TP-Link WR802N; a cheap and power efficient RIPE Atlas probe alternative.

I didn't want RIPE to send me extra Atlas probes for experimenting since I already own three (one V4 and two V5 versions on three different ISP's). I felt that requesting a fourth one is a bit too much to ask for.

So I experimented with alternatives; a VMware CentOS image on a Windows 10 PC, an Ubuntu 18.04 mini PC running a self compiled Atlas probe and a Raspberry CM4 based OpenWRT router. All worked fine but consume relatively speaking a lot of energy just for running this small task.

So I had a look around and found what I believe is the cheapest, smallest and (very) energy efficient OpenWRT travel router; the TP-Link WR802N, available for instance on Amazon and eBay for just over USD 20,-

Product specifications are here: https://www.tp-link.com/us/home-networking/wifi-router/tl-wr802n/

This remarkable little box has less volume than the RIPE v5 probe and consumes half its energy, I measured only 0,7 Watt when idle. I tested its performance against a V5 probe and they where equal.

The WR802N has a powerful enough MIPS processor, 32 Mb. of RAM and 8 Mb. flash memory, a 100Mbps ethernet port and a micro USB port for power. It comes with a nice white power cord, a short ethernet cable and a small 5V, 1A AC adapter. It runs fine with the supplied cable from any USB type A port.

Note: if you buy second hand or old stock, make sure you get version 4 of the device (latest version), earlier versions only had 4 Mb. flash memory and can't be made to run a recent version of OpenWRT, required for the Atlas software.

A more advanced version (the WR902AC) is also available, which adds 802.11ac wireless instead of 802.11n and an USB port, I will do a write-up for this device when time permits.

The WR802N comes with “proprietary” TP-Link firmware (based on OpenWRT) but it is also fully supported by the latest generic version of OpenWRT (at this moment version 22.03) and a perfectly working version of the RIPE Atlas software is also available for OpenWRT (with a lot of thanks to the excellent OpenWRT community!).

Installing OpenWRT, the RIPE Atlas software and its dependencies is straightforward with one caveat; the limited amount of flash storage on the device of which just around 2 Mb. is available after boot.

OpenWRT implements a compressed overlay filesystem whereby the firmware from flash is loaded in RAM after boot and the overlay filesystem overlays the firmware’s filesystem in RAM with the mentioned 2 Mb. flash storage. Thus any change to the filesystem’s /etc, /root, /usr and /bin directories is actually written to flash and retained after a reboot.

The problem is that the RIPE Atlas OpenWRT software and its dependencies require some 9 Mb. in total, way too much for the available 2 Mb. overlay in flash. But there is more than enough RAM available, which can be used to store all the files. With the caveat that they are lost after a boot, for which we need to find a solution; stay tuned.

The following is required to turn the WR802N into an Atlas probe:

- a functional LAN with a DHCP server in it (I used DHCP from my internet router);
- for installation a PC running a TFTP server (I used SolarWinds’ free TFTP server, see: https://www.solarwinds.com/free-tools/free-tftp-server);
- a PC with SCP (I used WinSCP, see: https://winscp.net/eng/download.php);
- an SSH terminal program (I used PuTTY, see: https://www.putty.org/);
- an FTP or webserver for storage of the required files (see below).

The following is not too detailed, if you run into problems, email me (see my email address on the last page).

Do the following to make it work:

1.
Download the OpenWRT 22.03 image for the device from:

https://downloads.openwrt.org/releases/22.03.2/targets/ramips/mt76x8/openwrt-22.03.2-ramips-mt76x8-tplink_tl-wr802n-v4-squashfs-tftp-recovery.bin

and rename it to **tp_recovery.bin**.

2.
Give your PC a fixed IP address of **192.168.0.66** and connect its ethernet port directly to the WR802N.

3.
Put the **tp_recovery.bin** file in the default TFTP server directory used by your TFTP server install.

4.
Install the OpenWRT firmware as explained in: https://openwrt.org/toh/tp-link/tl-wr802n_v4

5.
After a successful flash after a couple of minutes the green LED on the WR802N will light continuously.

Congratulations!

After a reboot the device runs OpenWRT 22.03. But it acts as a DHCP server with a fixed IP of 192.168.1.1 which we don't want, we need it to be a DHCP client with a non-fixed IP.

So you need to restart the WR802N, **still with your PC connected** to it, but before the restart configure the PC to get an IP automatically (which is probably your default setup for this PC).

After a restart with a browser on the connected PC go to 192.168.1.1 and you will be greeted by the OpenWRT GUI (called LuCI).

From the menu "Network" select "Interfaces" and press "Edit" for the LAN interface (there is only one interface). Select "DCHP client" for the protocol to use and press "Save". LuCI will prompt you whether you are sure since after the reboot the device will not be available anymore on 192.168.1.1.

After a reboot you can connect the WR802N to your LAN and it will be given an IP address within your LAN. It is always problematic to find which IP address, I use Advanced Port Scanner (see: https://www.advanced-port-scanner.com/) and after the scan a subsequent "ARP –a" command from the command line will show all MAC and IP addresses in the segment scanned. Look for a MAC address starting with **9C:A2:F4**, which is registered to TP-Link.

You now know the device's IP address and can go to that IP address in a web browser, again you will be greeted with the OpenWRT GUI LuCI.

From menu "System" select "Software" and under "Actions" press "Update lists". If all is well the package manager (OPKG) will retrieve all package lists from the OpenWRT servers.

If you now filter the "Available" packages for "Atlas" the following will be shown:

atlas-probe	**2.4.1-2**	**206.8 KB**	**RIPE Atlas is a global, open, distributed Internet measurement platform,…**
atlas-sw-probe	**5040-1**	**15.4 KB**	**RIPE Atlas SW probe is software variant of RIPE Atlas Probe.…**
atlas-sw-probe-rpc	**5040-1**	**1.6 KB**	**Provides ubus calls for probe.**

Looks fine, apparently that will fit in the mentioned 2Mb. flash storage. However, if you select any of these packages and press "Install" **(without yet continuing with the install !!!)** it will show that the following dependencies also will need to be installed:

bind-dig	**rpcd**	**librt**	**libopenssl1.1**
openssh-client	**sudo**	**dropbearconvert**	

And especially bind-dig also installs a number of dependencies. Installing all of these in one go (done by OPKG when you would continue with the install) would result in an error; the device will run out of storage space in flash.

So what is the solution? To install these dependencies separately **one after the other** and after every install log into the device, move the largest files (mostly libraries) to some directory within the file-system in RAM (command: mv <source> <destination>), preferably a subdirectory in /tmp, say **/tmp/files**, replacing the original file with a link to the file in its new location (command: ln –s <target> <source>).

If this is done properly (say you used "mkdir /tmp/files") then that directory will hold (until a reboot) the following files (sudo and misc are directories):

libbind9-9.18.7.so	**scp-openssh**	**libcrypto.so.1.1**
libdns-9.18.7.so	**ssh-openssh**	**libssl.so.1.1**
sudo	**libisc-9.18.7.so**	**sudoers.so**
libsudo_util.so.0.0.0	**libisccfg-9.18.7.so**	**misc**
visudo		

Make sure you use an SCP program (I used WinSCP) to copy those files to your PC.

The Linux command to create the soft links is **ln**:

ln –s <target> <source>

for instance:

ln –s /tmp/files/libssl.so.1.1 /usr/lib/libssl.so.1.1

Make a note of where files have been installed to use as the <source> location. Since these are always installed in the /overlay/upper directory (before you move them) I created a listing of files in that directory (command: ls –al –R /overlay/upper) before and after install of one of the components to find out where files were installed.

Apart from configuration files and such (small files in /etc for instance, don't bother with these, they are too small to gain space in /overlay) most of the files after an install went to **/usr/lib** and **/usr/bin**, the following will help (again using /tmp/files as an example):

/usr/lib :

libbind9-9.18.7.so	-> /tmp/files/libbind9-9.18.7.so
libcrypto.so.1.1	-> /tmp/files/libcrypto.so.1.1
libdns-9.18.7.so	-> /tmp/files/libdns-9.18.7.so
libisc-9.18.7.so	-> /tmp/files/libisc-9.18.7.so
libisccfg-9.18.7.so	-> /tmp/files/libisccfg-9.18.7.so
libns-9.18.7.so	-> /tmp/files/libns-9.18.7.so
libssl.so.1.1	-> /tmp/files/libssl.so.1.1

/usr/bin :

scp	-> /tmp/files/scp-openssh
scp-openssh	-> /tmp/files/scp-openssh
ssh	-> /tmp/files/ssh-openssh
ssh-openssh	-> /tmp/files/ssh-openssh
sudo	-> /tmp/files/sudo

After all this, **install curl and install dropbearconvert** from the LuCI GUI (will be installed in flash).

Again: make sure to copy all the mentioned files above to your PC since they are lost after a reboot! If you don't save them you will have to start from the beginning!

If all is well, you have enough storage left in flash to install the three Atlas components from the LuCI GUI. One of the installs results in LuCI OPKG showing you what commands to use to generate your public key, obtaining it and how to register your probe with it.

But… as I said, after a reboot all files mentioned above are gone, so we need a method to copy them over after a reboot. All soft links will remain since these are kept in the /overlay filesystem in flash. We don't need to recreate those after a reboot.

There are of course various ways to copy the files over after a reboot (or power outage).

Make sure that when the executable files come from a non-Linux environment they are made executable once copied over (command: "chmod 700 <file>") for the files in /tmp/files that have a soft link in /usr/bin.

After a reboot OpenWRT will try to start the Atlas software which will fail since the required libraries are lost from /tmp/files (or any other name you used). Thus before copying the files over you need to login to the device and do a "/etc/init.d/atlas stop".

Initially I used WinSCP to copy the files over but if you have a FTP-, TFTP- or web-server running in your network you can fully automate this.

I have in my LAN an Apple Time Capsule (running NetBSD and rsync and which is accessible with SSH, time for another write-up) running a webserver. So I copied all files in a subdirectory on that webserver.

The program copying them over from the webserver to the probe I called "restart.sh", it uses CURL to copy them from the webserver.

If you add "restart.sh" to /etc/rc.local then OpenWRT will run it after a reboot.

My **/etc/rc.local** looks like this:

```
/bin/sh /root/restart.sh
exit 0
```

do not forget to make restart.sh executable: "chmod 755 restart.sh".

For editing local files on the probe, "vi" is available, but if you want a small excellent editor install nano from the LuCI interface, then you will need to save "libncursesw.so.6.3" also. Also from the LuCI GUI you can go to "System", "Startup", "Local Startup" and edit rc.local from there.

My **restart.sh** looks like this:

```
#!/bin/sh

/etc/init.d/atlas stop                  # stop the atlas script
while ! ping -c 1 [IP-of-webserver]     # wait until the device has an IP
    do
    sleep 1
done
```

(continued on next page)

```
cd /tmp
curl -s -O http://[IP-of-webserver]/files.tgz      # use curl to get the files to /tmp
cd /

tar x -zf /tmp/files.tgz                           # untar and uncompress to /tmp/files
rm /tmp/files.tgz                                  # remove the archive

/etc/init.d/atlas start                            # start the atlas script
exit 0
```

the while-done loop is required to make sure the LAN interface is up and the WR802N has been supplied with an IP address before copying the files.tgz file.

On the WR802N I created the “files.tgz” file using tar:

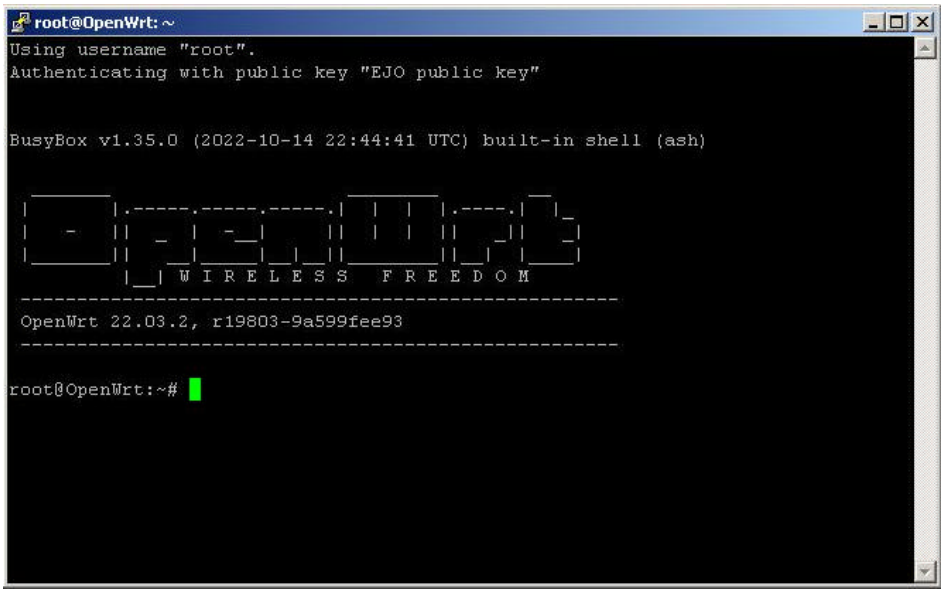
```
cd /tmp
tar c -zf files.tgz /tmp/files/
```

Make sure the executable files (scp-openssh, scp-openssl and sudo) are executable as explained above before the archive is created. After that I copied files.tgz from the /tmp directory over to the webserver using WinSCP and I deleted it from the device (command: “rm /tmp/files.tgz” but a reboot will also delete the file).

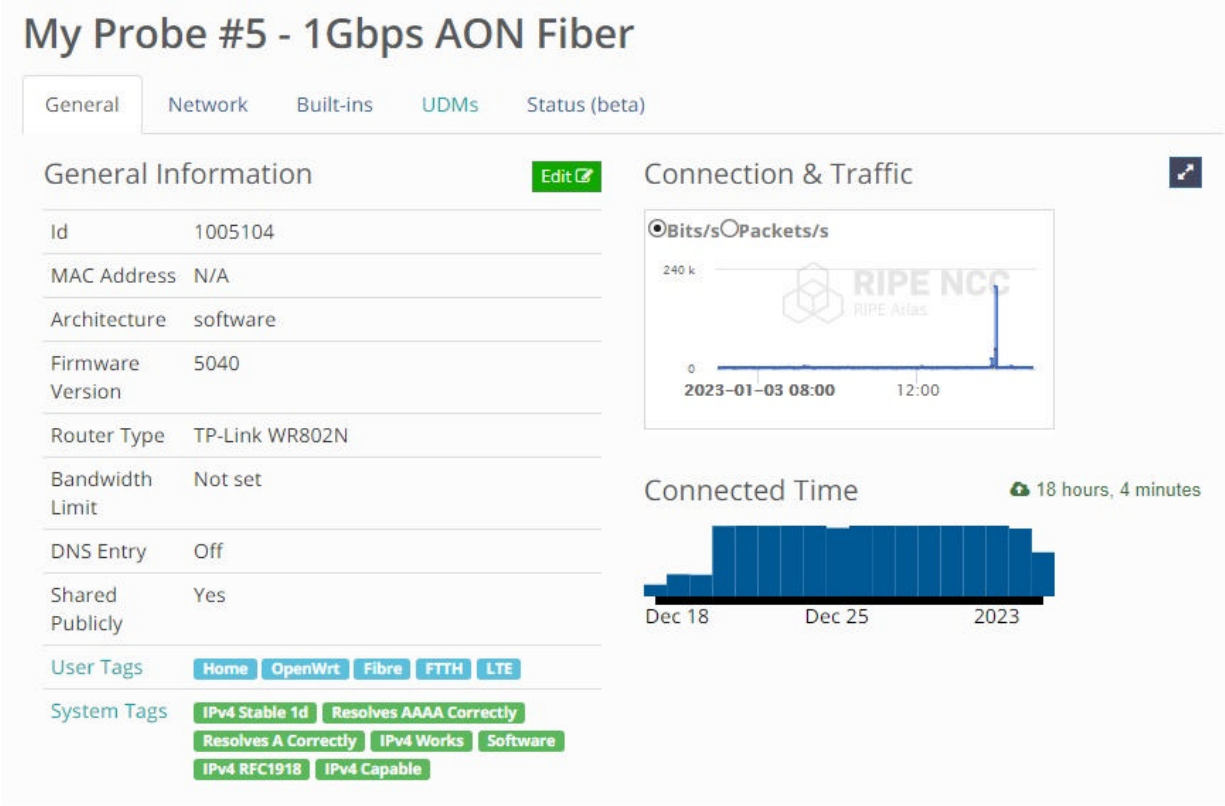
Et voila! ***A fully functioning Atlas probe consuming almost no energy.***

For convenience I also installed htop and midnight commander (MC) on the device. MC requires glib2 and two other libraries that need to be installed as above, flash storage is also too small for these.

I also generated a public/private key pair to be used by PuTTY so that it logs in automatically:



The result:



Some tips re. security

Since the probe will be unobtrusively running 24/7 there might be security concerns.

To harden the device it is advised to only allow access via a private/public key mechanism (disabling password access) and disable the LuCI GUI (which can always be restarted when needed via SSH).

See the dropbear man page (for instance at: https://linux.die.net/man/8/dropbear) on how to generate a keypair, put the private key in /root/.ssh/authorized_keys and do a chmod 600 /root/.ssh/authorized_keys.

Then change /etc/config/dropbear to:

```
    config dropbear
        option PasswordAuth         'off'
        option RootPasswordAuth     'off'
        option Port                 '22'
#       option BannerFile           '/etc/banner'
```

Make sure you test first whether login with your public key works before disabling password access since you might otherwise lock yourself out

In rc.local or in restart.sh add “/etc/init.d/uhttpd disable” and “/etc/init.d/uhttpd stop” to disable uhttpd.

Whenever LuCI is required again log in safely using SSH and do a “/etc/init.d/uhttpd start”.
Disabling utthpd also frees valuable memory.

Happy hacking!

Questions? Don’t hesitate to contact me:

ernstoud@euronet.nl or ernstoud@gmail.com

Version 2 - January 2023

Ernst J. Oud