Part 72
-
mDNS

Version: 2021-02-02

mDNS

If you're tired of looking up the IP addresses of devices you frequently access via remote login, SSH, and other means on your home network, you can save yourself a lot of time by assigning an easy to remember `.local` address to the device.

Why Do I Want to Do This?

Most likely your home network uses DHCP IP assignments, which means that each time a device leaves the network and returns a new IP address is assigned to it. Even if you set a static IP for a frequently used device (e.g. you set your ESP32 to always be assigned to number `192.168.1.99`), you still have to commit that entirely unintuitive number to memory. Further, if you ever need to change the number for any reason you would have to remember a brand new one in its place.

Doing so isn't the end of the world, but it is inconvenient. Why bother with memorizing IP strings when you can give you local devices easy to remember names like esp32.local or tempsensor.local?

Now, some of you (especially those of you with a more intimate knowledge of DNS, domain naming, and other network address structures) might be wondering what the catch is. Isn't there an inherent risk or problem in just slapping a domain name onto your existing network? It's important here to make note of the *big* distinction between Fully Qualified Domain Names (FQDNs), which are officialy recognized suffixes for top-level domains (e.g. the .com) and domain names that are either not recognized by the global naming/DNS system or are outright reserved for private network usage.

For example, `.internal` is, as of this writing, not a FQDN; there are no registered domains anywhere in the world that end with `.internal` and thus if you were to configure your private network to use `.internal` for local addresses, there would be no chance of a DNS conflict. That could, however, change (though the chance is remote) in the future if `.internal` became an official FQDN and addresses ending in `.internal` were externally resolvable through public DNS servers.

Conversely, the `.local` domain, has been officially reserved as a Special-Use Domain Name (SUDN) specifically for the purpose of internal network usage. It will never be configured as a FQDN and as such your custom local names will never conflict with existing external addresses.

What Do I Need?

The secret sauce that makes the entire local DNS resolution system work is known as Multicast Domain Name Service (mDNS). Confusingly, there are actually two implementations of mDNS floating around, one by Apple and one by Microsoft. The mDNS implementation created by Apple is what undergirds their popular Bonjour local network discovery service. The implementation by Microsoft is known as Link-local Multicast Name Resolution (LLMNR). The Microsoft implementation was never widely adopted thanks to its failure to adhere to various standards and a security risk related to which domains could be captured for local use.

Apple's mDNS implementation Bonjour enjoys a much wider adoption rate, has better support, and a huge number of applications for platforms big and small.

If you have computers running Apple's OS X on your network, there's nothing you need to do beyond following along with the tutorial to set things up on the ESP32 (or other Linux device) side of things. You're set to go as your computers already support it.

If you're running a Windows machine that does not have iTunes installed (which would have installed a companion Bonjour client for mDNS resolution), you can resolve the lack of native mDNS support by downloading Apple's Bonjour Printer Service helper app here (https://support.apple.com/kb/DL999?locale=en_US). Although the download page makes it sound like it's a printer-only tool, it effectively adds mDNS/Bonjour support across the board to Windows.

Installing Bonjour Support on Your Raspberry Pi
Connect to your Pi either via keyboard and screen or via SSH. Once at the terminal, take a moment to update and upgrade.

```
sudo apt update
sudo apt -y upgrade
```

After the update/upgrade process is complete, it's time to install `avahi`, a fantastic little open source mDNS implementation. Enter the following command at the prompt

```
sudo apt-get install avahi-daemon
```

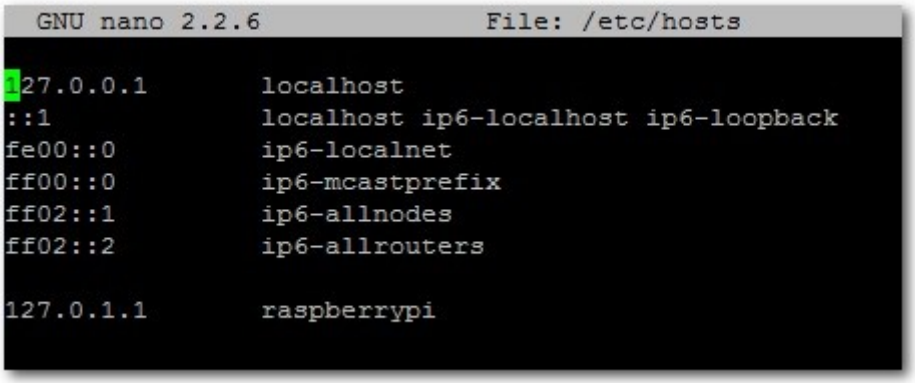
Note: On the latest Raspbian OSes, avahi is already installed which you might see running above command.

Once the installation process is complete, you don't even have to reboot the device. Your Raspberry Pi will begin immediately recognizing local network queries for its hostname (by default "`raspberrypi`") at `raspberrypi.local`.

Changing the Host on your Pi
If you want to give a more specific name to your Raspberry Pi, then you need to do the following. Type the following command to open the hosts file:

```
sudo nano /etc/hosts
```

Your hosts file will look like so:

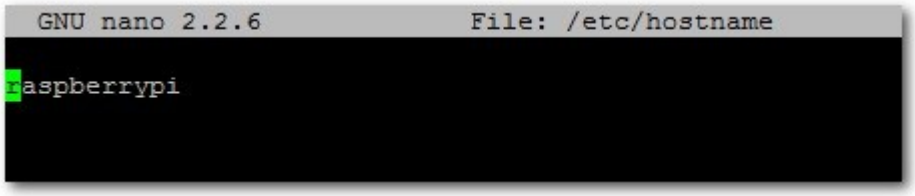


Leave all of the entries alone except for the very last entry labeled 127.0.1.1 with the hostname "raspberrypi". This is the only line you want to edit. Replace "raspberrypi" with whatever hostname you desire. (e.g. "`MeRasPi-WeatherStation`")
Press CTRL+X to close the editor and agree to overwrite the existing file and save it.

Back at the terminal, type the following command to open the hostname file:

```
sudo nano /etc/hostname
```

This file only contains your current hostname:



Replace the default "raspberrypi" with the same hostname you put in the previous step ("MeRasPi-WeatherStation")
Again, press CTRL+X to close the editor, agree to overwrite the existing file and save it.
Finally, we need to commit the changes to the system and reboot the system for the changes to take effect. At the terminal, enter the following command to commit the changes:

```
sudo /etc/init.d/hostname.sh
```

Follow that command with:

```
sudo reboot
```

Once the system comes back online, you can check the device list in your router to see if the new hostname has properly resolved and/or you can ping the Pi like this

```
ping MeRasPi3B-WeatherStation.local

Pinging MeRasPi3B-WeatherStation.local [192.168.1.63] with 32 bytes of data:
Reply from 192.168.1.63: bytes=32 time=1ms TTL=64
Reply from 192.168.1.63: bytes=32 time=1ms TTL=64
Reply from 192.168.1.63: bytes=32 time=1ms TTL=64
Reply from 192.168.1.63: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.63:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```