B.4. ip neighbor

Part of the **iproute2** command suite, **ip neighbor** provides a command line interface to display the neighbor table (ARP cache), insert permanent entries, remove specific entries and remove a large number of entries. For peculiarities and commonalities of the **iproute2** tools, refer to Section H.2, “Some general remarks about **iproute2** tools”.

The more commonly used analog to **ip neighbor show**, **arp -n** displays the ARP cache in a possibly more recognizable format.

Example B.13. Displaying the ARP cache with ip neighbor show

```
[root@tristan]# ip neighbor show
192.168.99.254 dev eth0 lladdr 00:80:c8:f8:5c:73 nud reachable
```

On routers and other machines with large ARP caches, you may find you wish to look at the ARP cache only on a particular interface. By specifying the interface on which you wish to see the neighbor table, you can limit the output.

Example B.14. Displaying the ARP cache on an interface with ip neighbor show

```
[root@wan-gw]# ip neighbor show dev eth0
205.254.211.39 lladdr 00:02:b3:a1:b8:df nud delay
205.254.211.54 lladdr 00:d0:b7:80:ce:ce nud delay
205.254.211.179 lladdr 00:80:c8:f8:5c:72 nud reachable
```

Another way to limit the output is to specify the subnet in which you are interested. Simply append the subnet specification to the command.

Example B.15. Displaying the ARP cache for a particular network with ip neighbor show

```
[root@masq-gw]# ip neighbor show 192.168.100.0/24
192.168.100.1 dev eth3 lladdr 00:c0:7b:7d:00:c8 nud stale
192.168.100.17 dev eth0 lladdr 00:80:c8:e8:4b:8e nud reachable
```

Note that in the case of `masq-gw`, there are neighbor table entries for IPs on more than one interface, because `masq-gw` breaks the 192.168.100.0/24 network into two parts. This is an advanced technique described in fuller detail in Section 9.3, “Breaking a network in two with proxy ARP”.

In addition to displaying the neighbor table, it is possible to make static mappings. For paranoid systems administrators, who do not want to enable ARP on their networks or on particular links, the **ip neighbor add** command may prove useful. Refer to Section 2.3, “ARP filtering” for a discussion of the ramifications of disabling ARP.

In Example B.16, “Entering a permanent entry into the ARP cache with ip neighbor add”, let's assume that the service router is incapable of correctly answering ARP requests. The administrator of `masq-gw` could make a permanent entry in the ARP cache mapping 192.168.100.1 to the link layer address of `service-router`.

Example B.16. Entering a permanent entry into the ARP cache with ip neighbor add

```
[root@masq-gw]# ip neighbor add 192.168.100.1 lladdr 00:c0:7b:7d:00:c8 dev eth3 nud permanent
```

This creates an entry in the neighbor table which maps 192.168.100.1 to link layer address 00:c0:7b:7d:00:c8. Subsequent IP packets bound for 192.168.100.1 will be encapsulated in Ethernet frames with 00:c0:7b:7d:00:c8 in the destination bytes. This permanent mapping cannot be overridden by ARP. It would need to be removed with **ip neighbor delete**.

For those who insist on such a thing, there is support for creating and deleting proxy ARP entries with **ip neighbor**, although this has been deprecated. For a long discussion of this topic, see this discussion on the kernel mailing list. Other tools should be used to create proxy ARP entries. Refer to Section B.1, “**arp**”, Section 9.3, “Breaking a network in two with proxy ARP” and Section 2.2, “Proxy ARP”.

Example B.17. Entering a proxy ARP entry with ip neighbor add proxy

```
# -- this is deprecated; use arp or kernel proxy_arp instead --#
[root@masq-gw]# ip neighbor add proxy 192.168.100.1 dev eth0
# -- this is deprecated; use arp or kernel proxy_arp instead --#
```

Strangely, the **ip neighbor show** command does not display any entries added and deleted with **ip neighbor add proxy**, so **arp** is required to view these entries. In short, don't use **ip neighbor add proxy**.

Entries can also be modified at any time. This allows learned entries to be replaced with static entries if there's already an entry in the ARP cache for a specified IP.

Example B.18. Altering an entry in the ARP cache with ip neighbor change

```
[root@tristan]# ip neighbor add 192.168.99.254 lladdr 00:80:c8:27:69:2d dev eth3
RTNETLINK answers: File exists
[root@tristan]# ip neighbor show 192.168.99.254
192.168.99.254 dev eth0 lladdr 00:80:c8:f8:5c:73 nud reachable
[root@tristan]# ip neighbor change 192.168.99.254 lladdr 00:80:c8:27:69:2d dev eth3
[root@tristan]# ip neighbor show 192.168.99.254
192.168.99.254 dev eth0 lladdr 00:80:c8:27:69:2d nud permanent
```

To remove the entry we added above in Example B.16, “Entering a permanent entry into the ARP cache with ip neighbor add”, we could run the following command. This invalidates the entry forcing the NUD of the entry into *failed* state.

Example B.19. Removing an entry from the ARP cache with ip neighbor del

```
[root@masq-gw]# ip neighbor del 192.168.100.1 dev eth3
[root@masq-gw]# ip neighbor show dev eth3
192.168.100.1  nud failed
```

Subsequent attempts to reach the IP address 192.168.100.1 will require the generation of a new ARP request, which (you hope!) returns the new or currently available link layer address.

While I have never found a good use for the **ip neighbor flush** command, it is provided, and accepts a destination network address as an argument. Without a destination network address, an interface specification is required.

Example B.20. Removing learned entries from the ARP cache with ip neighbor flush

```
[root@tristan]# ip neighbor flush dev eth3
```

Although it is not commonly required, the **ip neighbor** tool is a convenient tool for displaying and altering the ARP cache (neighbor table).

Prev B.3. **ip link** | Up Home | Next B.5. **mii-tool**