FreeBSD 10.x on Ubiquiti EdgeRouter Lite

- **Update 2014/11/19:** New, untested images available for 10.1-RELEASE
- **Update 2014/09/21:** New version of the build script creates more inodes in root filesystem.

The Ubiquiti EdgeRouter Lite is a neat little device that costs less than US$100, has three Ethernet ports, and can run FreeBSD/mips. It's based on the Cavium Octeon CN5020 platform and features a dual core 500mhz MIPS64 processor, 512MB RAM, and 4GB storage on removable USB.

The EdgeRouter Lite in the foreground, near a Netgear WNDR3700 and a bulky ISP-provided cablemodem.

This page provides ready-to-use images of FreeBSD 10.x-RELEASE. Thanks to the open nature of the EdgeRouter Lite, it's very easy to install and use these images; just follow the instructions below. Thanks to the fine folks at the FreeBSD Project, building your own is almost as easy. A script to build them, along with instructions, is also provided. Special thanks is due to Juli Mallett and Warner Losh, without whose hard work and generous assistance none of this would be possible.

Note that this is experimental software which comes with no warranty of any kind. These builds are works in progress and are not fit or suitable for any purpose whatsoever. By proceeding you assume all risks.

On my EdgeRouter Lite, the builds provided below are stable and pretty much fully functional. There are two outstanding issues:

1. Performance could be a little better, though it's more than adequate for my home Internet connection. Basic packet passing between two Gigabit hosts seems to top out at about 250Mbits/sec.
2. There is currently no way to pass boot options (such as single-user mode) to the kernel from U-Boot.

Hardware crypto acceleration via `/dev/crypto` seems to work. Use AES in CBC mode to see a huge speedup over CTR.

Image Files

2014/11/19	FreeBSD 10.1-RELEASE-r274683 *untested*	SHA512
2014/01/24	FreeBSD 10.0-RELEASE r261039	SHA512

You can also easily build your own image from the FreeBSD sources using the provided `mkerlimage` script, then return here to continue with installation.

Getting Started

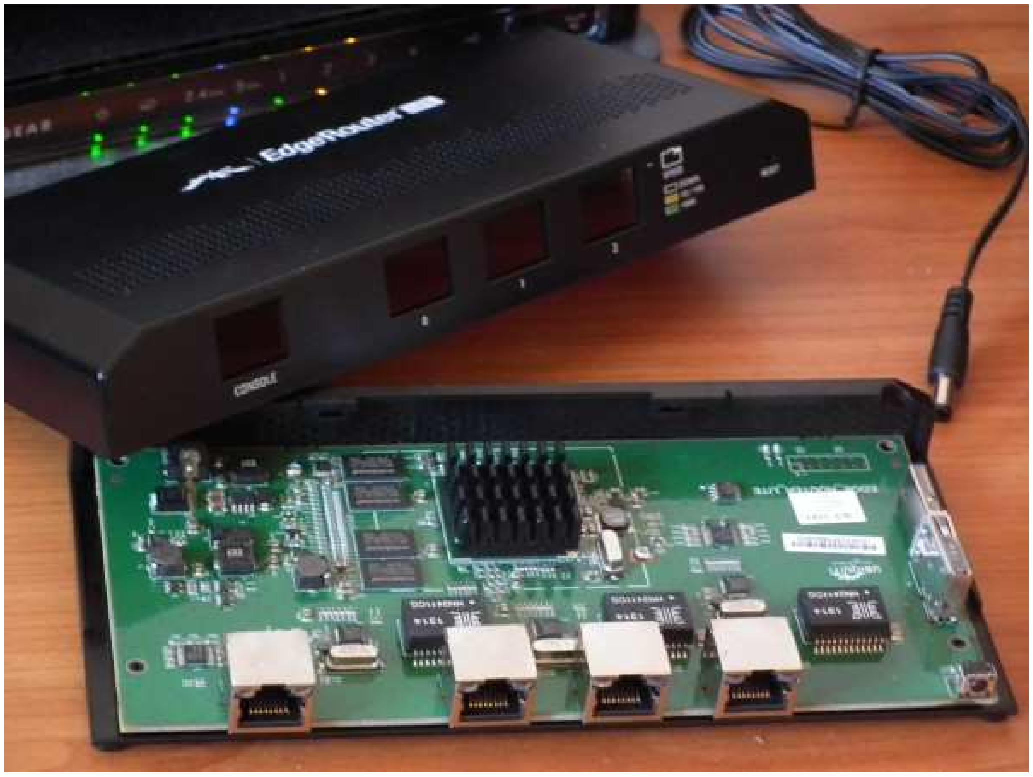
All you need are the following:

1. a computer with a USB port
2. the device
3. an RJ-45 serial console cable
4. unless your computer already has a 9-pin serial port, a USB-to-DB9 adapter

   and last but certainly not least,

5. a Phillips-head screwdriver.

The EdgeRouter's Lite on-board storage is a removable USB drive. Installing the image is just a matter of opening the device, taking the drive out, writing the image to it, and putting it back in.



The EdgeRouter Lite with the cover removed. Inside is just a board. The removable USB storage is on the right.

Configuring the bootloader afterwards only takes a minute, and then you're done. If you haven't already set up a serial console for your device, though, you'll need to do that first.

Serial Console

You may wish to skip to the next section if you already have a working connection to the serial console of your EdgeRouter Lite.

The serial console allows you to interact with the device's firmware before boot, which you will need to do to configure it to boot FreeBSD. Since you'll need it anyway, the provided images don't enable the ssh server by default, and you'll log into your new FreeBSD system for the first time this way, as well.

First, connect the RJ-45 console cable to the port marked "Console" on the device. Connect the other end to a 9-pin serial port on your computer if you have one; otherwise connect it to a USB-to-DB9 adapter, and connect that to a USB port on your computer.

Next, you will need any serial communication software. On FreeBSD and Linux systems, you can install `minicom` by entering one of the following commands:

System	Command
FreeBSD 10.x	`pkg install minicom`
FreeBSD 9.x	`pkg_add -r minicom`
Debian, Ubuntu, Mint, etc.	`apt-get install minicom`
Fedora, CentOS, RHEL, etc.	`yum install minicom`

On other Unix-like systems, you can try building minicom from source. On MacOS you can do that or try to install it from Fink. On Windows you're on your own; try RealTerm.

Determine the name of the serial port you're using. On FreeBSD systems, USB serial adapters will have names like `/dev/ttyU0` and hard-wired serial ports, `/dev/ttyu0` with a lowercase 'u.' On Linux systems, the equivalent files are `/dev/ttyUSB0` and `/dev/ttyS0`, respectively. If you have more than one, the number might be greater than '0.' On Windows, it's usually called `COM1`, otherwise `COM2`, `COM3`, and so on. On MacOS, it might be `/dev/cu.usbserial`; look for other `/dev/cu.*` files if not.

Configure the software to use the port, with settings 115200 8N1 and no flow control. Those are usually close to the defaults, but it's important to get them right as accidentally leaving flow control enabled or another error could leave you with a partially or not at all working connection. To configure minicom, enter the command `minicom -s` as root, then navigate to the "Serial Port Setup" menu.

Start the software and plug power into the device. You should see the EdgeOS Linux kernel boot and ultimately present you with a login prompt. If your device was already powered on, just hit the Enter key once or twice to get a new login prompt. Type something at the login prompt to verify that your connection is working properly. For example, you can log into EdgeOS (the default user and password are both `ubnt`) and run the `halt` command to shut it down and prepare for installing FreeBSD.

Installation

Unplug the device completely and open the plastic case by removing all the screws. Notice the lone USB port containing a removable drive on one end of the board. You can use your fingers to take the drive out, but it's a little slippery, so you might need to use your shirt for grip. It turns out there's a benefit to wearing shirts around the house, after all.

Plug the drive into your computer. For FreeBSD, Linux and other Unix-like systems, the command to write the image to the drive is below. You'll need to download the `xz` software for MacOS or if it's otherwise not installed. On Windows, which doesn't have `dd`, you're on your own; try the Ubuntu Image Writer (you'll also have to download `xz`).

Determine the device name assigned by your system. On FreeBSD, it will be something like `/dev/da3`, where the '3' will be the first available number after all your other SCSI and USB drives. On Linux, it will be something like `/dev/sdf`, where the 'f' will be the first available letter after all your other ATA, SCSI and USB drives. On MacOS, it will be something like `/dev/rdisk1`.

Ensure it wasn't mounted automatically, by substituting your device name in the following commands and running them as root:

```
mount | grep '^/dev/devicename' | awk '{print $1}' | xargs -n1 umount
mount | grep '^/dev/devicename'
```

If these commands produce no output, and you're certain you have the device name correct, proceed with the following command as root, substituting your device name and the filename of the image you downloaded or built. **Warning: a simple typo in the device name can wipe out your system's boot partition. Check carefully before hitting the enter key.**

```
xzcat freebsd-ubiquiti-edgerouter-lite.debug.image.xz | dd of=/dev/devicename bs=1024k
```

This will take a few minutes. When finished, remove the drive and put it back into the EdgeRouter Lite. Reconnect the console cable, ensure your terminal software is running, and plug power into the device.

Configuring U-Boot

The first time you power up the device with the new image, the bootloader won't know what to do and will drop you to a prompt. Enter these commands to save the command sequence needed to boot FreeBSD to a variable (`boot_freebsd` in this example):

```
setenv boot_freebsd 'fatload usb 0 $loadaddr kernel/kernel ; bootoctlinux $loadaddr coremask=0x3
saveenv
```

You can now boot to FreeBSD simply by entering this command:

```
run boot_freebsd
```

Optionally, enter the following command to skip the pre-boot countdown. Personally, using minicom, I am still able to access the U-Boot console by sending a `BREAK` at the first signs of life on the serial port, so I prefer not to have a delay. If you don't want to **risk locking yourself out** of the U-Boot console, skip this command or replace the `0` with something like `3` (it's in seconds):

```
setenv bootdelay 0
```

Finally, to boot directly to FreeBSD by default, enter these commands:

```
setenv bootcmd_orig $(bootcmd)
setenv bootcmd 'run boot_freebsd'
saveenv
reset
```

Growing the Filesystem

At this point, you should be looking at a fully booted multi-user FreeBSD system. One of the first things you'll want to do is expand the filesystem to use all the free space left over on the drive after you wrote the relatively small image to it.

Log in as root, then enter this command to bring the system to single-user mode:

```
init 1
```

Press `ENTER` to accept the default shell, then enter the following commands:

```
df -h /
mount -o ro /
```

```
gpart resize -i 2 da0
gpart resize -i 1 da0s2
growfs /dev/da0s2a
df -h /
```

You should be able to see much more available space in the second `df` invocation. Enter the `reboot` command to reboot back into multi-user mode.

Success!

Change the root password, edit the `/etc/rc.conf` file to set up networking and services, and refer to The FreeBSD Handbook for system administration basics.

Building Your Own

> *Update 2014/09/21:* the latest version here fixes the problem of too few inodes, first spotted by Maximilian Präger. Also available via `git` with change history.

To build your own version from the FreeBSD sources, you will also need a computer running the same version of FreeBSD you wish to build. If your FreeBSD version is too old, a newer source tree's `buildworld` may fail. I recommend first building, installing and booting the same version of the source on the build system as you plan to cross-build for the router. Refer to The FreeBSD Handbook for more information about obtaining FreeBSD sources, the `buildworld` process and updating your FreeBSD system.

Even if you use another OS, you can probably install FreeBSD 10.x inside a virtual machine, and use it to build a version of itself for your EdgeRouter Lite.

The `mkerlimage` script does everything needed to build the image from a fresh copy of the FreeBSD sources. It performs the following steps:

1. If `sys/mips/conf/ERL` doesn't exist, creates a simple one based on the `OCTEON1` configuration.
2. Sets `TARGET=mips` and `TARGET_ARCH=mips64`, then invokes `buildworld` and `buildkernel` to build the base system and kernel.
3. Invokes `installworld` and `distribution` to install the newly built system to a temporary root.
4. Packages the whole thing up in a ~760MByte image, consisting of an MS-DOS partition for the kernel (so that U-Boot can read it), and a UFS fileystem for the rest. (Note that you can always grow the filesystem to the size of the media after installing and booting the image.)

Please review the script before running it. In particular, make sure that you check the following:

- The FreeBSD build system stores intermediate object files in `/usr/obj` by default. For the mips64 build, this will require at least 1.3GBytes of free space. Set the `MAKEOBJDIRPREFIX` environment variable if you want to put them somewhere else.
- The script itself requires at least 2.0GBytes of temporary space to hold the installation root, UFS image, and final result image before compression. Check the `temp_root` setting at the top of the script, or it will all go to `/tmp` by default.
- Check that the number of jobs in the `make_args` setting at the top of the script is appropriate for your build machine.

On success, you will have an `xz`-compressed image, ready to decompress and `dd` onto the destination. Follow the installation instructions, above.

Contact

The author of this page can be reached at erl@rtfm.net.