Tricking Squeezebox Server into downloading firmware files from local server

Gosh ... I'll never (as in _never_) cease my home internet again before moving out of the flat. Here's another workaround to trick my Squeezebox Server into downloading firmware files from a local server - just because of lack of internet connection. Dooooh!

After connecting my new SqueezeBox Radio, I had to take already some efforts to get it up and running because of no internet.
But for some obscure reason, the firmware installed (an ancient 7.4 series) didn't work to well with my Squeezebox Server running on version 7.5.1.
Obviously the Squeezebox Server tried hard to download current firmware as seen in the logfile:

[11-07-12 22:21:22.9387] Slim::Utils::Firmware::downloadAsyncError (557) Warning: Firmware: Failed to download http://update.mysqueezebox.com/update/firmware/7.5.1/baby.version (Couldn't resolve IP address for: update.mysqueezebox.com), will try again in 10 minutes.

Looking for the obvious at the Squeezebox Community Forums I came around a hint to just download the firmware and copy it to the 'Firmware' folder manually.
Still, the server kept trying to download the firmware, even after restarting the service, completely ignoring the files I just dropped in there :-(
(note: we'll take about this again a bit later ...)

Well, I'm not giving in that easy, so I tried another way: I created the proper directory structure in the '~/Downloads' folder on my Mac:

```
$ cd ~/Downloads
$ mkdir -p update/firmware/7.5.1
```

Then I downloaded the firmware files (actually the files 'baby.version', 'baby.version.sha', 'baby_7.5.1_r9218.bin' and 'baby_7.5.1_r9218.bin.sha') in my local '~/Downloads/update/firmware/7.5.1' folder.

Afterwars I started the built-in python web-server of OS X from the '~/Downloads' folder:

```
$ cd ~/Downloads
$ sudo python -m SimpleHTTPServer 80
Password:
Serving HTTP on 0.0.0.0 port 80 ...
```

Since my Squeezebox Server runs from a QNAP NAS, I logged into it through SSH.
QNAP, as many other vendors, use Linux as operating system. As I don't have a local DNS server in my network, my intention was to override the IP address of 'update.mysqueezebox.com' through a manual entry in the /etc/hosts file on the NAS server, so I added this new line to it:

```
192.168.100.54 update.mysqueezebox.com
```

A quick check using 'ping' reveals that it works properly.

```
[/] # ping update.mysqueezebox.com
PING update.mysqueezebox.com (192.168.100.54): 56 data bytes
64 bytes from 192.168.100.54: icmp_seq=0 ttl=64 time=1.0 ms
64 bytes from 192.168.100.54: icmp_seq=1 ttl=64 time=1.0 ms
```

Inspecting the logfile after restarting Squeezebox Server revealed however that this didn't work as expected. It seemed like the application prefers DNS and ingores the local /etc/hosts on the NAS.
I checked in /etc/nsswitch.conf as well to see if we may have a different-than-default lookup order there:

```
$ grep -e hosts /etc/nsswitch.conf
#hosts:     db files nisplus nis dns
hosts:      files dns wins
```

Oh well, exactly what I expected to see. After all I didn't want to dig into this too much (at least not more as I did already).
So I just searched the code to see where the Firmware download is being handled. I ended up with the file at /share/MD0_DATA/.qpkg/SqueezeBoxServer/var/home/SqueezeboxServer/Slim/Utils/Firmware.pm.

I located the line where the download URL is constructed on line 432:

```
my $url = BASE() . '/' . $::VERSION . '/' . basename($file);
```

So I changed it temporarily to read like this:

```
#my $url = BASE() . '/' . $::VERSION . '/' . basename($file);
my $url = 'http://192.168.100.54/update/firmware/' . $::VERSION . '/' . basename($file);
```

After restarting Squeezebox Server it finally downloaded the firmware from my Mac's folder.

Checking back where the files had been actually stored, I saw that the new firmware ended up in '/share/MD0_DATA/.qpkg/SqueezeBoxServer/var/home/SqueezeboxServer/Cache/updates' and not '/share/MD0_DATA/.qpkg/SqueezeBoxServer/var/home/SqueezeboxServer/Firmware' as supposed.
Well, at least I tricked the whole thing into doing what I wanted and my Squeezebox Radio finally accepted to download the new firmware from the Squeezebox Server thereafter.

And yes, this whole trickery would have been obsolete if I had an internet connection. Of course, I could have enabled interconnection sharing on my Mac, which was hooked up to my iPhone. Now don't ask me why, but somehow that was not working. After setting the network default gateway on the NAS to point to the Mac I'd still have no connection, so I suppose I either had one (or two) NAT sessions to much or some other thing on the Mac went terribly wrong. After all my approach worked out, that counts too ;-)