



LAB 1 ARP (Address Resolution Protocol)

CMPE 208: NETWORK ARCHITECTURE AND PROTOCOL

Prof. Shai Silberman

Submitted by

Team 12

Team Members

Akash Kumar Athghara

akash.athghara@sjsu.edu

010757929

Kshama Shalini

kshama.shalini@sjsu.edu

010763792

Gunveet Singh Arora

gunveet.singh@sjsu.edu

010641904

Sashank Malladi

sashank.malladi@sjsu.edu

010466651

Contribution of each team members

Akash Kumar Athghara:

Environment Setup, ARP Components and Address resolution between devices on same LAN.

Kshama Shalini:

ARP Message Format, Payload Format and Packet Header

Gunveet Singh Arora:

Environment setup, Commands on ARP Cache table and Conclusion.

Sashank Malladi:

Concept of Gratuitous in ARP, Attacks involving ARP and Learning Outcomes.

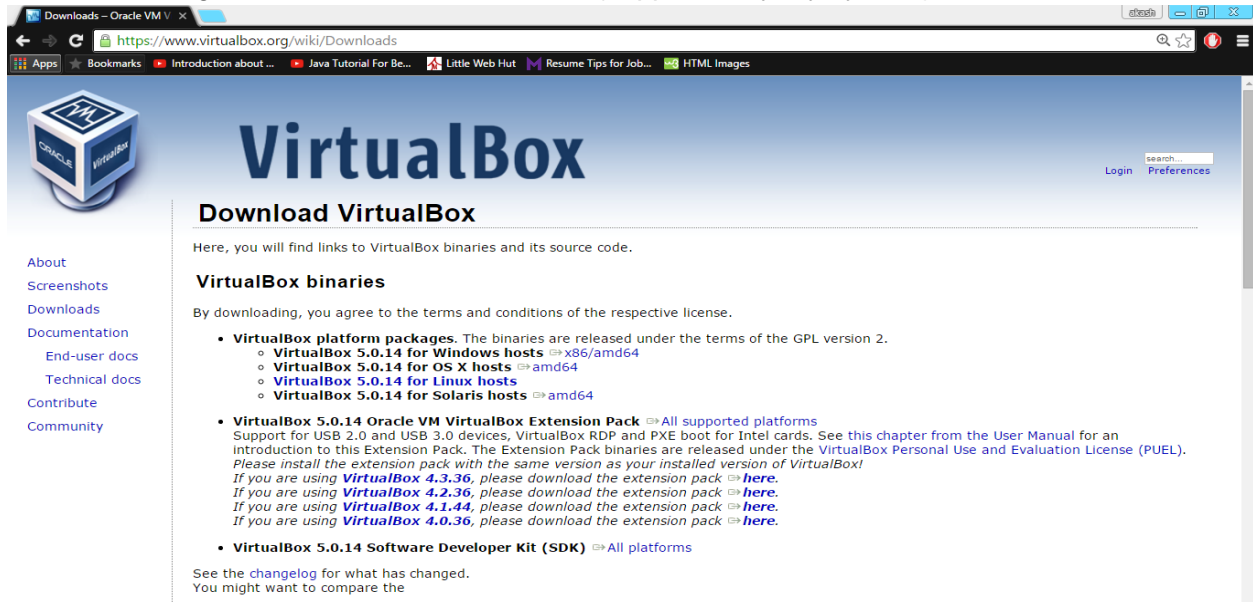
LAB SETUP:

The following steps need to be followed to set up the Lab:

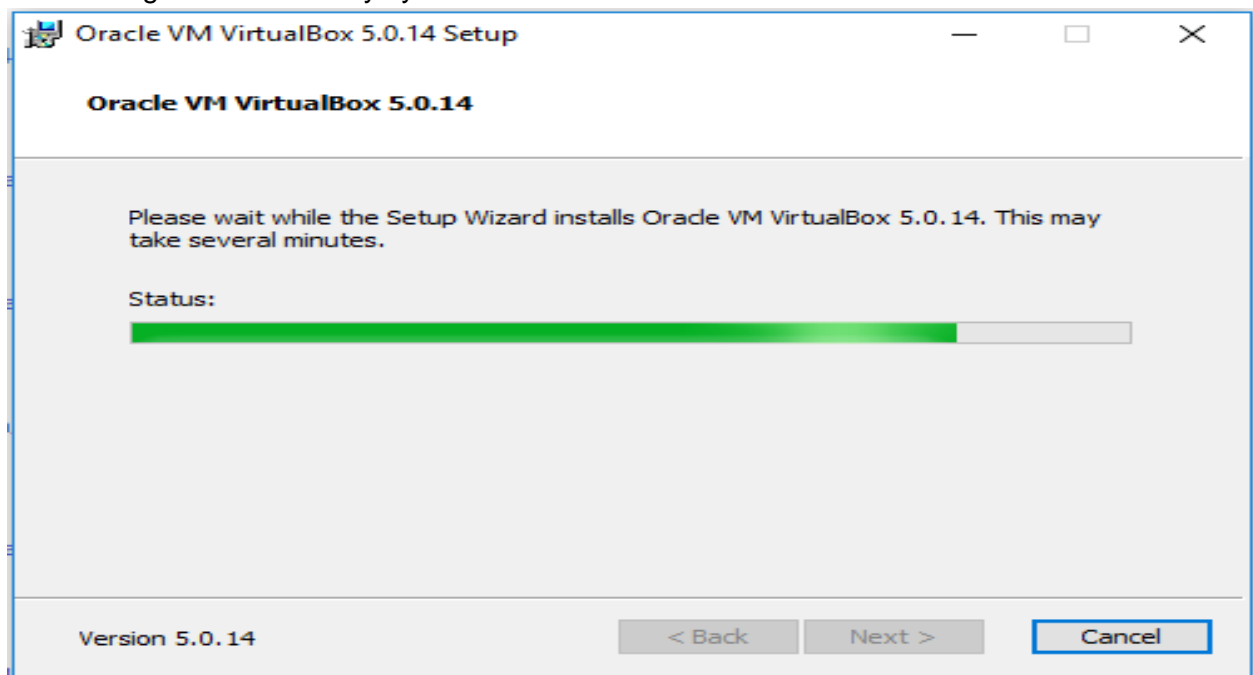
1. Installing VirtualBox
2. Installing Ubuntu inside VirtualBox
3. Installing Wireshark inside Ubuntu

1. Installing VirtualBox

A: Downloading Windows VirtualBox Version (supported by my system):



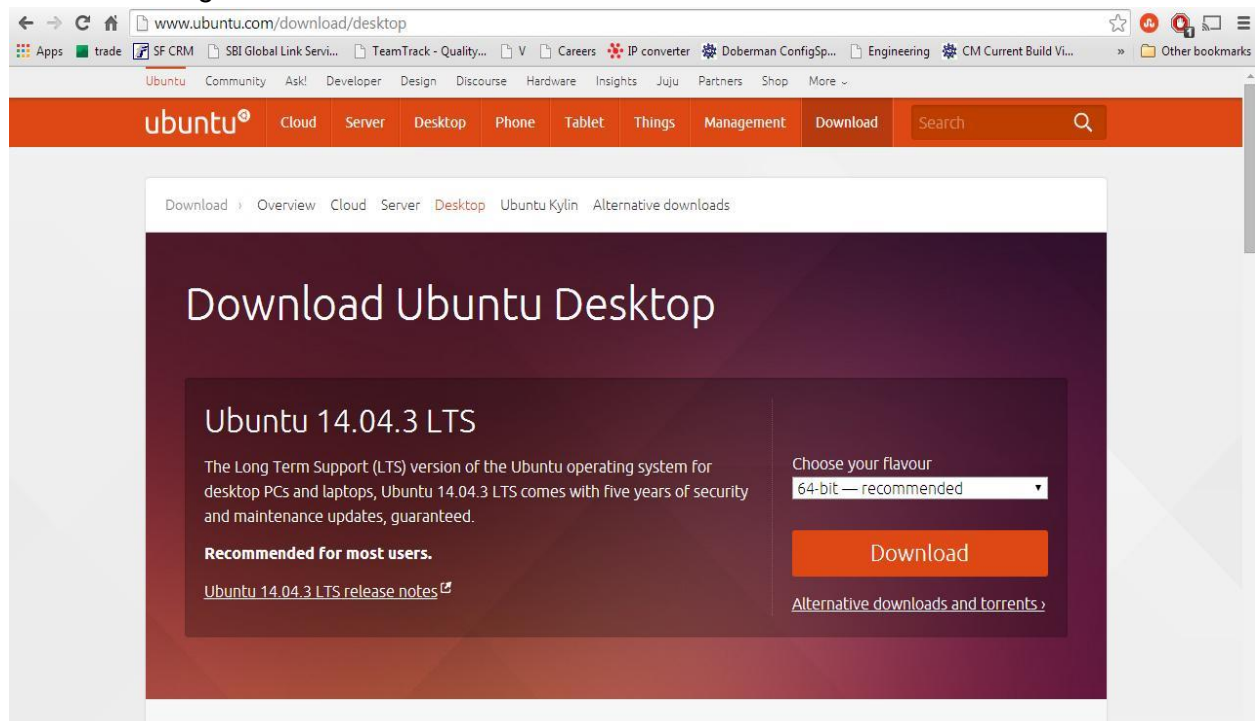
B: Installing VirtualBox in my system:



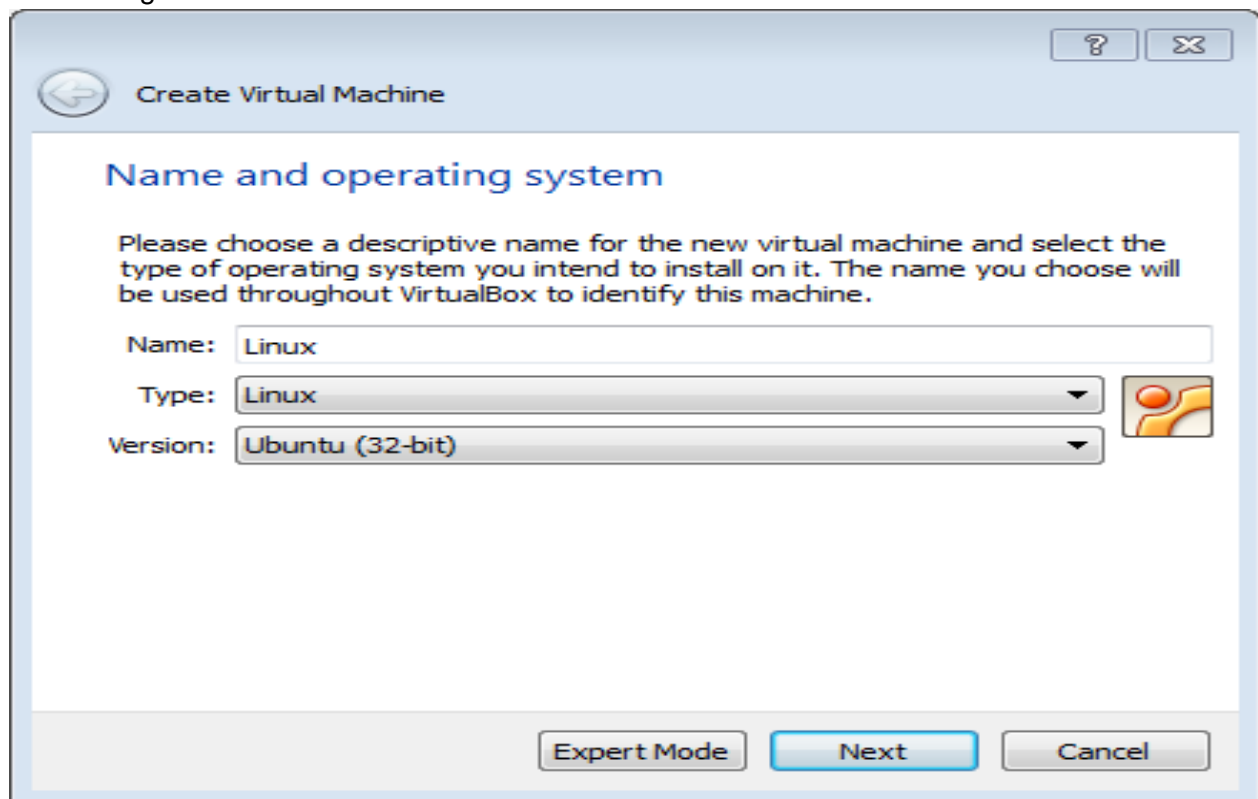


2. Installing Ubuntu inside VirtualBox

A: Downloading Ubuntu to be installed in VirtualBox:



B: Creating the Linux Virtual Machine:



The screenshot shows the 'Create Virtual Machine' window. The title bar includes a back arrow, the text 'Create Virtual Machine', and help and close buttons. The main heading is 'Name and operating system'. Below this, a paragraph explains that the user should choose a name and an operating system type. The 'Name' field contains 'Linux'. The 'Type' dropdown is set to 'Linux', and the 'Version' dropdown is set to 'Ubuntu (32-bit)'. To the right of these fields is a small icon of a person. At the bottom, there are three buttons: 'Expert Mode', 'Next', and 'Cancel'.

Create Virtual Machine

Name and operating system

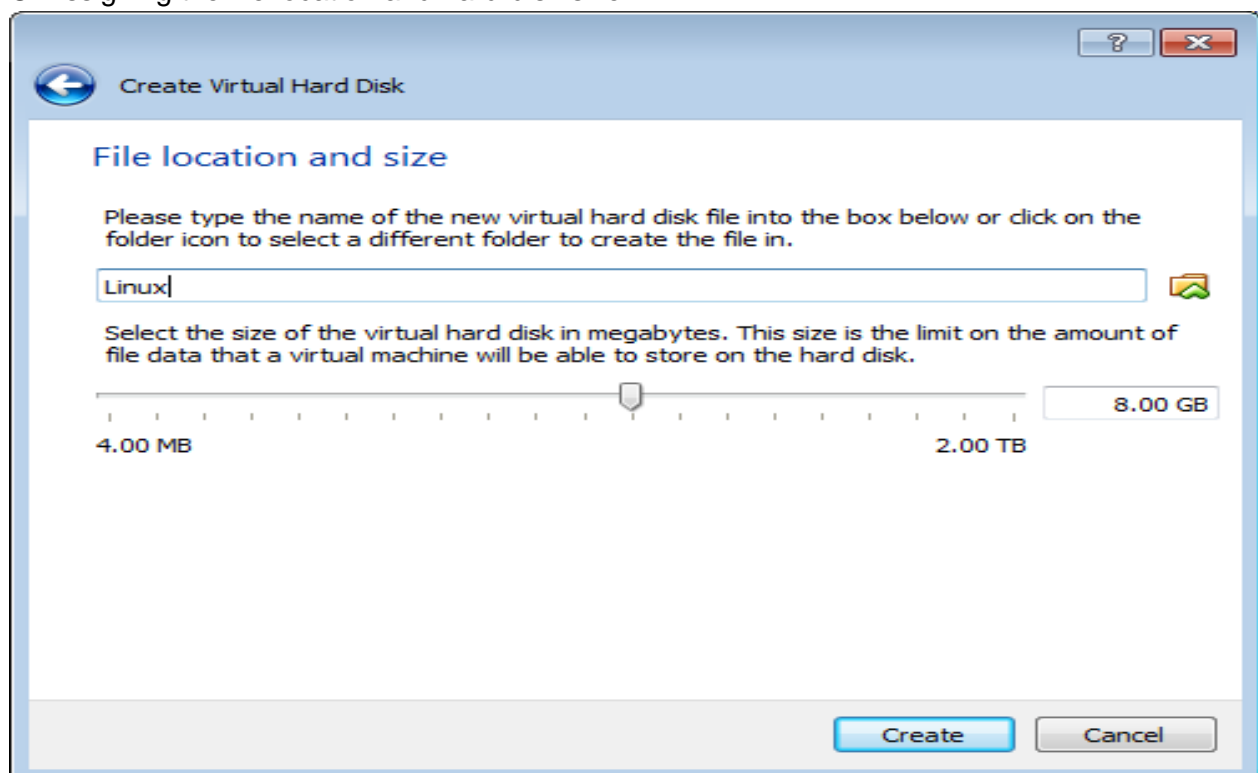
Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

Type:

Version:

C: Assigning the file location and hard-disk size:




The screenshot shows the 'Create Virtual Hard Disk' window. The title bar includes a back arrow, the text 'Create Virtual Hard Disk', and help and close buttons. The main heading is 'File location and size'. Below this, a paragraph explains that the user should type the name of the new virtual hard disk file or click on a folder icon. The 'Name' field contains 'Linux'. Below this, a paragraph explains that the user should select the size of the virtual hard disk in megabytes. A slider bar is shown with a range from 4.00 MB to 2.00 TB. The current value is 8.00 GB. At the bottom, there are two buttons: 'Create' and 'Cancel'.

Create Virtual Hard Disk

File location and size

Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

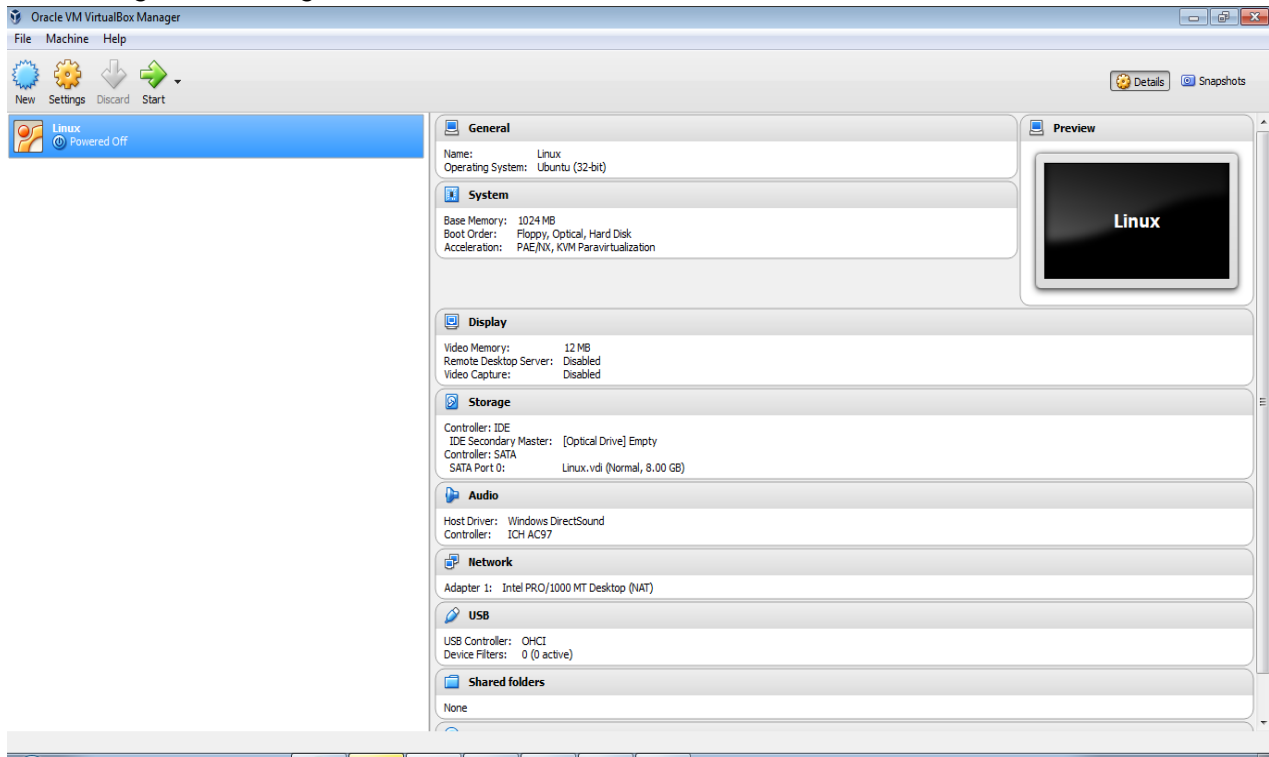


Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

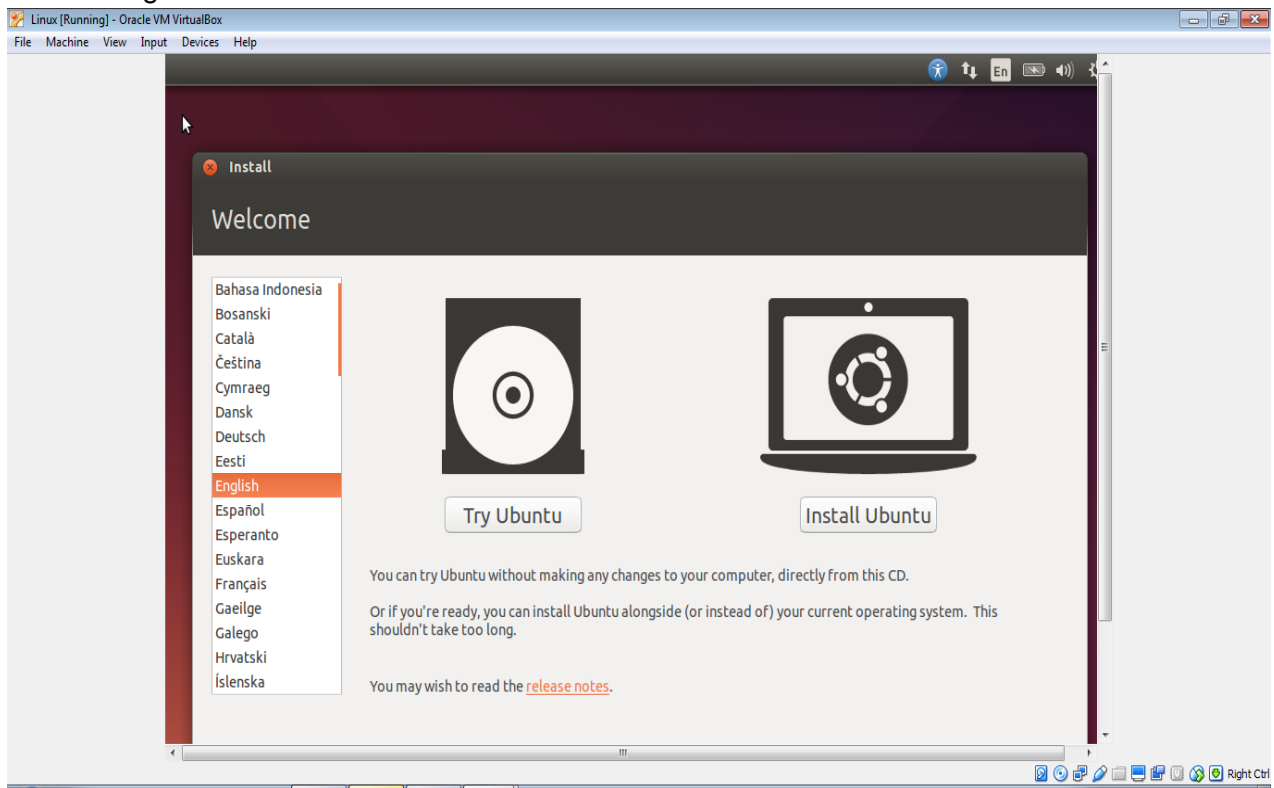
8.00 GB

4.00 MB 2.00 TB

D: Loading and Starting the LINUX Virtual Machine:



E: Installing Ubuntu:



3. Installing Wireshark inside Ubuntu

A: Using the command “sudo add-apt-repository ppa:pi-rho/security”:

```
akash@akash-Dell-System-XPS-L502X: ~  
akash@akash-Dell-System-XPS-L502X:~$ sudo add-apt-repository ppa:pi-rho/security  
[sudo] password for akash:  
A place for security-related packages  
  
Terminal Objective #1: Make current builds available to the latest LTS and the latest few normal releases  
Terminal Objective #2: Make wilder packages behave in a system-installed way  
Terminal Objective #3: Maximize the use of the toolchain's hardening capabilities for all packages  
Terminal Objective #4: Make packages debian-friendly without being too debiany  
More info: https://launchpad.net/~pi-rho/+archive/ubuntu/security  
Press [ENTER] to continue or ctrl-c to cancel adding it  
  
gpg: keyring '/tmp/tmp7qy5zlog/seccring.gpg' created  
gpg: keyring '/tmp/tmp7qy5zlog/pubring.gpg' created  
gpg: requesting key 779C27D7 from hkp server keyserver.ubuntu.com  
gpg: /tmp/tmp7qy5zlog/trustdb.gpg: trustdb created  
gpg: key 779C27D7: public key "Launchpad PPA for pi-rho" imported  
gpg: Total number processed: 1  
gpg:         imported: 1 (RSA: 1)  
OK  
akash@akash-Dell-System-XPS-L502X:~$
```

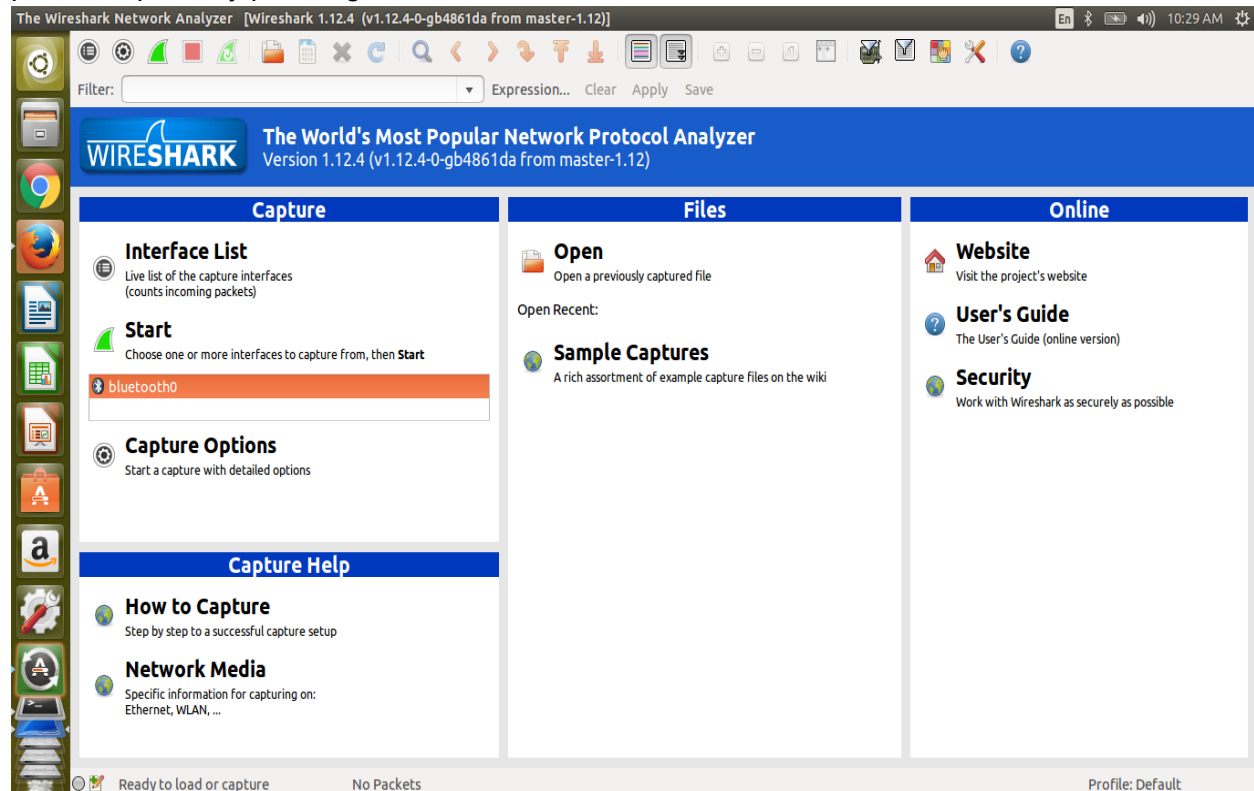
B: Using the command “sudo apt-get update”:

```
akash@akash-Dell-System-XPS-L502X: ~  
akash@akash-Dell-System-XPS-L502X:~$ sudo apt-get update  
Ign http://dl.google.com stable InRelease  
Get:1 http://dl.google.com stable Release.gpg [181 B]  
Ign http://us.archive.ubuntu.com trusty InRelease  
Get:2 http://dl.google.com stable Release [1,345 B]  
Get:3 http://dl.google.com stable/main amd64 Packages [1,213 B]  
Get:4 http://us.archive.ubuntu.com trusty-updates InRelease [65.9 kB]  
Get:5 http://security.ubuntu.com trusty-security InRelease [65.9 kB]  
Hit http://ppa.launchpad.net trusty InRelease  
Ign http://extras.ubuntu.com trusty InRelease  
Get:6 http://dl.google.com stable/main i386 Packages [795 B]  
Get:7 http://extras.ubuntu.com trusty Release.gpg [72 B]  
Hit http://ppa.launchpad.net trusty/main amd64 Packages  
Get:8 http://us.archive.ubuntu.com trusty-backports InRelease [65.9 kB]  
Hit http://extras.ubuntu.com trusty Release  
Hit http://ppa.launchpad.net trusty/main i386 Packages  
Hit http://us.archive.ubuntu.com trusty Release.gpg  
Hit http://ppa.launchpad.net trusty/main Translation-en  
Get:9 http://us.archive.ubuntu.com trusty-updates/main Sources [260 kB]  
Hit http://extras.ubuntu.com trusty/main Sources  
Ign http://dl.google.com stable/main Translation-en_US  
Get:10 http://security.ubuntu.com trusty-security/main Sources [105 kB]  
Ign http://dl.google.com stable/main Translation-en  
Hit http://extras.ubuntu.com trusty/main amd64 Packages  
Get:11 http://us.archive.ubuntu.com trusty-updates/restricted Sources [5,352 B]  
Hit http://extras.ubuntu.com trusty/main i386 Packages  
Get:12 http://us.archive.ubuntu.com trusty-updates/universe Sources [150 kB]  
Get:13 http://security.ubuntu.com trusty-security/restricted Sources [4,035 B]  
Get:14 http://security.ubuntu.com trusty-security/universe Sources [33.0 kB]  
Get:15 http://security.ubuntu.com trusty-security/multiverse Sources [2,767 B]  
Get:16 http://us.archive.ubuntu.com trusty-updates/multiverse Sources [5,547 B]  
Get:17 http://security.ubuntu.com trusty-security/main amd64 Packages [428 kB]  
Get:18 http://us.archive.ubuntu.com trusty-updates/main amd64 Packages [710 kB]  
Ign http://extras.ubuntu.com trusty/main Translation-en_US  
Ign http://extras.ubuntu.com trusty/main Translation-en  
Get:19 http://security.ubuntu.com trusty-security/restricted amd64 Packages [13.0 kB]  
Get:20 http://us.archive.ubuntu.com trusty-updates/restricted amd64 Packages [15.9 kB]  
Get:21 http://security.ubuntu.com trusty-security/universe amd64 Packages [124 kB]  
Get:22 http://us.archive.ubuntu.com trusty-updates/universe amd64 Packages [338 kB]  
Get:23 http://us.archive.ubuntu.com trusty-updates/multiverse amd64 Packages [13.2 kB]  
Get:24 http://us.archive.ubuntu.com trusty-updates/main i386 Packages [689 kB]
```

C: Using the command “sudo apt-get install wireshark”:

[illegible]

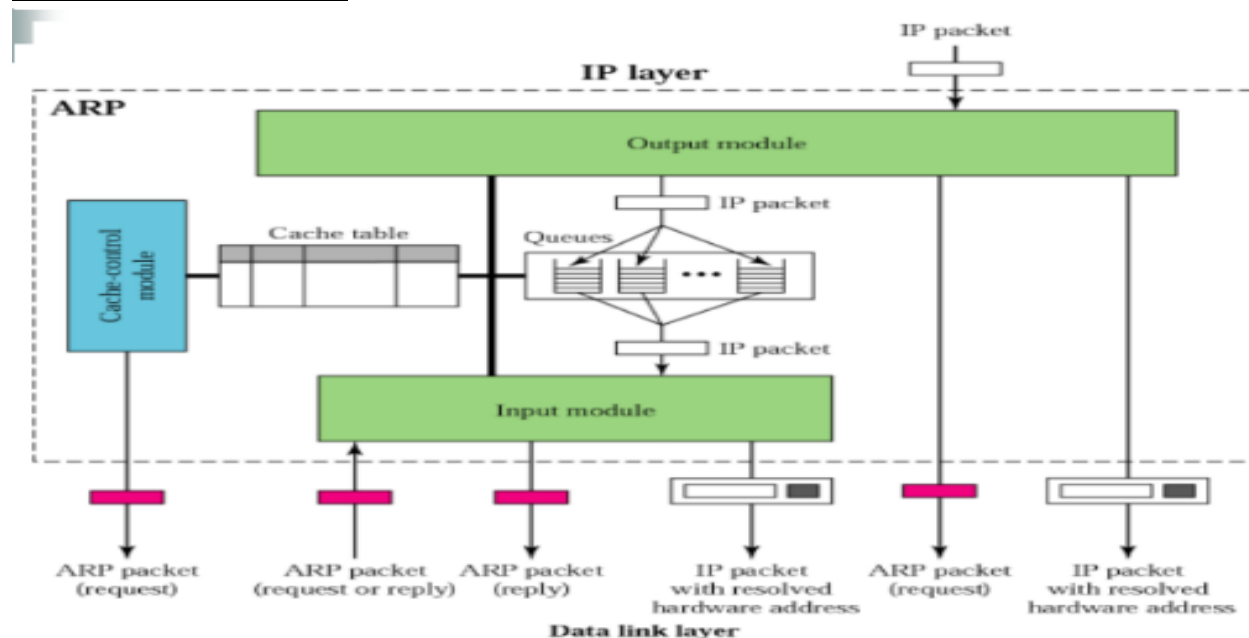
D: Wireshark getting successfully installed. By selecting the network connection we can start the packet capture by pressing the “Start” button:



WHAT IS ARP & HOW DOES IT WORK?

The Address Resolution Protocol(ARP) is a telecommunication protocol used for resolution of network layer addresses into link layer addresses, a critical function in multiple-access networks. ARP is used for mapping a network address (e.g. an IPv4 address) to a physical address like an Ethernet address (also named a MAC address). When a packet destined for a host machine on a particular local area network arrives at a gateway, the gateway assigns the ARP program to find a physical host or MAC address that matches the IP address. The ARP program first looks into the ARP cache for the address, if address is present then the packet is converted to the right packet length and format and sent to the machine. If no entry is found for the IP address, ARP broadcasts a request packet to all the machines on the LAN to see if any machine knows the IP address associated to it. A machine which identifies the IP address as its own will reply. ARP first updates the ARP cache for future reference and then sends the packet to the MAC address that replied.

COMPONENTS OF ARP



ARP HAS THE FOLLOWING COMPONENTS:

1. Cache Control Module
2. Cache Table
3. Queues
4. Input Module
5. Output Module

1. Cache Control Module –

The cache control module checks the cache table periodically(almost every 5 seconds) and updates the fields of the table.

Jitter Timer:

Change in packet transfer time:

- For some packets, transfer may be preceded by ARP request and response. If the hardware address is available in the Cache Table, the packet will be directly transferred.
- Timeout can become zero at any time, after which the ARP request and response may be required.
- To avoid jitter: Revalidation Timer is used.

Revalidation Timer:

On expiry of Revalidation Timer, CCM checks whether the hardware address has been used recently. If so, when the RT times out, a new Request is sent out (even when no new packet for outputting is there.). If Timeout becomes 0, while we are waiting for a response to the request, the row is not cleared unless the max no of attempts have been tried.

2. Cache Table -

The fields in the Cache Table are as follows:

- State : resolved , pending, free
- Queue number
- Number of attempts*
- Time out.
- IP address
- Hardware address
- Last use: set to 0 initially and made equal to time when the row is used
- Revalidation timer : made equal to 50 to 75% of timeout.
- Hardware type address
- Protocol type
- HA length in bytes
- PA length in bytes
- Interface number (multi-homed host routers)

3. Queues -

One for each destination.

4. Input Module -

- Extract the sender's address and if the cache contains the item, update it.
- Examine operation
 - If ARP request –
Compare target IP address with the local IP address.
 - If identical –
 - Reverse the sender & target binding.
 - Insert the hardware address.
 - Change Operation field to 2.
 - Send the response.
 - If not identical STOP.

- If ARP Response –
 - The system may have been waiting for it.
- If the state is Pending
 - Update the entry.
 - While the Q is not empty DeQ one packet, send the packet.
- If the state is Resolved
 - Update the entry.

5. Output Module -

(Input to the Module: from IP layer)

STEPS: Check the cache.

- If found:
 - If State is resolved extract HA and Send.
 - If state is pending enqueue the packet.
- If not found
 - Create a Q.
 - Enqueue the Packet.
 - Create a cache entry with STATE Pending, ATTEMPTS one
 - Send an ARP Request.

Cache control – Sends ARP requests, if more than one Attempt is required.

ARP MESSAGE FORMAT

Address resolution using ARP is accomplished through the exchange of messages between the source device seeking to perform the resolution, and the destination device that responds to it. As with other protocols, a special message format is used containing the information required for each step of the resolution process. ARP messages use a relatively simple format. It includes a field describing the type of message (its operational code or opcode) and information on both layer two and layer three addresses. In order to support addresses that may be of varying length, the format specifies the type of protocol used at both layer two and layer three and the length of addresses used at each of these layers. It then includes space for all four of the address combinations.

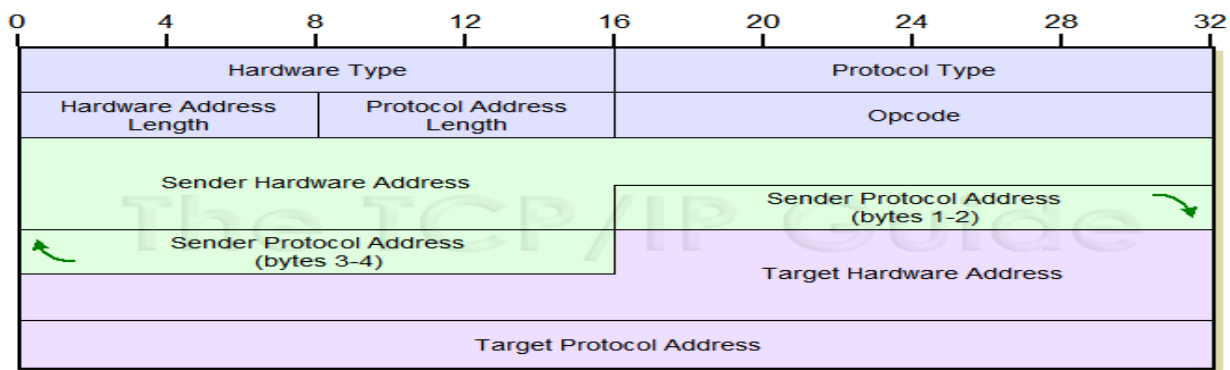


Figure 49: Address Resolution Protocol (ARP) Message Format

PACKET CAPTURE & PAYLOAD FORMAT

In Wireshark, every field represents many different pieces of information regarding the network and the data. The software is used to capture all the network traffic that runs through the user's machine in the form of packets. Once we select a particular protocol as mentioned in the captured data packets list, we can see a detailed display of the data information in the detail section.

Payload is one of the important pieces of data that is being carried in a packet. It is the area where higher-level PDUs such as IP datagrams are placed. Traditionally, the payload area consists of around 1500 bytes representing the MTU for Ethernet. In some cases, payload may be padded with 0 bytes to ensure that the overall frame meets the minimum length requirements.

Payload format has several fields such as:

1. Hardware Type: The type of MAC address being in use.
2. Protocol Type: The Layer 3 protocol that is in use.
3. Hardware Size: It is the length of the MAC address.
4. Protocol Size: The length of the address of the protocol in use.
5. Opcode: The type of ARP message displayed.

It also has a header that identifies the source and the destination of a packet and the actual data is the payload.

There are also fields continued:

Hardware Type:

- i) Sender MAC address: The MAC address of the machine sending the request to send packets
- ii) Sender IP address: The protocol address of the machine sending the ARP request
- iii) Target MAC address: The MAC address that is being used
- iv) Target IP address: The protocol address of the destination where the packets need to be sent.

PACKET HEADER

It shows the details about the packet selected in the packet-listing window. In order to select a packet, place the cursor over the packet's one-line summary in the packet-listing window. These details will include the Ethernet frame and IP datagram that consists of this packet data.

The amount of IP-layer and Ethernet displayed can be extended or minimized by clicking on the right-pointing or down-pointing arrowhead to the left of the Ethernet frame or IP diagram line in the packet details window.

The packet header consists of the Ethernet II source and destination broadcast details. It takes care of the information about where the packets are originated from and where they are transmitted. The order of the data packets is also maintained and displayed. The additional details include what type of protocol is used and the IP address of them.

Lab Overview:

The following scenarios are analyzed on Address Resolution Protocol (ARP) using Wireshark and Linux terminal.

- Address resolution between devices on same LAN.
- Commands on ARP Cache table
- Concept of Gratuitous in ARP
- Attacks involving ARP

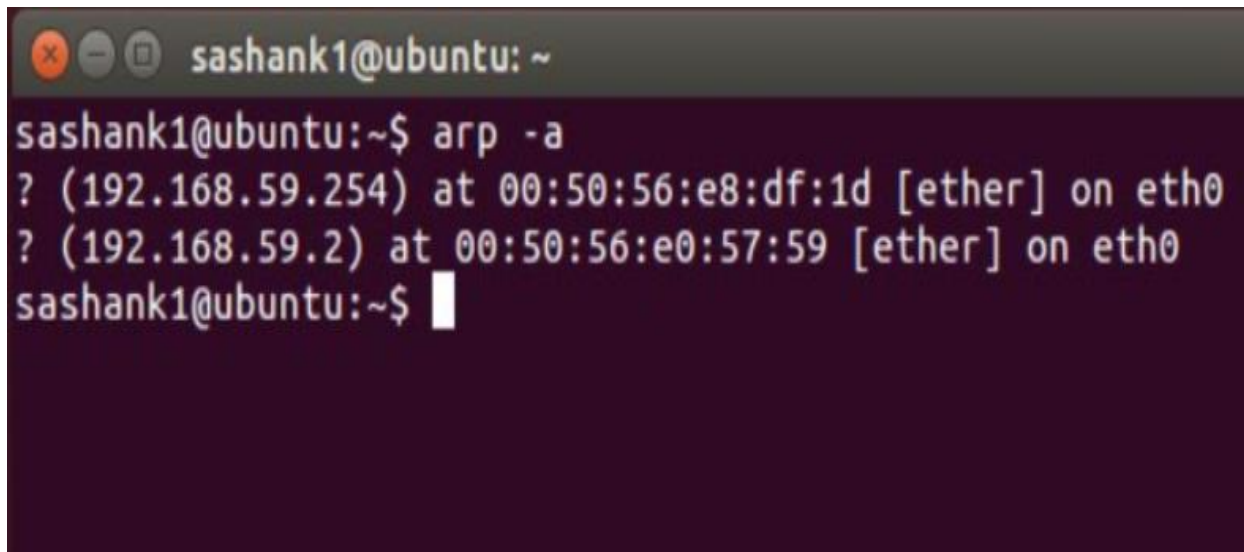
Address resolution between devices on same LAN

Consider following two Linux configuration machines with ipconfig 192.168.59.131 and 192.168.59.129 respectively.

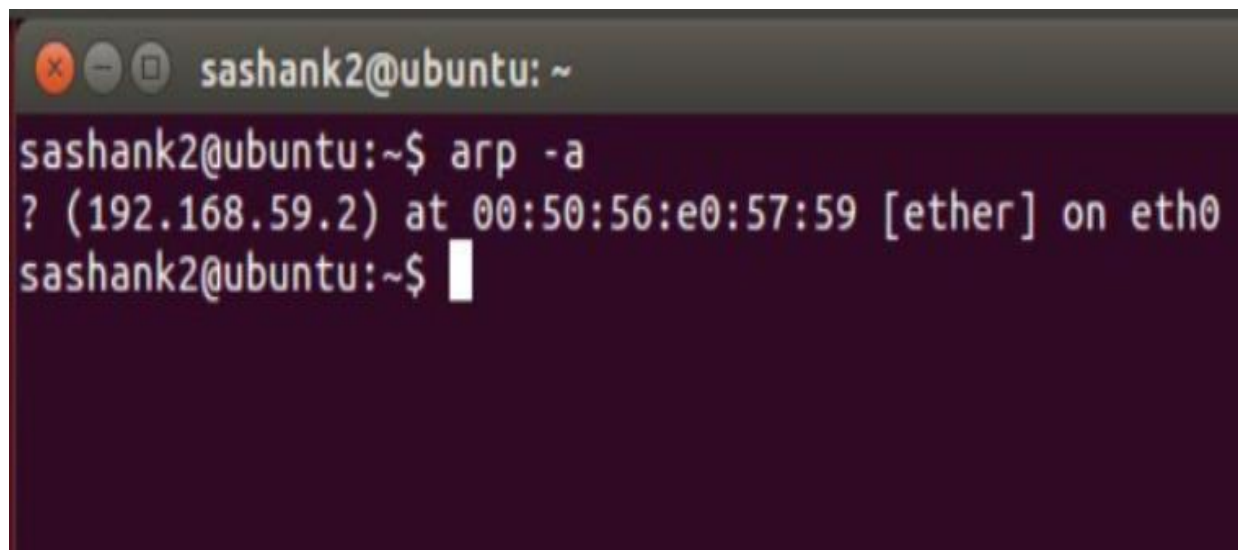
```
sashank1@ubuntu: ~  
sashank1@ubuntu:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 00:0c:29:e8:b0:72  
          inet addr:192.168.59.131  Bcast:192.168.59.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fee8:b072/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:15209 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:3245 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:19209117 (19.2 MB)  TX bytes:290068 (290.0 KB)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:420 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:420 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:39268 (39.2 KB)  TX bytes:39268 (39.2 KB)  
  
sashank1@ubuntu:~$
```

```
sashank2@ubuntu: ~  
sashank2@ubuntu:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 00:0c:29:0b:61:49  
          inet addr:192.168.59.129  Bcast:192.168.59.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fe0b:6149/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:16421 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:4304 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:20570033 (20.5 MB)  TX bytes:396322 (396.3 KB)  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:678 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:678 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:64140 (64.1 KB)  TX bytes:64140 (64.1 KB)  
  
sashank2@ubuntu:~$
```

Contents of their initial ARP table are as follows:



```
sashank1@ubuntu: ~  
sashank1@ubuntu:~$ arp -a  
? (192.168.59.254) at 00:50:56:e8:df:1d [ether] on eth0  
? (192.168.59.2) at 00:50:56:e0:57:59 [ether] on eth0  
sashank1@ubuntu:~$
```



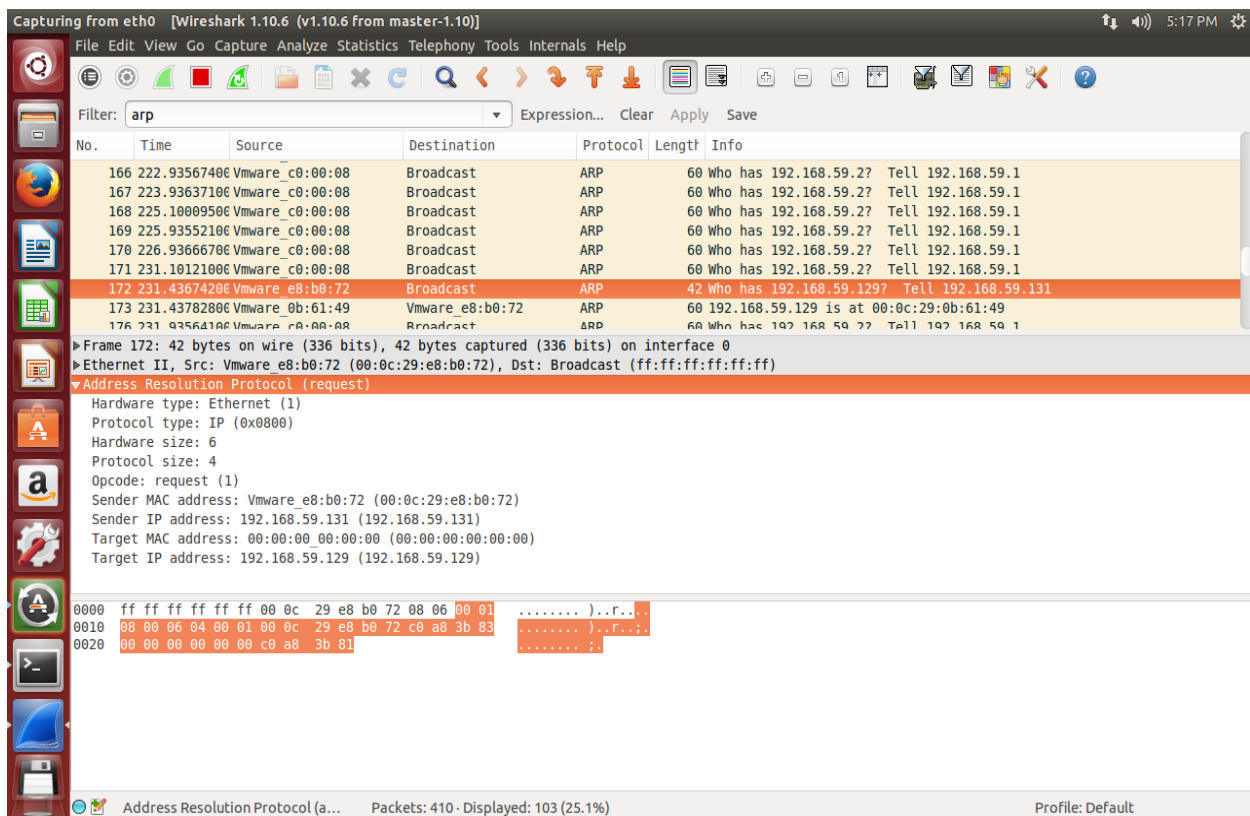
```
sashank2@ubuntu: ~  
sashank2@ubuntu:~$ arp -a  
? (192.168.59.2) at 00:50:56:e0:57:59 [ether] on eth0  
sashank2@ubuntu:~$
```

Ping is used to start the Address resolution and add corresponding MAC address in to ARP cache table.

For address resolution purpose host 2 (192.168.59.129) is pinged from host 1(192.168.59.131). Pinging is done with the “ping” command followed by the IP address of the target.


```
sashank1@ubuntu: ~
sashank1@ubuntu:~$ ping 192.168.59.129
PING 192.168.59.129 (192.168.59.129) 56(84) bytes of data.
64 bytes from 192.168.59.129: icmp_seq=1 ttl=64 time=3.01 ms
64 bytes from 192.168.59.129: icmp_seq=2 ttl=64 time=0.622 ms
64 bytes from 192.168.59.129: icmp_seq=3 ttl=64 time=0.584 ms
64 bytes from 192.168.59.129: icmp_seq=4 ttl=64 time=0.614 ms
64 bytes from 192.168.59.129: icmp_seq=5 ttl=64 time=0.274 ms
64 bytes from 192.168.59.129: icmp_seq=6 ttl=64 time=0.286 ms
64 bytes from 192.168.59.129: icmp_seq=7 ttl=64 time=0.532 ms
64 bytes from 192.168.59.129: icmp_seq=8 ttl=64 time=0.572 ms
^C
--- 192.168.59.129 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 6999ms
rtt min/avg/max/mdev = 0.274/0.812/3.018/0.844 ms
sashank1@ubuntu:~$
```

When ping is done the address resolution can be observed from the packets captured from Wireshark. The following are wire shark packets for both request and reply.



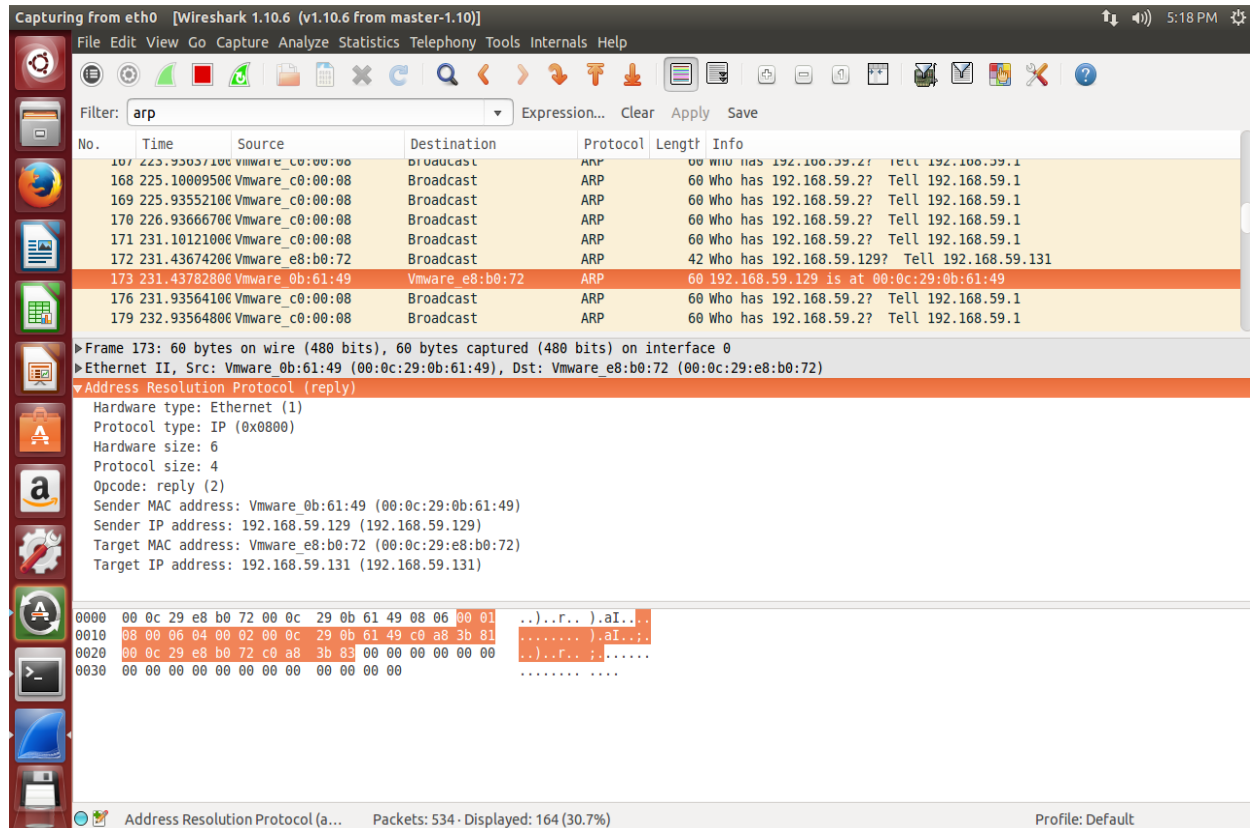
The above figure shows the ARP packet header and payload. We can observe that it contains Sender IP address, Sender MAC address, Target IP address and Type field. The most important information to be observed that it doesn't contain information on Target MAC address.

We can also observe following fields with corresponding values

Opcode: 1

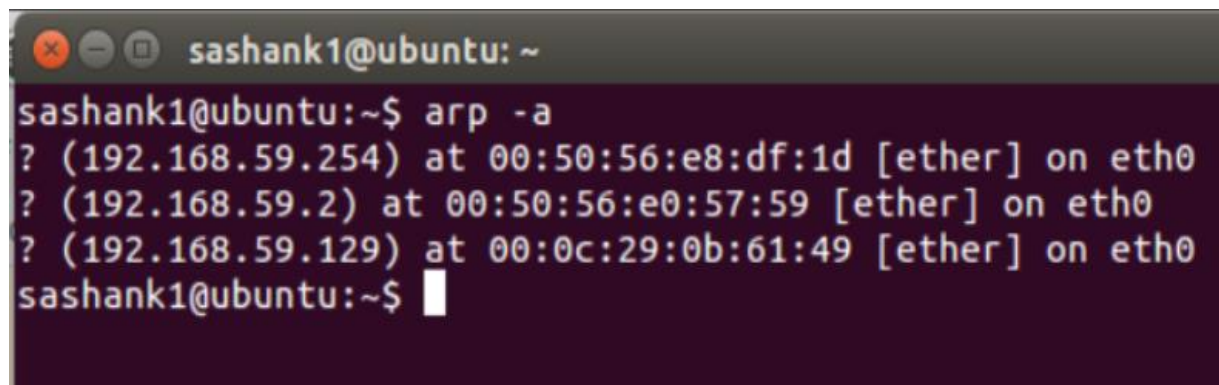
Broadcast Address: ff:ff:ff:ff:ff:ff

Target Hardware Address: 0.



The above figure shows the ARP reply packet on Wireshark. In the reply packet, the Target MAC address is resolved from the request packet. The sender MAC address in the reply packet constitutes the target MAC address of the request packet.

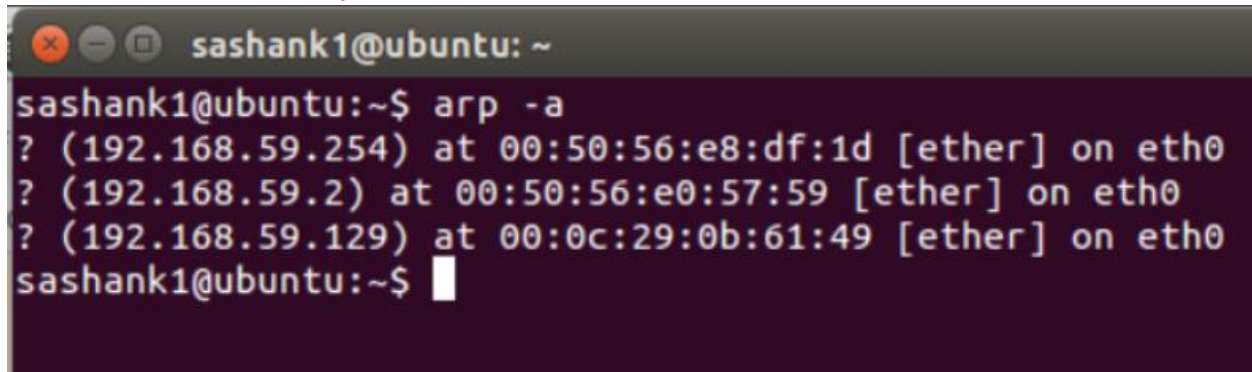
The following figure show the updated table of ARP Cache with the target host MAC address after address resolution (192.168.59.129 is included into ARP table).



Commands on ARP Cache table

Some of the ARP commands can be observed from following screen shots

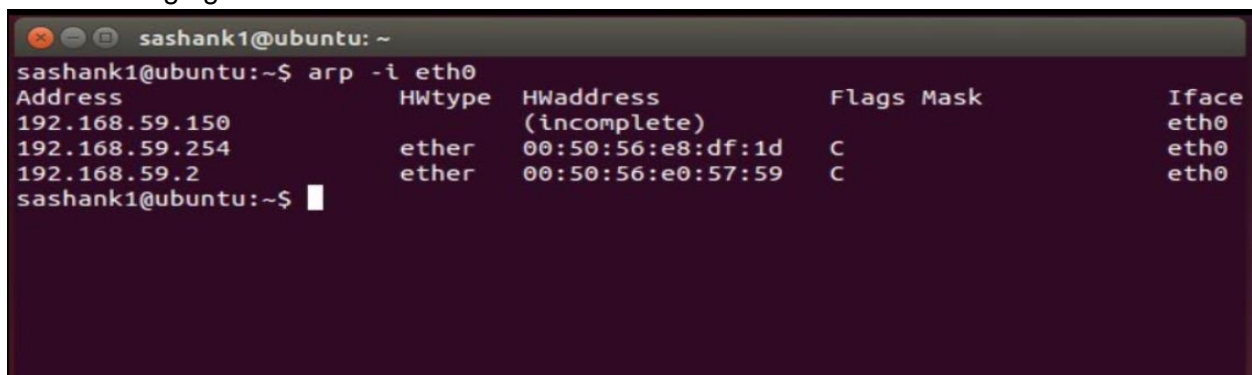
1. “arp -a” command displays ARP cache table



```
sashank1@ubuntu: ~  
sashank1@ubuntu:~$ arp -a  
? (192.168.59.254) at 00:50:56:e8:df:1d [ether] on eth0  
? (192.168.59.2) at 00:50:56:e0:57:59 [ether] on eth0  
? (192.168.59.129) at 00:0c:29:0b:61:49 [ether] on eth0  
sashank1@ubuntu:~$
```

2. Devices connected to a specific interface can be retrieved using
arp -i <Interface name> -d <IP Address> for eg. “arp -I eth0”

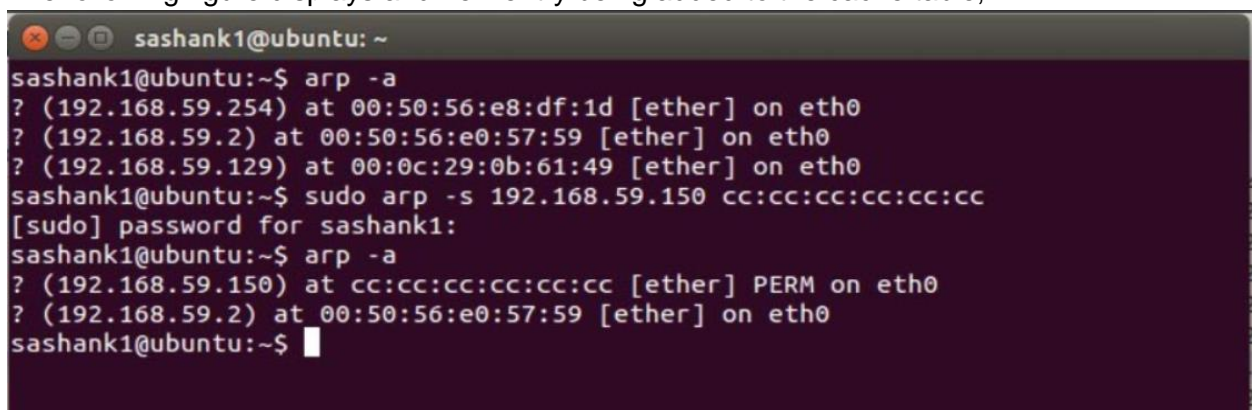
The following figure shows how interface command is executed.



```
sashank1@ubuntu: ~  
sashank1@ubuntu:~$ arp -i eth0  
Address HWtype HWaddress Flags Mask Iface  
192.168.59.150 (incomplete) eth0  
192.168.59.254 ether 00:50:56:e8:df:1d C eth0  
192.168.59.2 ether 00:50:56:e0:57:59 C eth0  
sashank1@ubuntu:~$
```

3. A new entry can be added to the existing ARP cache table using following command
arp -i <Interface name> -s <IP Address> <MAC Address>

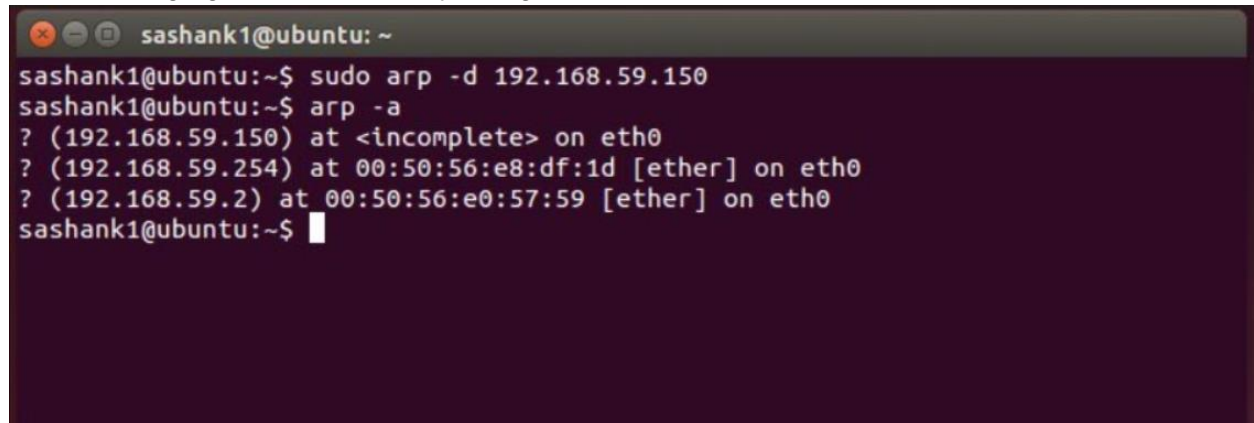
The following figure displays and new entry being added to the cache table,



```
sashank1@ubuntu: ~  
sashank1@ubuntu:~$ arp -a  
? (192.168.59.254) at 00:50:56:e8:df:1d [ether] on eth0  
? (192.168.59.2) at 00:50:56:e0:57:59 [ether] on eth0  
? (192.168.59.129) at 00:0c:29:0b:61:49 [ether] on eth0  
sashank1@ubuntu:~$ sudo arp -s 192.168.59.150 cc:cc:cc:cc:cc:cc  
[sudo] password for sashank1:  
sashank1@ubuntu:~$ arp -a  
? (192.168.59.150) at cc:cc:cc:cc:cc:cc [ether] PERM on eth0  
? (192.168.59.2) at 00:50:56:e0:57:59 [ether] on eth0  
sashank1@ubuntu:~$
```

- Any entry can be deleted from the cache table using following command
`arp -i <Interface name> -d <IP Address>`

The following figure show an entry being deleted from ARP cache table



```
sashank1@ubuntu: ~  
sashank1@ubuntu:~$ sudo arp -d 192.168.59.150  
sashank1@ubuntu:~$ arp -a  
? (192.168.59.150) at <incomplete> on eth0  
? (192.168.59.254) at 00:50:56:e8:df:1d [ether] on eth0  
? (192.168.59.2) at 00:50:56:e0:57:59 [ether] on eth0  
sashank1@ubuntu:~$
```

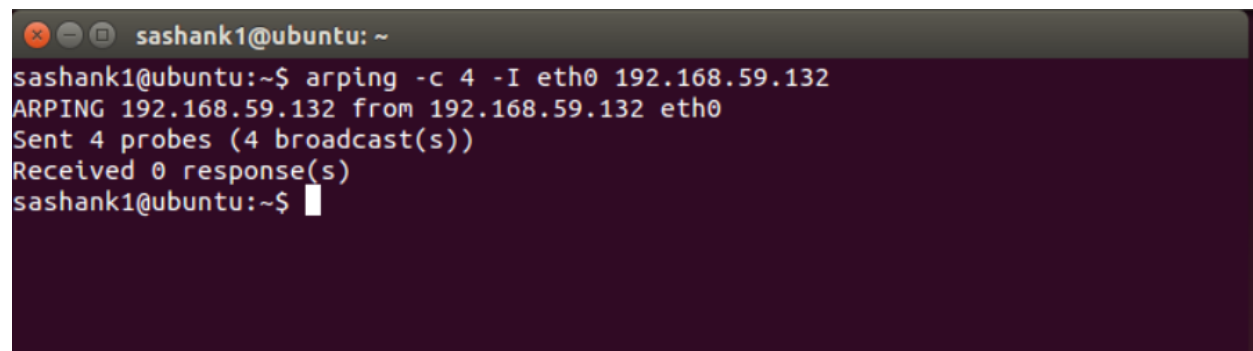
Concept of Gratuitous in ARP

Gratuitous of ARP occurs when a host sends an ARP request looking for its *own* address. This is usually done when the interface is configured “up” at bootstrap time.

This makes sure that no other host is using same IP address. Hence, the host sending the gratuitous ARP request will not be expecting a reply.

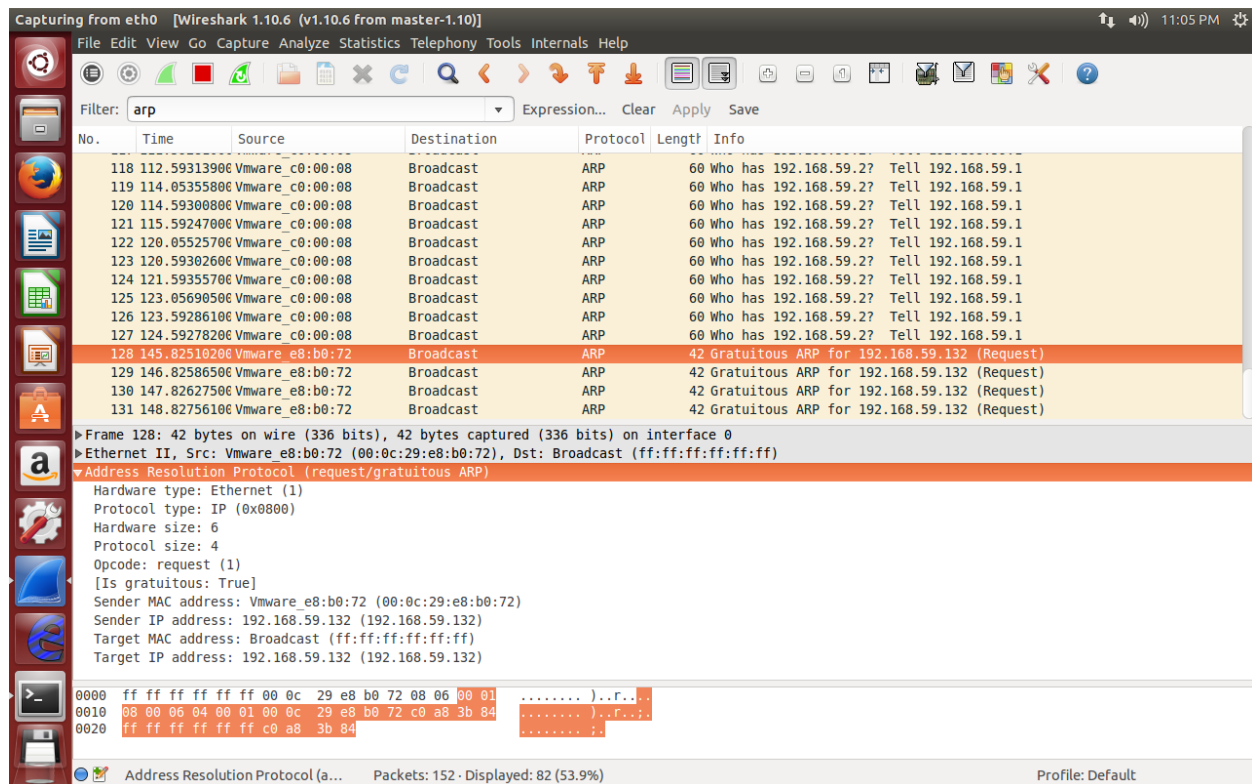
This also makes sure all the devices in the network be updated with the sender hosts physical address in their ARP cache table.

The following figure displays and gratuitous request from 192.168.59.131



```
sashank1@ubuntu: ~  
sashank1@ubuntu:~$ arping -c 4 -I eth0 192.168.59.132  
ARPING 192.168.59.132 from 192.168.59.132 eth0  
Sent 4 probes (4 broadcast(s))  
Received 0 response(s)  
sashank1@ubuntu:~$
```

Following is the captured request packet for above gratuitous request



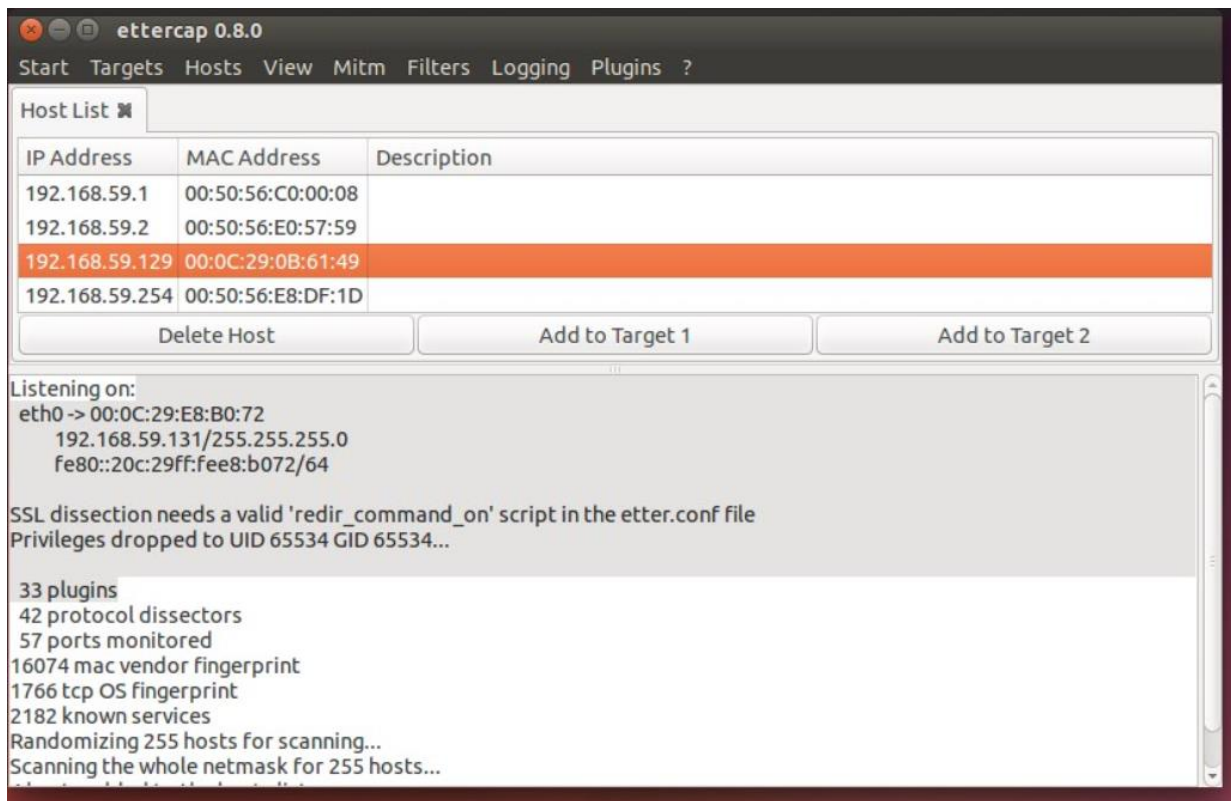
Attacks involving ARP

There have been series of attacks involving ARP.

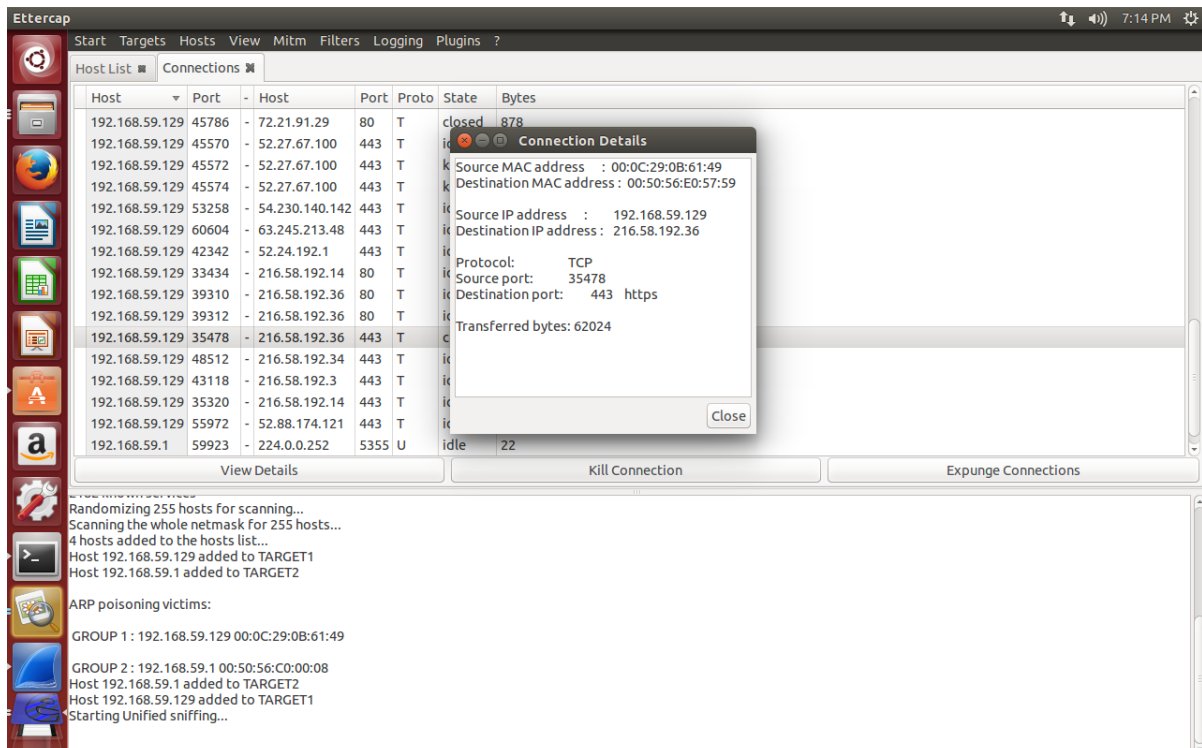
ARP spoofing is one of the technique by which the attacker sends spoofed ARP messages on to LAN. Generally, Attacker aims at another host in the LAN and associate his MAC address as default gateway, causing any traffic being redirected to attacker.

The following screen shot from Ettercap showing how a host at IP 192.168.59.129 is being spoofed

Consider 192.168.59.129 is a host in a network and its corresponding gateway is 192.168.59.1. An attacker host 192.168.59.31 is targeting the host 192.168.59.129.



The below screen show displays how the data of 192.168.59.129 is being redirected through the attacker IP address. 216.58.192.36 is IP address request from the victim.



Learning Outcomes

ARP is used for attaining physical address from the network address.

For the actual data transfer MAC address plays a very important role. The source first checks its ARP cache for the corresponding destination MAC address. If the record is not present in cache, the system generates an ARP request message which is broadcasted through the LAN. The device with the corresponding IP address responds with an ARP reply message with its MAC address to the source after updating its ARP cache. Once reply message is received, it is processed by the source and ARP cache table is updated.

Now source and destination devices have the corresponding addresses to communicate between each other over the LAN.

ARP is a reply and request protocol. The packet structure of ARP uses a simple message format. And the packet structure has been studied.

There are two types of messages request and reply. Request is a broadcasted message while reply message is sent from only one host in a network. The target IP address of the request message is the same as the source IP address of the response message.

The ARP cache reduces a lot of load from ARP translation process, by storing the known addresses for certain amount of time.

The ARP cache table can be viewed and updated using ARP commands.

A host can send itself an ARP request message for the purpose of checking any duplicate IPs existing in network or for updating its old Physical address in other hosts ARP cache.

In LAN connection, handshake mechanism is utilized in packet exchange.

It is vulnerable to attacks like in ARP spoofing where middle man attack is possible.

Conclusion

ARP is a protocol existing between Ethernet and IP. It converts network address to a physical address. Hence ARP is used to find the Ethernet address that corresponds to the local IP address.

In this lab, we have observed and studied characteristics of ARP using Wireshark software.