Python: how to calculate the number of available IP addresses in a subnet

Calculating the number of available IP addresses in a subnet is a fundamental skill for networking professionals. In this article, we will illustrate how to use Python to perform these calculations, taking advantage of the potential of the language and some useful libraries.

Before proceeding with the Python code, it is important to understand some basic concepts about subnetting and IP addresses. An Internet Protocol (IP) address is a unique identifier assigned to each device on a network. IP addresses can be of two types: IPv4 and IPv6. Here we will focus on IPv4 addresses, which are made up of 32 bits and are usually expressed in dotted decimal notation (example: 192.168.1.1).

A subnet is a division of an IP network into smaller blocks, which allows for more efficient traffic management and greater security. The capacity of a subnet is determined by its netmask, which specifies how many bits of the IP address are reserved for identifying the network and how many for hosts within that network.

The formula to calculate the number of available addresses in a subnet is:

$$\text{Number of addresses} = 2^{(\text{32-bit number of netmask})}$$

For a subnet with a 24-bit netmask (commonly expressed as /24 or 255.255.255.0), the calculation would be:

$$2^{(32-24)} = 256$$

Of these 256 addresses, one is reserved for the network address and one for the broadcast, thus leaving 254 usable addresses for hosts.

Python offers several libraries that can help calculate available IP addresses. One of the most common is `ipaddress`. Here is a code example that uses this library to calculate the number of available IP addresses in a subnet:

```
import ipaddress

def calc_available_addresses(subnet):
     # Creating an IP network object using the
ipaddress library
     net_obj = ipaddress.ip_network(subnet)

     # Calculating the total number of addresses in
the subnet
     total_addresses = net_obj.num_addresses

     # Stealing network and broadcast addresses
     available_addresses = total_addresses - 2

     return available_addresses

# Example of use
subnet = "192.168.1.0/24"
available_addresses =
calc_available_addresses(subnet)
print(f"Usable IP addresses: {available_addresses}")
```

In conclusion, calculating the number of IP addresses in a subnet is essential for planning and managing networks. Using Python and its libraries, you can greatly simplify this task, making calculations fast and precise. With proper knowledge and the right tools, managing subnets becomes a much more manageable task.