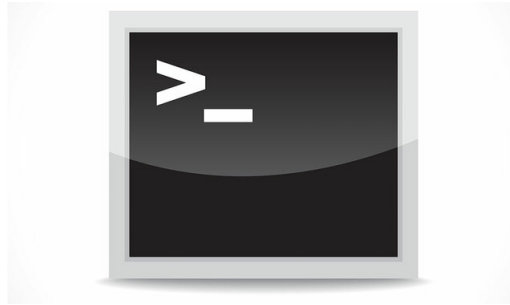


## How to clear ARP cache on Linux / Windows for a single IP address / Flush All IPs ARP cache

Author : admin



On times of Public Internet IP migration or Local IPs between Linux servers or especially in clustered **Linux Application Services** running on environments like **Pacemaker / Corosync / Heartbeat** with services such as **Haproxy**.

Once an IP gets migrated due to complex network and firewall settings often the Migrated IP from **Linux Server 1 (A)** to **Linux Server 2 (B)** keeps time until a request to reload the Internet server IP ARP cache with to point to the new IP location, causing a disruption of accessibility to the Newly configured IP address on the new locations. I will not get much into details here what are the **ARP (Address Resolution protocol)** and Network ARP records on a Network attached Computer and how they correspond uniquely to each IP address assigned on Ethernet or Aliased network Interfaces (*eth0 eth0:1 eth0:2*) . But in this article, I'll briefly explain once **IP Version 4 address** is migrated from one server Data Center location to another DC, how the unique corresponding ARP record kept in OS system memory should be flushed in the ARP corresponding Operating System so called **ARP table** (of which you should think as a logical block in memory keeping a **Map of where IP addresses are located physically on a Network recognized by the corresponding Unique MAC Address**).

### 1. List the current ARP cache entries do

**Arp** is part of **net-tools** on **Debian GNU / Linux** and is also available and installed by default on virtually any Linux distribution **Fedora / CentOS / RHEL / Ubuntu / Arch Linux** and even m\$ Windows NT / XP / 2000 / 10 / whatever, the only difference is Linux tool has a bit of more functionality and has a bit more complex use.

Easiest use of arp on GNU / Linux OS-es is.

**# arp -an**

```
root@pcfreak:~# arp -a
? (192.168.0.170) at 1c:39:47:cd:b0:b3 [ether] on eth1
? (192.168.0.8) at 00:24:01:90:8f:37 [ether] on eth1
? (169.254.137.13) at 1c:39:47:cd:b0:b3 [ether] on eth1
? (192.168.6.1) at 00:30:48:a0:0b:44 [ether] on eth0
ip-65.bergon.net (83.228.93.65) at 00:30:48:a0:0b:44 [ether] on eth0
? (192.168.0.210) at 00:0a:e4:c9:7b:59 [ether] on eth1
root@pcfreak:~# arp -d 192.168.0.8
root@pcfreak:~#
```

The -a lists all records and -n flag is here to omit IP resolving as some IPs are really slow to resolve and output of command could get lagged.

## 2. Delete one IP entry from the cache

Assuming only one IP address was migrated, if you want to delete the IP entry from local ARP table on any interface:

**# arp -d 192.168.0.8**

It is useful to delete an ARP cached entry for IP address only on a certain interface, to do so:

**# /usr/sbin/arp -i eth1 -d 10.0.0.1**

### **3. Create ARP entry MAC address with a static one for tightened security**

A useful Hack is to (assign) / bind specific Static MAC addresses to be static in the ARP cache, this is very useful to improve security and fight an ARP poisoning attacks.

Doing so is pretty easy, to do so:

Above will statically make IP **192.168.0.8** to always appear in the ARP cache table to the MAC **00:50:ba:85:85:ca**. So even if we have another system with the same MAC trying to spoof our location and thus break our real record location for the Hostname in the network holding in reality the MAC **00:50:ba:85:85:ca**, poisoning us trying to make our host to recognize **192.168.0.8** to a different address this will not happen as the static ARP will be kept unchanged in ARP caching table.

```
# arp -s 192.168.0.8 00:50:ba:85:85:ca
```

### **4. Flush all ARP records only for specific Ethernet Interface**

After the IP on interface was migrated run:

```
# ip link set arp off dev eth0 ; ip link set arp on dev eth0
```

### **5. Remove a set of few IPs only migrated ARP cache entries**

```
# for i in 192.168.0.1 10.0.0.1 172.168.0.3; do sudo arp -d $i; done
```

Once old ARP entries are removed the arp command would return as:

```
linux:~$ arp
? (192.168.0.8) at  on eth1
? (172.168.0.3) at  on eth2
```

The 192.168.0.8 / 172.168.0.3 entry now shows as incomplete, which means the ARP entry will be refreshed when it is needed again, this would also depend on the used network switches / firewalls in the network settings so often could take up to 1 minute or so..

## 6. Flush all ARP table records on Linux

```
root@pcfreak:~# ip -s -s neigh flush all
*** Round 1, deleting 1 entry ***
168.172.0.1 dev enp0s1 lladdr 01:08:3d:w4:u8:e1 ref 1 used 0/0/1 probes 4 REACHABLE
*** Round 2, deleting 1 entry ***
168.172.0.2 dev enp0s1 lladdr 00:09:2s:a2:t5:y2 ref 1 used 0/0/1 probes 4 REACHABLE
*** Round 3, deleting 1 entry ***
168.172.0.3 dev enp0s1 lladdr 00:01:7b:d6:a3:u8 ref 1 used 0/0/1 probes 4 REACHABLE
*** Round 3, deleting 1 entry ***
192.168.0.1 dev eth0 lladdr 00:24:01:90:8f:37 ref 1 used 0/0/1 probes 4 REACHABLE
root@pcfreak:~#
root@pcfreak:~# █
```

```
# ip -s -s neigh flush all
```

## 7. Delete ARP Cache on FreeBSD and other BSDs

```
# arp -d -a
```

## 8. Flush arp cache on Windows

Run command prompt as Administrator -> (**cmd.exe**) and do:

```
C:\> ipconfig /all  
netsh interface ip delete arpcache
```

## 9. Monitoring the arp table

On servers with multiple IP addresses, where you expect a number of IP addresses migrated to change it is useful to use **watch** + **arp** like so:

```
# watch -n 0.1 'arp -an'
```

```
...
```

The **-n 0.1** will make the `arp -an` be rerun every 10 milliseconds and by the way is a useful trick to monitor stuff returned by commands that needs a higher refresh frequency.

## Conclusion

In short in this article, was explained how to **list your arp cache table**. The **arp** command is also available both on Linux and Windows) and as integral part of OS networking it is useful to check thoroughly to its man page (**man arp**).

Explained was how to create **Static ARP table records** to prevent ARP poisoning attacks on a server. I went through how to delete only a single **ARP records** (in case if) only certain IPs on a host are changed and an **ARP cache entry reload** is needed, as well as how to flush the complete set of ARP records need to get refreshed, sometimes useful on networks with **Buggy Network Switches** or when completely changing the set of IP-addresses assigned on a server host.