

Introduction into the Mesh topic

Mesh on PirateBox is currently in testing. I don't recommend large productive use, because

1. It is not known if there are major design flaws, which let it not work on a big scale.
2. Maybe the Mesh-Ad-hoc network will be "encrypted" via WEP, so users cannot accidentally join "the wrong network". This makes current implementation incompatible and creates a **Cell-split**

What is a Mesh-Network



Mesh networking (topology) is a type of networking where each node must not only capture and disseminate its own data, but also serve as a relay for other nodes, that is, it must collaborate to propagate the data in the network. A mesh network can be designed using a flooding technique or a routing technique. When using a routing technique, the message is propagated along a path, by hopping from node to node until the destination is reached. To ensure all its paths' availability, a routing network must allow for continuous connections and reconfiguration around broken or blocked paths, using self-healing algorithms. A mesh network whose nodes are all connected to each other is a fully

connected network. Mesh networks can be seen as one type of ad hoc network. Mobile ad hoc networks (MANET) and mesh networks are therefore closely related, but MANET also have to deal with the problems introduced by the mobility of the nodes.` **Source:**en.wikipedia.org - [Mesh networks](#)

- Read more about that here: [Wireless Mesh Networks - wikipedia](#)
- [How stuff works.com - How wireless mesh networks work](#)

Protocoll

On PirateBoxes in the mesh-package 0.2.x, the protocol **B.A.T.M.A.N. advanced** is used.



- B.A.T.M.A.N. on [wikipedia](#)
- Details on [open-mesh.org](#)

Batman-advanced works on Layer2 which enables every kind of higher protocol (TCP,UDP whatever) on top of it.

What currently works in the PirateBox implementation?

It is currently possible to get a complete Network generated out of one shell script in combination with the protocol above. If you know the remote IP address, you are able to access this box directly- chat and download files. Uploading-Box appears but the upload only happens on the PirateBox you are currently connected to (Name-resolution problem). You are able to use all the stuff of the remote Boxes, too !

The following explanations uses the following terms:

1. local PirateBox \Rightarrow The PirateBox the client is directly associated to, it is the AP or the wired ethernet port it is using
2. remote PirateBox \Rightarrow The PirateBox the client can access across the mesh network. It has no other direct contact.
3. Mesh network \Rightarrow The IPv6 layer of the network.

In the version introduced with LibraryBox 2.0 the complete mesh network implementation was reworked. The mesh network consists of two overlapping segments. The IPv4 address '192.168.1.x' is used "locally" between the local PirateBox and the other clients which are locally associated.

On OpenWrt, the wifi "network" and the ethernet ports are virtually connected. This technique is called "bridge". In the PirateBox implementation, the mesh-network-interface is **directly** connected to that bridge as well. During setup of the different parts of the mesh stuff, we establish an IPv4 packet filter, that between the mesh-interface and the bridge **no** IPv4 packets are able to pass.

But how is it possible to interact between other *remote PirateBoxen*? For that, at the complete mesh network an IPv6 network is setup with a prefix, that is available for every client. So each box has it's own unique IPv6 address, like the client. So it is possible to connect across the mesh network to remote PirateBoxes **and** other clients (maybe serving files). Every Client is served with IPv6 address via a radvd daemon.

The Prefix for the IPv6 addressing is: `fdcd::f:fea` with the following mask `/48`

This setup allows, to use a single OpenWrt router being a mesh node only, relaying traffic from i.e. an RaspberryPi connected via ethernet and providing more powerful services.

The discovery of LibraryBox and PirateBox is working with the use of the [avahi/](#) [bonjour](#) protocol. If you have an avahi daemon and browser enabled, you can browse the list of the available devices.

The mesh network itself as a wifi-interface is only working on OpenWrt currently! After we released it on OpenWrt, the scripts will be modified to work on other environment, but as we can use any OpenWrt-able device as a mesh-entry-node - it is not our priority topic.

What currently does not work?

Because of the *cheap* price one idea is, to use two MR3020 instead of only one. One is responsible for normal clients and the other one only for Mesh on a different channel (so no collisions anymore)- that is not so easy in the current implementation 😊

Name resolution. Even if you know the unique name of a remote Box, you cannot simply type it into your browser. There is the need to finish and test a script, that maps into normal DNS resolution. Also we need a list on the web frontend, that you can simply navigate to other boxes.


Due to the current state of development, the upload only works with the *local PirateBox*. If you visit the website of a *remote PirateBox* the upload-Box on the right only points to the upload button from the *local* one. The problem at this point is the name resolution. (The easy fix is to change the hostname used on the index.html file, but that needs some testing before making use of that).

Clients, which are not able to use IPv6, won't have mesh access.

Flaws of the current mesh implementation - PirateBox

- [ars Technica - Multi-hop matters: the state of wireless mesh networking](#)
- The most low-budget router only have one wireless card, which results in the following Problem
 - Only one “router in Range is allowed -to speak-”, everthing else causes collisions and drops the rate
 - It is possible to run AP & mesh-ad-hoc network on one chip on the same channel, which reduces the transfer rate alot and causes collisions.
 - So it is slow - **But it works** .
- In general -multi-hop- networks with using one wireless card & channel results in a drop of bandwith during every “Hop” (jump over the next node).
- Using one big network for thousands of devices can bring the network in a not-usable state because of broadcast storms.

Forban

 **Fix Me!** Forban is currently discontinued for PirateBox. This is not because I don't like it, it is because we had a major structure reorganisation that made the Forban package incompatible with the current PirateBox 1.0 implementation. Even the default shipped configuration doesn't work with the current IPv6 (better) implementation of PirateBox.

 **Fix Me!** I leave the part inside this page to explain Forban in our context a bit.

Most people reading or thinking about Mesh-network, that it is something self-synchronizing. Currently this feature is implemented with the use of [Forban](#) (but in fact, that is not the Mesh-itself (see above).



Forban is a relative “heavy” Software for the small Boxes like MR3020. The software comes preconfigured in “Shared-Mode” only. Forban sends regular informations, that there is a Forban active via Broadcast across the network. In combination with the implemented Mesh-network, the result is, that if Boxes reaches their connectivity area, the Forban detects the remote Forban and lists it in his webinterface. **Share-Mode** only offers data and the user can download it directly from the device to his client.

The “**opportunistic**”-Mode is correctly set up, but not activated. This mode reacts on incoming Broadcasts and tries do download every file from the remote Forban. This works on the following rules:

- Search for filter (default: *everything*)
- Exclude from search filter (default: *nothing*)
- Download until max freespace of x GB is reached (default: *0GB*)
- Download a maximum filesize x MB ((default: *no limit*)
- Download location (default: *directly into Share-folder*)

The current flaws of the implementation (0.0.34) are:

- Only downloads File if the remote file is bigger than the current one (not taking account of Hash-values or unfinished downloads)

- No download resume (because it would break files) - downloads a file completely at once.
- **OpenWRT** Larger files (>100MB) over Mesh-Network seems to cause trouble (unfinished, alot of CPU/IO load). This causes sometimes to unfinished downloads never finished.
- Download this file only from one Box (if this box vanishes and the file is somewhere else (on the same path) , it will be downloaded from there)
- Forban is a “closed System” with not the possibility of getting more information about the (PirateBox)features of the other boxes
- ****Time saving problems**** of the cheap **OpenWRT** routers makes it impossible to detect *real* newer files.

Sidenote: I don't want to blame Forban in a hard way. Adulau did a good job and he already proposed a list of possible improvements to circumvent a lot of the problems I listed above. You can find this list [on his github project ; Forban2.0](#).

Other Forban alternatives

All option would be an “opt-in”, like the mesh implementation.

All alternatives should have a resource limiting factor on the server side to prevent to much load and a resulting bad user experience on the Box downloaded from.

We are discussing lightweight options for synchronization such as:

the FTP way

LibraryBox brings currently an FTP synchronization. This FTP sync is used for a 1:n synchronization, where the node knows his specific main-source for downloading. The FTP implementation brings a bare skeleton for anonymous access. This in combination with a clever avahi lookup can be used for an automatic download script.

the rsync way

The more open option is to use an rsync daemon and client (combined with an avahi lookup script) to download from remote content.

Why is not in PirateBox 1.0 and where does it stuck?

The current problem is time and I don't want to make promises I can't fulfill. I tried with the lines above to describe how complex an implementation like this is and I showed some of the pitfalls.

Currently, I'm trying to nail down the mesh stuff for the 1.1 release of PirateBox and to bring a development snapshot on its way. In the current *environment* it might almost impossible to reproduce my work by other, which means no one can effectively help developing.

On github, the guy called “stylesuxx” helped me alot with automating the tasks for creating snapshots. I'm currently removing the rough edges and after that I hope to get more input and work from the community (because they see & test the latest state more easier).

Future plans

1. mDNS to sDNS resolution via hosts file (50% done)
2. display known PirateBox on webpage (0% done)
3. Global Chatbox (50% done)
4. Create ideas about gamification with each node.
5. Closer look at <http://www.open-mesh.org/projects/open-mesh/wiki/Alfred>