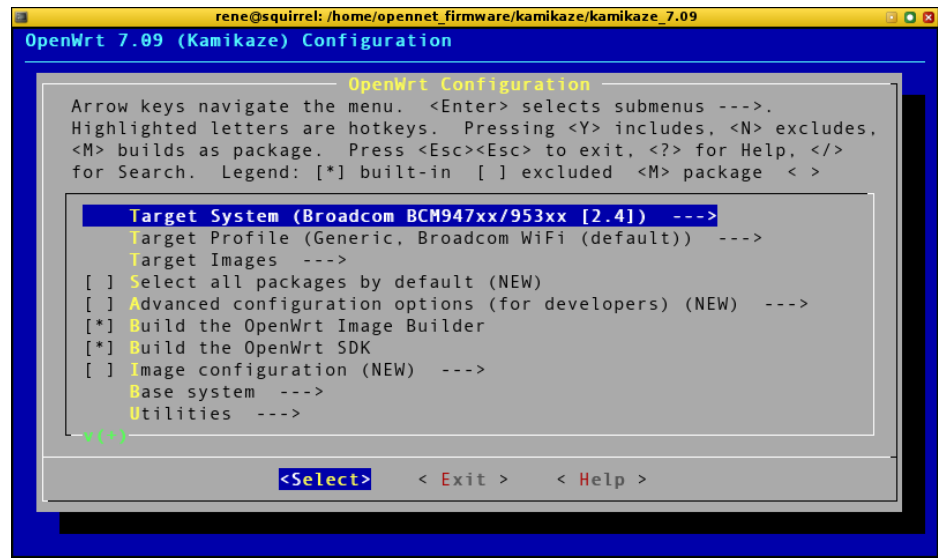
```
#> svn co https://svn.openwrt.org/openwrt/trunk/  
#> cd trunk
```

configure OpenWrt to build the right target (don't change too much for the first run, but **select SDK and Image Builder**)

```
#> make menuconfig
```

download all sources, patch them and **compile buildchain and targets**

```
#> make
```



```
rene@squirrel: /home/opennet_firmware/kamikaze/kamikaze_7.09  
OpenWrt 7.09 (Kamikaze) Configuration  
-----  
OpenWrt Configuration  
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> builds as package. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> package <>  
-----  
Target System (Broadcom BCM947xx/953xx [2.4]) --->  
Target Profile (Generic, Broadcom WiFi (default)) --->  
Target Images --->  
[ ] Select all packages by default (NEW)  
[ ] Advanced configuration options (for developers) (NEW) --->  
[*] Build the OpenWrt Image Builder  
[*] Build the OpenWrt SDK  
[ ] Image configuration (NEW) --->  
Base system --->  
Utilities --->  
-----  
<Select> <Exit> <Help>
```

test your new OpenWrt

target images could be found in `/bin`

select the image you like to flash,
a `*trx` is hardware-independent

flash your router with the new firmware

```

FS: Mounted root (ext2 filesystem) readonly.
freeing unused kernel memory: 132k freed
Warning: unable to open an initial console.
input: ImPS/2 Generic Wheel Mouse as /class/input/input1

Please press Enter to activate this console.

BusyBox v1.4.2 (2007-08-02 21:29:40 CEST) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

-----
      |_  _  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-----
               |  | W I R E L E S S   F R E E D O M
-----
KAMIKAZE (7.07) -----
* 10 oz Vodka      Shake well with ice and strain
* 10 oz Triple sec Mixture into 10 shot glasses.
* 10 oz lime juice Salute!

-----
----- clean this log with 'clean_restart_log' -----
----- restart times: (possibly by watchdog) -----
- (no time retrieved) - system restart ---
root@OpenWrt:/# _

```

scp the image to routers `/tmp`, then ssh into router and reflash:

```
#> mtd -r -e linux write <firmware.trx> linux
```

add some new packages

standalone SDK can do this

- have a look in the docs directory

<http://forum.openwrt.org/viewtopic.php?id=6809>

example: olsrd

- create directory 'olsrd' in packages
- create Makefile there

additional options:

- DESCRIPTION
- MAINTAINER
- DEPENDS

```
include $(TOPDIR)/rules.mk

PKG_NAME:=olsrd
PKG_VERSION:=0.5.4
PKG_RELEASE:=rc1

PKG_BUILD_DIR:=$(BUILD_DIR)/$(PKG_NAME)-$(PKG_VERSION)
PKG_SOURCE:=$(PKG_NAME)-$(PKG_VERSION).tar.bz2
PKG_SOURCE_URL:=http://www.olsr.org/releases/0.5
PKG_MD5SUM:=471d0d268fae388d18f925ea9dfe0150

include $(INCLUDE_DIR)/package.mk

define Package/olsrd
    SECTION:=net
    CATEGORY:=Network
    TITLE:=OLSR (Optimized Link State Routing) daemon
    URL:=http://www.olsr.org/
endef

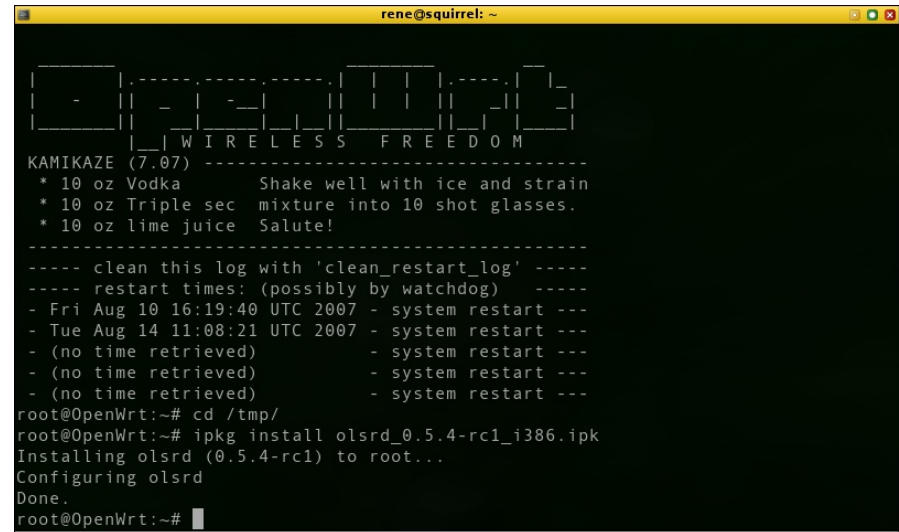
define Package/olsrd/install
    $(INSTALL_DIR) $(1)/usr/sbin
    $(CP) $(PKG_BUILD_DIR)/olsrd $(1)/usr/sbin/
endef

$(eval $(call BuildPackage,olsrd))
```


build your new package

make it.

```
#> find package/olsrd/  
package/olsrd/  
package/olsrd/Makefile  
#> make  
    make[2] package/compile  
    make[3] -C package compile  
    make[4] -C package/olsrd compile  
#> ls bin/packages/  
olsrd_0.5.4-rc1_mipsel.ipk  
#>
```



```
rene@squirrel: ~  
-----  
|_| W I R E L E S S F R E E D O M  
-----  
KAMIKAZE (7.07)  
* 10 oz Vodka      Shake well with ice and strain  
* 10 oz Triple sec mixture into 10 shot glasses.  
* 10 oz lime juice Salute!  
-----  
---- clean this log with 'clean_restart_log' ----  
---- restart times: (possibly by watchdog) ----  
- Fri Aug 10 16:19:40 UTC 2007 - system restart ---  
- Tue Aug 14 11:08:21 UTC 2007 - system restart ---  
- (no time retrieved)          - system restart ---  
- (no time retrieved)          - system restart ---  
- (no time retrieved)          - system restart ---  
root@OpenWrt:~# cd /tmp/  
root@OpenWrt:~# ipkg install olsrd_0.5.4-rc1_i386.ipk  
Installing olsrd (0.5.4-rc1) to root...  
Configuring olsrd  
Done.  
root@OpenWrt:~#
```

Install your new package on the OpenWrt device with

```
#> ipkg install olsrd_0.5.4-rc1_mipsel.ipk
```

add some patches / files

put patches for the retrieved sources in your **package subdirectory patches**

```
#> ls package/olsrd/patches/  
100-olsrd-cvs.patch  
103-olsrd-rt-exportroute-cleanup.patch  
107-olsrd-sven-ola-cfg.patch  
122-olsrd-lqnatthresh.patch  
131-olsrd-policy-metric.patch  
132-save-the-fish.patch  
133-neighbor_routes-fix.patch  
#>
```

if you like to add some files, put them in the **files subdirectory**

```
#> ls package/olsrd/files/  
olsrd.init  olsrd_secure_key  olsr.sh  
#>
```

adapt Makefile to build **multiple packages from one source**, if you like

OpenWrt Image Builder

complete build-environment for OpenWrt – images

- put additionally required packages in packages directory
- add files you like to add/replace
- make your new image

```
#> ls
build_i386 files Makefile rules.mk
staging_dir_mipsel include
packages scripts target

#> make image PACKAGES="olsrd olsrd-
mod-httpinfo olsrd-mod-
nameservice openssl-util fdisk
grub diffutils" FILES=files/
```

```
#> find files
files
files/usr
files/usr/sbin
files/usr/sbin/grub-switch-active-system
files/usr/sbin/grub-install-wrap
files/usr/sbin/sync_spare_partition_config
files/usr/sbin/install_new_system
files/usr/sbin/clean_restart_log
files/etc
files/etc/olsrd.conf
files/etc/init.d
files/etc/init.d/ntpclient
files/etc/init.d/firewall
files/etc/init.d/watchdog
files/etc/init.d/dnsmasq
files/etc/init.d/modules
files/etc/init.d/boot
files/etc/init.d/sshd
files/etc/modules.d
files/etc/modules.d/50-madwifi
files/etc/modules.d/50-scx200-wdt
files/etc/config
files/etc/config/network
...
```

the end, the beginning

join the community network movement

build your own Community Mesh

- it's a free information infrastructure
 - it's a great learning environment and playground, it's fun
 - it creates an interesting community of tech and non-tech people
- join the **mailing list** at <http://bangalore-wifi-mesh.absorb.it>

join the local mesh

join the local Mesh

- (if available :))
- download B.A.T.M.A.N <http://downloads.open-mesh.net/batman/development/>
- or download OLSR <http://olsr.org/index.cgi?action=download>
- grab yourself a unique IP http://b-w-m.absorb.it/index.php/FOSS_Mesh
- switch your card to ad-hoc mode and join ESSID “freifunk.foss.in”

Thanks

to all the developers of **Free Software**, without your work and enthusiasm it wouldn't have been possible to build any Wireless Mesh

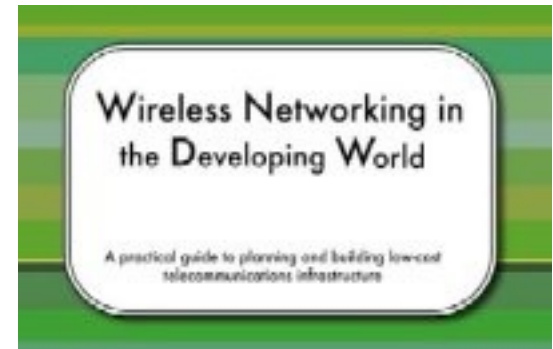
special thanks to all developers of introduced software projects I missed to mention, sorry

some further introduction to really build a mesh:

Flickenger, R., Aichele, C. E., Fonda, C., Forster, J., Howard, I., Krag, T. & Zennaro, M. (2006):

Wireless Networking in the Developing World.

<http://wndw.net/>



Thank you !

René Ejury

email: contact@absorb.it

www: <http://absorb.it>



<http://bangalore-wifi-mesh.absorb.it>

<http://opennet-initiative.de>

<http://freifunk.net>