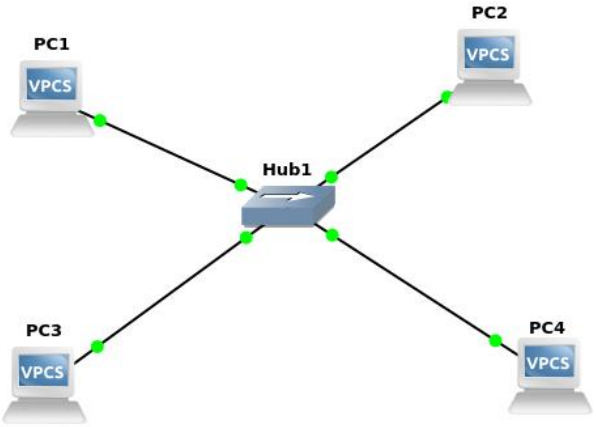
**Pratham U-210905407**

# Computer Networks Lab 5:Computer Network Design using HUB in GNS3

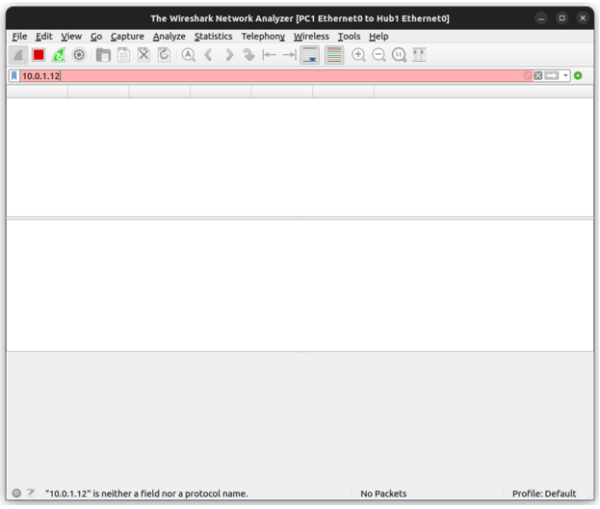
**1)Design network configuration shown in Figure 5.29 for all parts. Connect all four VMs to a single Ethernet segment via a single hub as shown in Figure 5.29. Configure the IP addresses for the PCs as shown in Table 6.1.**



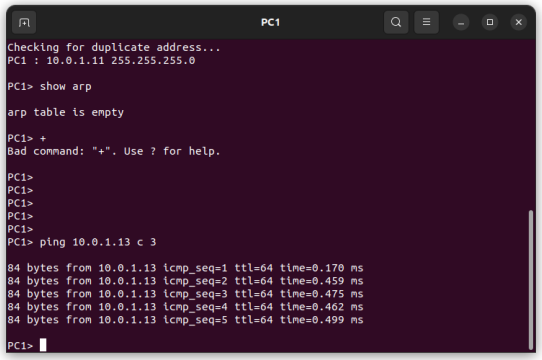

**a. On PC1, view the ARP cache with show arp**
**b. Start Wire shark on PC1-Hub1 link with a capture filter set to the IP address of PC2.**
**c. Issue a ping command from PC1 to PC2:**
**PC1% ping 10.0.1.13 –c 3**
**a)**

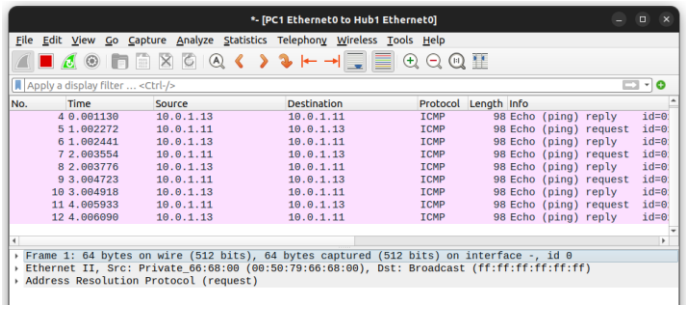
```
PC1> show arp

00:50:79:66:68:01  10.1.1.12 expires in 116 seconds
```

**b)**



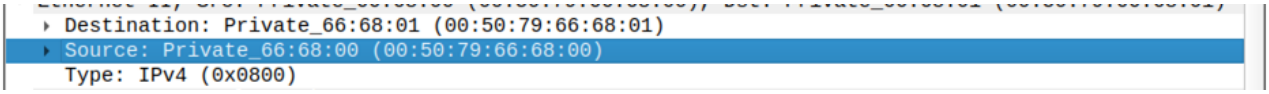
**c)**



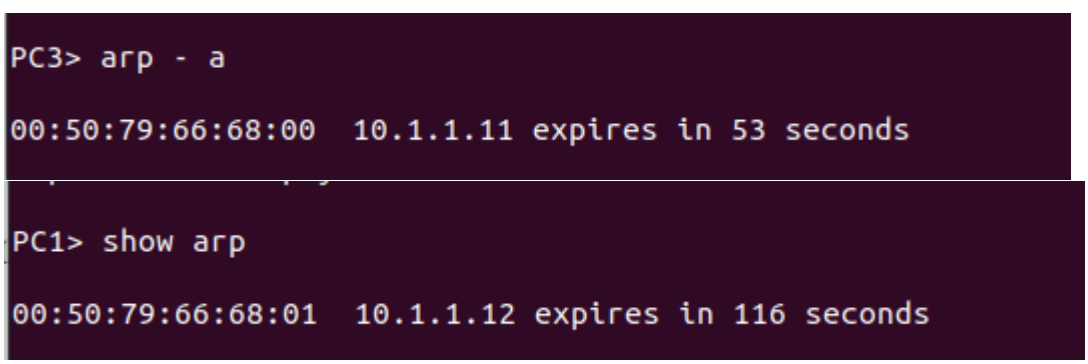




**Observe the ARP packets in the Wireshark window. Explore the MAC addresses in the Ethernet headers of the captured packets.**
**Direct our attention to the following fields:**

- **The destination MAC address of the ARP Request packets.**
- **The Type Field in the Ethernet headers of ARP packets.**




**d. View the ARP cache again with the command arp -a. Note that ARP cache entries can get refreshed/deleted fairly quickly (~2 minutes).**
**show arp**
**e. Save the results of Wireshark.**



**2. To observe the effects of having more than one host with the same (duplicate) IP address in a network.**
**After completing Exercise 1, the IP addresses of the Ethernet interfaces on the four PCs are as shown in Table 6.2 below. Note that PC1 and PC4 are assigned the same IP address.**

```
PC4> ip 10.0.1.11
Checking for duplicate address...
10.0.1.11 is being used by MAC 00:50:79:66:68:00
Address not changed

PC4>
```

**a. Delete all entries in the ARP cache on all PCs.**
**b. Run Wire shark on PC3-Hub1 link and capture the network traffic to and from the duplicate IP address 10.0.1.11.**

| | | | | | |
|---|---|---|---|---|---|
| 13 | 68.831999 | Private_66:68:00 | Broadcast | ARP | 64 Gratuitous ARP for 10.0.1.11 (Request) (duplicate use of 10.0.1.11 detected!) |
| 14 | 69.832718 | Private_66:68:00 | Broadcast | ARP | 64 Gratuitous ARP for 10.0.1.11 (Request) (duplicate use of 10.0.1.11 detected!) |
| 15 | 70.833247 | Private_66:68:00 | Broadcast | ARP | 64 Gratuitous ARP for 10.0.1.11 (Request) (duplicate use of 10.0.1.11 detected!) |
| 16 | 79.043937 | Private_66:68:03 | Broadcast | ARP | 64 Gratuitous ARP for 10.0.1.14 (Request) |
| 17 | 80.044972 | Private_66:68:03 | Broadcast | ARP | 64 Gratuitous ARP for 10.0.1.14 (Request) |

**c. From PC3, issue a ping command to the duplicate IP address, 10.0.1.11, by typing**
**PC3% ping 10.0.1.11 –c 5**

| | | | | | |
|---|---|---|---|---|---|
| 30 | 107.416865 | 10.0.1.11 | 10.0.1.13 | ICMP | 98 Echo (ping) reply id=0x2e6f, seq=5/1280, ttl=64 (request in 29) |
| 31 | 177.911895 | 10.0.1.13 | 10.0.1.11 | ICMP | 98 Echo (ping) request id=0x756f, seq=1/256, ttl=64 (reply in 32) |
| 32 | 177.912119 | 10.0.1.11 | 10.0.1.13 | ICMP | 98 Echo (ping) reply id=0x756f, seq=1/256, ttl=64 (request in 31) |
| 33 | 178.913059 | 10.0.1.13 | 10.0.1.11 | ICMP | 98 Echo (ping) request id=0x766f, seq=2/512, ttl=64 (reply in 34) |
| 34 | 178.913200 | 10.0.1.11 | 10.0.1.13 | ICMP | 98 Echo (ping) reply id=0x766f, seq=2/512, ttl=64 (request in 33) |
| 35 | 179.914207 | 10.0.1.13 | 10.0.1.11 | ICMP | 98 Echo (ping) request id=0x776f, seq=3/768, ttl=64 (reply in 36) |
| 36 | 179.914391 | 10.0.1.11 | 10.0.1.13 | ICMP | 98 Echo (ping) reply id=0x776f, seq=3/768, ttl=64 (request in 35) |
| 37 | 180.915366 | 10.0.1.13 | 10.0.1.11 | ICMP | 98 Echo (ping) request id=0x786f, seq=4/1024, ttl=64 (reply in 38) |
| 38 | 180.915484 | 10.0.1.11 | 10.0.1.13 | ICMP | 98 Echo (ping) reply id=0x786f, seq=4/1024, ttl=64 (request in 37) |
| 39 | 181.916613 | 10.0.1.13 | 10.0.1.11 | ICMP | 98 Echo (ping) request id=0x796f, seq=5/1280, ttl=64 (reply in 40) |

**d. Stop Wireshark, save all ARP packets and screenshot the ARP cache of PC3 using the arp –a command:**
**> PC3% arp – a**

```
PC3> arp -a

Invalid ID

PC3>
```

**3. To test the effects of changing the netmask of a network configuration.**
**a. Design the configuration as Exercise 1 and replace the hub with a switch, two hosts (PC2 and PC4) have been assigned different network prefixes.**
**Setup the interfaces of the hosts as follows:**
**VPCS IP Address of eth0 Network Mask**
**PC110.0.1.100 / 24255.255.255.0**
**PC210.0.1.101 / 28255.255.255.240**
**PC310.0.1.120 / 24255.255.255.0**
**PC410.0.1.121 / 28255.255.255.240**

>
PC1> ip 10.0.1.100 255.255.255.0
Checking for duplicate address...
PC1 : 10.0.1.100 255.255.255.0

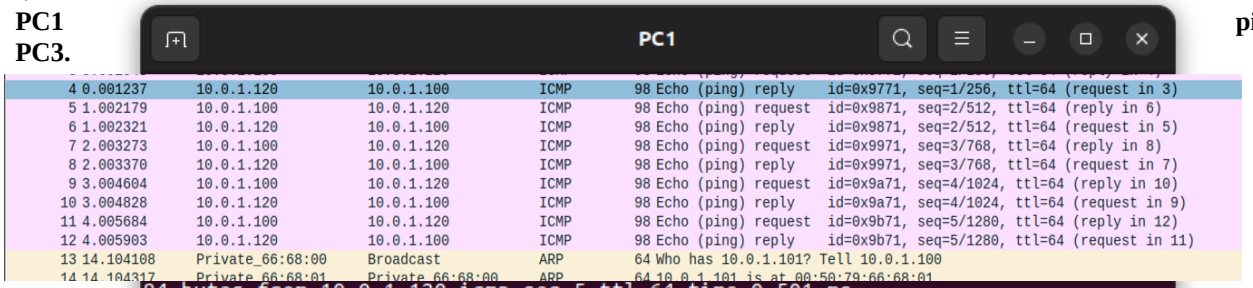
PC2> ip 10.0.1.101 255.255.255.240
Checking for duplicate address...
PC2 : 10.0.1.101 255.255.255.240

PC3> ip 10.0.1.120 255.255.255.0
Checking for duplicate address...
PC3 : 10.0.1.120 255.255.255.0

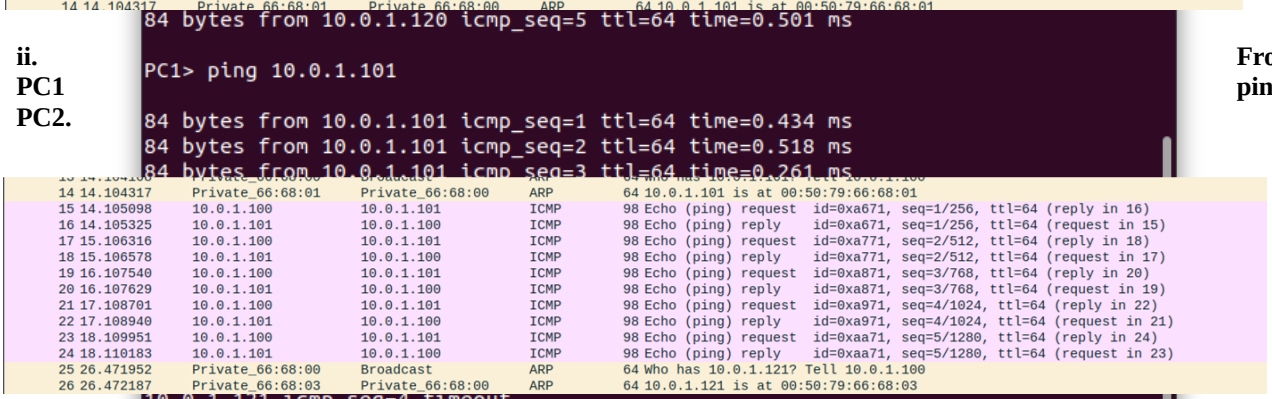
PC4> ip 10.0.1.121 255.255.255.240
Checking for duplicate address...
PC4 : 10.0.1.121 255.255.255.240

**b. Run Wireshark on PC1-Hub1 link and capture the packets for the following scenarios**
**i. From PC1 ping PC3.**
**ii. From PC1 ping PC2.**
**iii. From PC1 ping PC4.**

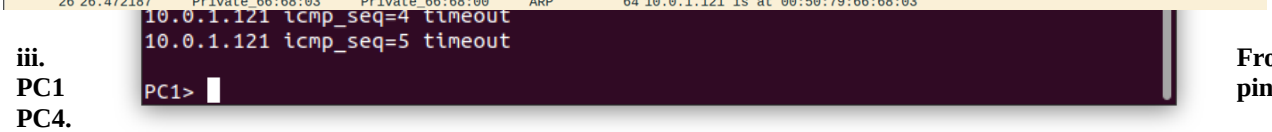
**i. From PC1 ping PC3.**




**ii. From PC1 ping PC2.**




**iii. From PC1 ping PC4.**



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 25 | 26.471952 | Private_66:68:00 | Broadcast | ARP | 64 | Who has 10.0.1.121? Tell 10.0.1.100 |
| 26 | 26.472187 | Private_66:68:03 | Private_66:68:00 | ARP | 64 | 10.0.1.121 is at 00:50:79:66:68:03 |
| 27 | 26.473007 | 10.0.1.100 | 10.0.1.121 | ICMP | 98 | Echo (ping) request id=0xb271, seq=1/256, ttl=64 (reply in 32) |
| 28 | 26.473188 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 29 | 27.473912 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 30 | 28.473348 | 10.0.1.100 | 10.0.1.121 | ICMP | 98 | Echo (ping) request id=0xb471, seq=2/512, ttl=64 (reply in 38) |
| 31 | 28.474606 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 32 | 29.474785 | 10.0.1.121 | 10.0.1.100 | ICMP | 98 | Echo (ping) reply id=0xb271, seq=1/256, ttl=64 (request in 27) |
| 33 | 29.474809 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 34 | 30.473446 | 10.0.1.100 | 10.0.1.121 | ICMP | 98 | Echo (ping) request id=0xb671, seq=3/768, ttl=64 (reply in 43) |
| 35 | 30.475345 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 36 | 31.476271 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 37 | 32.473676 | 10.0.1.100 | 10.0.1.121 | ICMP | 98 | Echo (ping) request id=0xb871, seq=4/1024, ttl=64 (reply in 47) |
| 38 | 32.477056 | 10.0.1.121 | 10.0.1.100 | ICMP | 98 | Echo (ping) reply id=0xb471, seq=2/512, ttl=64 (request in 30) |
| 39 | 32.477076 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |

**iv. From PC4 ping PC1.**

```
PC4> ping 10.0.1.100

host (255.255.255.240) not reachable

PC4>
```

Apply a display filter ... <Ctrl-/>

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 2 | 1.000864 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 3 | 2.000974 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |

**v. From PC2 ping PC4.**

PC2> ping 10.0.1.100

84 bytes from 10.0.1.100 icmp_seq=1 ttl=64 time=0.092 ms
84 bytes from 10.0.1.100 icmp_seq=2 ttl=64 time=0.278 ms
84 bytes from 10.0.1.100 icmp_seq=3 ttl=64 time=0.265 ms
84 bytes from 10.0.1.100 icmp_seq=4 ttl=64 time=0.459 ms
84 bytes from 10.0.1.100 icmp_seq=5 ttl=64 time=0.427 ms

| | | | | | | |
|---|---|---|---|---|---|---|
| 52 | 390.407914 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 53 | 391.408769 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 54 | 392.408883 | Private_66:68:03 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.121 |
| 55 | 538.551849 | Private_66:68:01 | Broadcast | ARP | 64 | Who has 10.0.1.100? Tell 10.0.1.101 |
| 56 | 538.552122 | Private_66:68:00 | Private_66:68:01 | ARP | 64 | 10.0.1.100 is at 00:50:79:66:68:00 |
| 57 | 538.552892 | 10.0.1.101 | 10.0.1.100 | ICMP | 98 | Echo (ping) request id=0xb273, seq=1/256, ttl=64 (reply in 58) |
| 58 | 538.552942 | 10.0.1.100 | 10.0.1.101 | ICMP | 98 | Echo (ping) reply id=0xb273, seq=1/256, ttl=64 (request in 57) |
| 59 | 539.554329 | 10.0.1.101 | 10.0.1.100 | ICMP | 98 | Echo (ping) request id=0xb373, seq=2/512, ttl=64 (reply in 60) |
| 60 | 539.554449 | 10.0.1.100 | 10.0.1.101 | ICMP | 98 | Echo (ping) reply id=0xb373, seq=2/512, ttl=64 (request in 59) |
| 61 | 540.555435 | 10.0.1.101 | 10.0.1.100 | ICMP | 98 | Echo (ping) request id=0xb473, seq=3/768, ttl=64 (reply in 62) |
| 62 | 540.555600 | 10.0.1.100 | 10.0.1.101 | ICMP | 98 | Echo (ping) reply id=0xb473, seq=3/768, ttl=64 (request in 61) |
| 63 | 541.556657 | 10.0.1.101 | 10.0.1.100 | ICMP | 98 | Echo (ping) request id=0xb573, seq=4/1024, ttl=64 (reply in 64) |
| 64 | 541.556862 | 10.0.1.100 | 10.0.1.101 | ICMP | 98 | Echo (ping) reply id=0xb573, seq=4/1024, ttl=64 (request in 63) |
| 65 | 542.557803 | 10.0.1.101 | 10.0.1.100 | ICMP | 98 | Echo (ping) request id=0xb673, seq=5/1280, ttl=64 (reply in 66) |
| 66 | 542.558056 | 10.0.1.100 | 10.0.1.101 | ICMP | 98 | Echo (ping) reply id=0xb673, seq=5/1280, ttl=64 (request in 65) |

**vi. From PC2 ping PC3.**

PC2> ping 10.0.1.120

host (255.255.255.240) not reachable

```
PC4> ping 10.0.1.100

host (255.255.255.240) not reachable
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 67 | 691.656031 | Private_66:68:01 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.101 |
| 68 | 692.656665 | Private_66:68:01 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.101 |
| 69 | 693.657331 | Private_66:68:01 | Broadcast | ARP | 64 | Who has 255.255.255.240? Tell 10.0.1.101 |

Save the Wireshark output to a text file (using the “Packet Summary” option from “Print”) , and save the output of the ping commands. Note that not all of the above scenarios are successful. Save all the output including any error messages.

When you are done with the exercise, reset the interfaces to their original values as given Table 6.1. (Note that /24 corresponds to network mask 255.255.255.0. and /28 to network mask 255.255.255.240)

**VII. EXERCISES**

**VII)**

**1)**
**• What is the destination MAC address of an ARP Request packet?**
**• What are the different Type Field values in the Ethernet headers that you observed?**
**• Use the captured data to analyze the process in which ARP acquires the MAC address for IP address 10.0.1.12.**

**a)**Dst: Broadcast (ff:ff:ff:ff:ff:ff)
**b)**
Ethernet II, Src:
Private_66:68:01
(00:50:79:66:68:01),
Dst:
Private_66:68:00
(00:50:79:66:68:00)
Destination:
Private_66:68:00
(00:50:79:66:68:00)
Source:
Private_66:68:01
(00:50:79:66:68:01)
Type: IPv4
(0x0800)
Internet Protocol Version 4, Src: 10.1.1.12,
Dst: 10.1.1.11
Internet Control Message Protocol
Type:ARP

**1**.ARP Request: A device wants to find the MAC address of the device with IP address 10.0.1.12. It sends an ARP Request packet with its own MAC and IP addresses and the target IP address (10.0.1.12) with an unknown MAC address. The destination MAC address in the Ethernet frame is the broadcast address (FF:FF:FF:FF:FF:FF).
**2.**ARP Response: The device with IP address 10.0.1.12 recognizes its IP address in the ARP Request. It sends an ARP Response packet containing its own MAC address (00:50:79:66:68:01) and IP address back to the requesting device. The destination MAC address in the Ethernet frame is set to the requesting device's MAC address.
**3.**Updating ARP Cache: The requesting device receives the ARP Response, extracts the MAC address (00:50:79:66:68:01), and updates its ARP cache to associate it with IP address 10.0.1.12. This allows the device to communicate directly with the device at that IP address on the local network.

**2)**
**Based On Lab Question 2**
**• Explain how the ping packets were issued by the hosts with duplicate addresses.**
The duplicating and sharing of the addresses did not work**.**
**• Did the ping command result in error messages?**
The ping command worked as the duplication failed

**• How can duplicate IP addresses be used to compromise the data security?**
Duplicate IP addresses can compromise data security by enabling IP spoofing, where attackers send packets with falsified source addresses to deceive recipients.
 For example, an attacker can send ARP packets with a duplicate IP address, causing incorrect entries in the target's ARP cache. This can redirect traffic to the attacker's device.
**• Give an example. Use the ARP cache and the captured packets to support your explanation.**
**Based On Lab Question 3**
**• Use your output data and ping results to explain what happened in each of the ping commands.**
**• Which ping operations were successful and which were unsuccessful? Why?**

 In a ping command analysis:
 Ping from Device C to duplicate IP may succeed, but data goes to the wrong device.
 Ping from Device A to its duplicate IP will likely succeed due to correct ARP cache.
 Ping from Device B to its duplicate IP will also likely succeed.

**3)**
**• Use your output data and ping results to explain what happened in each of the ping commands.**
**• Which ping operations were successful and which were unsuccessful? Why?**

i. From PC1 ping PC3. - successful
ii. From PC1 ping PC2.- successful
iii. From PC1 ping PC4.- timeout

iv. From PC4 ping PC1.- not reachable
v. From PC2 ping PC4.- not reachable
Either the machine does not exist on the network or we are trying to connect 2 different networks.