The Multicast Domain Name System (mDNS)

Pieter P

Multicast Domain Name System

DNS

Let's face it, constantly typing IP addresses is really cumbersome, and it would be impossible to remember all your favorite websites' addresses, especially if they use IPv6.
That's why domain names were introduced: a simple string of text that's easy to remember, for example www.google.com.

However, to send a request to a website, your computer still needs to know its IP address. That's where DNS comes in. It stands for Domain Name System, and is a way to translate a website's domain name to its IP address. On the Internet, there are a lot of DNS servers. Each DNS server has a long list of domain names and their corresponding IP addresses. Devices can connect to a DNS server and send a domain name, the DNS server will then respond with the IP address of the requested site.
You could compare it to a telephone directory: you can look up a name to find the corresponding phone number.

The DNS lookup happens completely in the background: when you go to a website in your browser, it will first send a request to a DNS server (this implies that the computer knows the IP address of the DNS server itself), wait for the response of the lookup, and then send the actual request to the right IP address.

mDNS

DNS works great for normal sites on the Internet, but most local networks don't have their own DNS server. This means that you can't reach local devices using a domain name, and you're stuck using IP addresses ...

Fortunately, there's another way: multicast DNS, or mDNS.
mDNS uses domain names with the *.local* suffix, for example http://esp8266.local. If your computer needs to send a request to a domain name that ends in *.local*, it will send a multicast query to all other devices on the LAN that support mDNS, asking the device with that specific domain name to identify itself. The device with the right name will then respond with another multicast and send its IP address. Now that your computer knows the IP address of the device, it can send normal requests.

Luckily for us, the ESP8266 Arduino Core supports mDNS:

```
#include <ESP8266WiFi.h>        // Include the Wi-Fi library
#include <ESP8266WiFiMulti.h>   // Include the Wi-Fi-Multi library
#include <ESP8266mDNS.h>        // Include the mDNS library

ESP8266WiFiMulti wifiMulti;     // Create an instance of the ESP8266WiFiMulti class, called 'wifiMulti'

void setup() {
  Serial.begin(115200);         // Start the Serial communication to send messages to the computer
  delay(10);
  Serial.println('\n');

  wifiMulti.addAP("ssid_from_AP_1", "your_password_for_AP_1");   // add Wi-Fi networks you want to connect to
  wifiMulti.addAP("ssid_from_AP_2", "your_password_for_AP_2");
  wifiMulti.addAP("ssid_from_AP_3", "your_password_for_AP_3");

  Serial.println("Connecting ...");
  int i = 0;
  while (wifiMulti.run() != WL_CONNECTED) { // Wait for the Wi-Fi to connect: scan for Wi-Fi networks, and connect to the
strongest of the networks above
    delay(1000);
    Serial.print(++i); Serial.print(' ');
  }
  Serial.println('\n');
  Serial.print("Connected to ");
  Serial.println(WiFi.SSID());              // Tell us what network we're connected to
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());           // Send the IP address of the ESP8266 to the computer

  if (!MDNS.begin("esp8266")) {             // Start the mDNS responder for esp8266.local
    Serial.println("Error setting up MDNS responder!");
  }
  Serial.println("mDNS responder started");
}

void loop() { }
```

Upload it and open ping again. Try to ping to *esp8266.local:*

```
user@computername:~$  ping esp8266.local
PING esp8266.local (10.92.237.128) 56(84) bytes of data.
64 bytes from 10.92.237.128: icmp_seq=1 ttl=128 time=5.68 ms
64 bytes from 10.92.237.128: icmp_seq=2 ttl=128 time=3.41 ms
64 bytes from 10.92.237.128: icmp_seq=3 ttl=128 time=2.55 ms
64 bytes from 10.92.237.128: icmp_seq=4 ttl=128 time=2.19 ms
64 bytes from 10.92.237.128: icmp_seq=5 ttl=128 time=2.29 ms
```

```
64 bytes from 10.92.237.128: icmp_seq=6 ttl=128 time=2.74 ms
^C
--- esp8266.local ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 2.190/3.148/5.687/1.202 ms
```

As you can see, ping will automatically find the IP address of the ESP for you.

mDNS is supported on Windows, OSX, Linux and iOS, but not (yet?) on Android.
It's a real shame that Android doesn't support it, you can help by starring this issue report for the Chromium project to ask for mDNS support in Chrome on Android.

Of course, you can change the domain name of the ESP by changing the parameter of `MDNS.begin`.