LINFO2142 : Multi-homing report

Targonski Krystian - 42942000

This report is a follow-up to the lab session conducted in the LINFO2142 course. It was focused on multi-homing, and how a device manages its network communication when multiple IP addresses are available, even across different ISPs.

In my setup, I had three IPv6 addresses: two from Starlink, `2a0d:3344:18fa:4916/64` and `fdcc:1705:6306:16/64`, and one from Tadaam, `fd4e:c6e4:ca93:2/64`. No IPv4 addresses were available. Throughout this report, any reference to "source IP" refers to the IP used as the output interface—the one used to send packets. In most cases it was the one my device also received packets on.

The limitation of not having an IPv4 address became apparent quite quickly when I tried to browse the internet as a typical user. Several websites were not reachable-for example, stackoverflow.com and starlink.com (our own provider's website, which I found quite amusing).

Others worked fine, such as most UCLouvain websites, with the exception of monhoraire.uclouvain.be (which turned out to be a separate bug). Since the unreachable sites were not actually down, I suspected a DNS-related issue. To verify this, I used the **ping** and **dig** commands.

What I observed was that, in normal conditions, my device only queried IPv6 addresses, which made sense since I don't have an IPv4 one, thus making **A** queries essentially useless. However those **AAAA** queries for unreachable websites didn't return an address for me to contact! This clarified why I couldn't open certain websites, they didn't have an IPv6, making communication impossible with them.

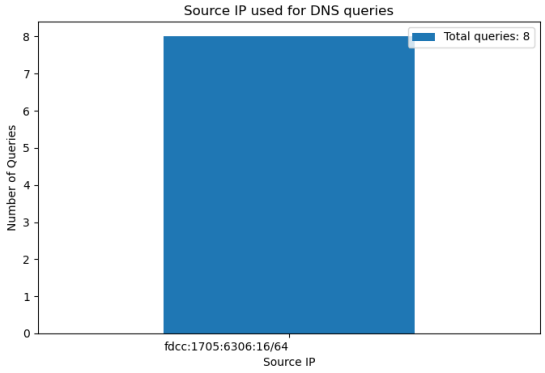

Figure 1: IP address selected for DNS queries

Another observation while examining the queries, was the seemingly exclusive use of `fdcc:1705:6306:16/64` (see fig.1) over other available addresses, each tested to be functional. This led me to believe that my device doesn't randomly use available addresses, but has a clear way to manage them. To check this hypothesis and try to essentially guess the selection method, I first checked when each address was used, what I observed can be seen in fig.2.

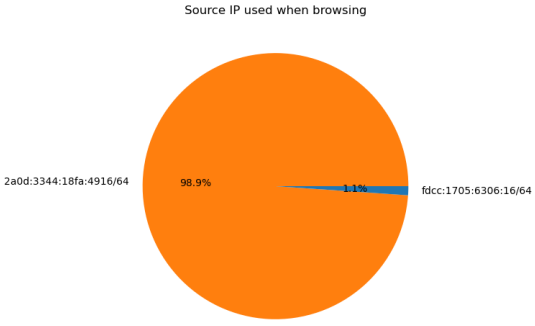

Figure 2: Normal browsing IP addresses usage

This was captured whilst simply browsing websites I could connect to. My first conclusion was that, for some reason one address had to be chosen as the preferred one (`2a0d:3344:18fa:4916/64`) for general traffic like UDP, TCP and I even saw some QUIC, another for DNS traffic (`fdcc:1705:6306:16/64`) and the last one was used only once (fig.3, only one address is mine, the 1st), which led me to believe that it was some sort of backup (`fd4e:c6e4:ca93:2/64`).

To try and confirm this hypothesis, I decided to capture what happens when the most used one was to go down, on Linux it can be done by using the **ip** command. My observations can be seen in fig.5. The moment the preferred one goes down can be seen at around 17s but then something I wasn't expecting happened, `fdcc:1705:6306:16/64` took over the traffic, and when the main one came back after some time (around `valid_lft` time) it went back to doing DNS queries. I was expecting `fd4e:c6e4:ca93:2/64` to show but it didn't (figs.4 & 5), meaning my hypothesis was void.

Source	Destination	Protocol	Length	Info
fd4e:c6e4:ca93:2:6626:7f09:4f3c:68ec	ff02::1	ICMPv6	86	Neighbor Advertisement fd4e:c6e4:ca93:2:6626:7f09:4f3c:68ec (ovr) is at 28:d0:43:7b:bd:1a
fd4e:c6e4:ca93:2:4d3b:742c:48e3:a146	ff02::1	ICMPv6	86	Neighbor Advertisement fd4e:c6e4:ca93:2:4d3b:742c:48e3:a146 (ovr) is at 28:d0:43:7b:bd:1a
fd4e:c6e4:ca93:2:4f8f:60df:b77b:d3	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question

Figure 3: Tadaam's IP usage

This prompted me to do some research, and after reading `RFC6724`, the behavior made sense. This RFC describes the decision-making process for selecting which IP address is used as the source. The following rules are applied in their respective order, if one is accepted, then the address is selected :

- Rule 1: Prefer same address.
- Rule 2: Prefer appropriate scope.
- Rule 3: Avoid deprecated addresses
- Rule 4: Prefer home addresses.
- Rule 5: Prefer outgoing interface. (Rule 5.5: Prefer addresses in a prefix advertised by the next-hop.)
- Rule 6: Prefer matching label.
- Rule 7: Prefer temporary addresses.
- Rule 8: Use longest matching prefix.

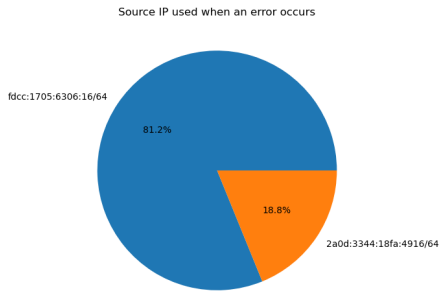

Figure 4: Issues on preferred IP address

In my case, the most relevant one was the last: when communicating with another device, it is preferred to select a source IP with the longest prefix match with destination address. Or at least that's what seemed to have the most sense.

Most of the destination IPs I contacted matched best with my `2a0d:3344:18fa:4916/64` address, which explains why it was "preferred" and mainly used. I then realized that the DNS server's IP had the closest match with `fdcc:1705:6306:16/64`, explaining why this address was handling all DNS queries. The lack of use of the last IP address `fd4e:c6e4:ca93:2/64` is also explained by this rule: if I had contacted a server whose IP matched this address more closely than the other two, it would have been used (as seen in 3).

Finally, I would like to mention, a similar behavior observed on Windows. I tested both Linux and Windows to see if there were any notable differences, and one major difference stood out. For each IP it received, Windows created a second, temporary address that it used instead. This behavior aligns with `RFC6724`, which states that temporary addresses are preferred over regular ones.

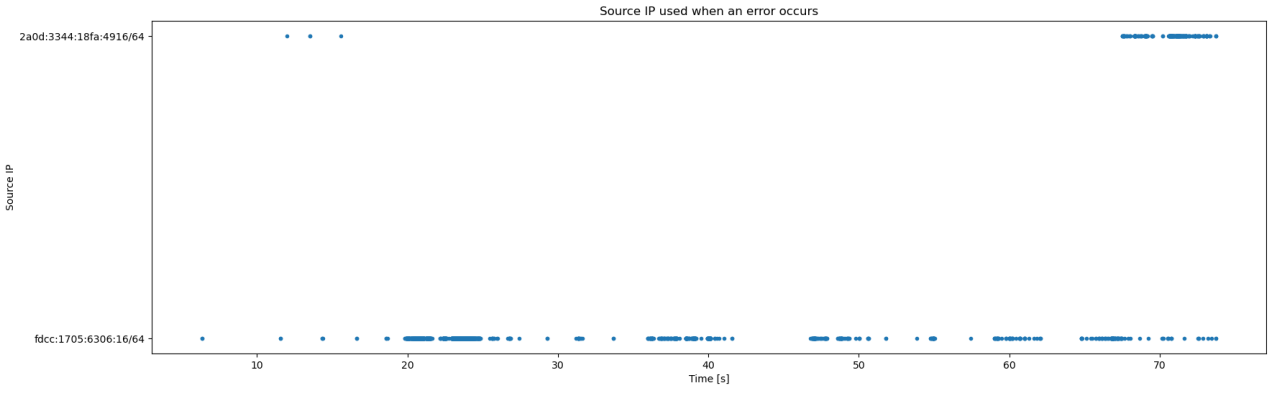

Figure 5: Removal of preferred IP address

My wireshark captures are available on the following git along with some screenshots I took : `https://github.com/Seito1090/LINFO2142_wireshark_packets`