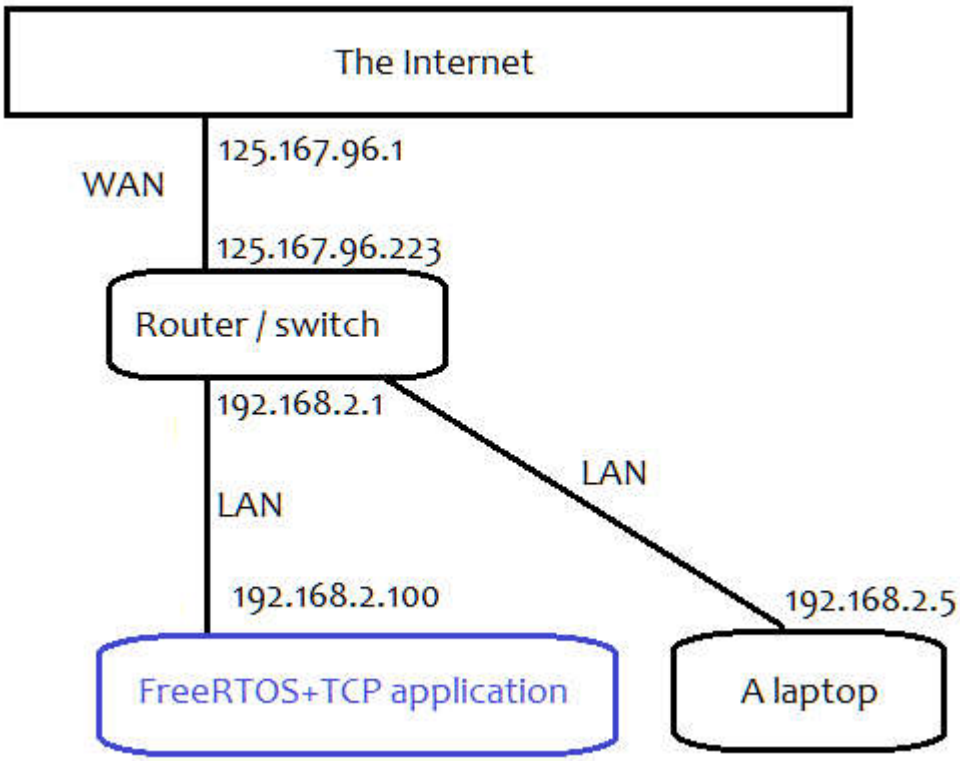
# FreeRTOS+TCP : From single to multiple interfaces.

**Single:**

Up until now, each FreeRTOS+TCP project had one physical interface, either a LAN or a WiFi. The driver for this interface is linked into the code and offers the possibility to send and receive packets. An application can only link with a single network interface, this is a static (compile-time) choice.

The library expects a network interface that exchanges raw IEEE 802.3 packets.

Here is an example of an application with a single interface:




An application can communicate with other nodes on the same LAN. The physical (MAC-) addresses are looked up by using ARP. The bindings can be stored in a cache.

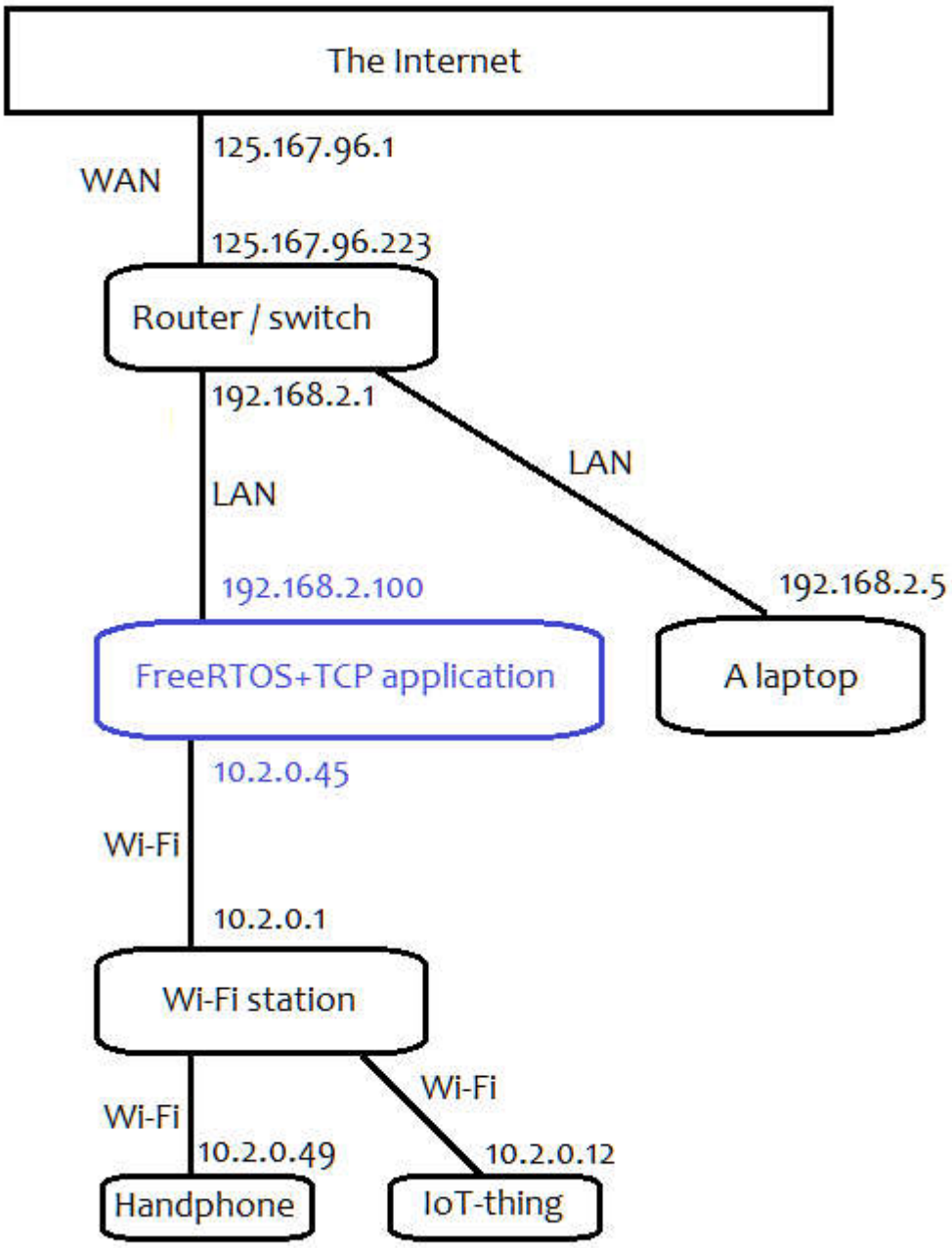
When an application wants to connect to the WEB, it must first find the remote IP-address. A DNS client takes care of this. Once the IP-address is known, the application must know a gateway address. This is normally a network router. If there is no gateway defined, a packet will be dropped.

The routing logic in the single driver is very simple. If the peer's IP-address is within the network mask, packets will be sent directly. For instance, the laptop here above ( 192.168.2.5 ) can be reached directly.
Packets to all non-matching IP-addresses will be passed to the Gateway address 192.168.2.1.

**Multiple:**

More and more applications want to make use of more than one network interface. In the multi library this is possible.

Here below a configuration with 2 network interfaces:




Now the FreeRTOS application is connected to a LAN and also to a Wi-Fi station.
It uses the addresses 192.168.2.100 and 10.2.0.45. These addresses can be configured either statically or automatically by the use of DHCP, or router advertisement ( "RA" ) in case of IPv6.

IP-routing becomes a little more complex now. Suppose the application wants to send a UDP packet to the IoT-thing. The library will try to find a matching network address:

| IP-address | Network | Mask | Interface | Gateway |
|---|---|---|---|---|
| 192.168.2.100 | 192.168.2.0 | 255.255.255.0 | LAN | 192.168.2.1 |
| 10.2.0.12 | 10.2.0.0 | 255.0.0.0 | Wi-Fi | - |

The solution is easy: 10.2.0.12 matches with the network address 10.2.0.0, which is the Wi-Fi network. No gateway is needed.

Now it wants to load a page from Google.com. DNS returns the address 64.233.167.113. This address does not match with either the LAN or the Wi-Fi network address.

The library will iterate through the list of interfaces again and look for a Gateway. The first gateway found will be used.

**Two networks, same network address.**

Some users on the FreeRTOS forum asked if two networks may have the same network address? I think that will bring confusion: the algorithm says that the first interface with a matching network address shall be used.
I was polite and I didn't say that this is bad design. Sometimes a programmer cannot influence the network infrastructure.

For an incoming ping request this is simple: if a request comes in on interface N, the answer will be returned to that same interface N.

The same for an incoming TCP connection: the very first SYN packet will set the MAC-address and the interface used.

As for UDP packets: incoming packets will be replied to through the interface that received the packet. For outgoing packets, there may be confusion. It could send out ARP requests to all interfaces, but that has not been implemented yet.

**All networks have a different network address.**

This is a clear situation, the library has no doubt which interface shall be used.

In the above text, I left out the so-called end-points. It is possible to assign more than one IP-address to an interface. For instance:

```
IPv6 public: 2001:470:ec54::4514:89d5:4589:8b79
IPv6 local : fe80::9355:69c7:585a:afe7
IPv4       : 192.168.2.100
IPv4       : 172.16.0.100
```

Each of these end-point has a list of properties: use DHCP / RA, Gateway, Network mask, a DNS address.

To be continued.