

ENSEA

Beyond Engineering

TP1 NETWORKING

RTS TP1

PAN Jimmy-Antoine - LI Guillaume

Part 1



FIGURE 1 – Router on GNS3

We added a router R1. At right, we can see that we started all nodes. The router image disk is well implemented.

Part 2

Why did we use the hub in between the two VPCs ? Could we have connected them directly ?

We used a hub between the two VPCs to connect them together in the same Local Area Network (LAN). If there are only two computer, we can connect them directly with Ethernet by using a crossover cable (Ethernet cable).

What is the difference between a hub and an Ethernet switch ?

When a hub receive an information it sends back to every ports, so to every device in the network. Why ? Because a hub can't tell differences from one computer to another. Whereas the switch has a "table" which matches ports to connected devices by their MAC addresses. The table is created when the first time data go through the switch.

```
jimmypan — PC1 — telnet localhost 5001 — 56...
~ — PC1 — telnet localhost 5001
Trying ::1...
Connected to localhost.
Escape character is '^]'.
PC1> ip 10.0.1.11/24
Checking for duplicate address...
PC1 : 10.0.1.11 255.255.255.0
PC1> ping 10.0.1.12
84 bytes from 10.0.1.12 icmp_seq=1 ttl=64 time=0.532 ms
84 bytes from 10.0.1.12 icmp_seq=2 ttl=64 time=1.630 ms
84 bytes from 10.0.1.12 icmp_seq=3 ttl=64 time=1.095 ms
84 bytes from 10.0.1.12 icmp_seq=4 ttl=64 time=1.109 ms
84 bytes from 10.0.1.12 icmp_seq=5 ttl=64 time=0.934 ms

jimmypan — PC2 — telnet localhost 5003 — 5...
~ — PC2 — telnet localhost 5003
Trying ::1...
Connected to localhost.
Escape character is '^]'.
PC2> ip 10.0.1.12/24
Checking for duplicate address...
PC2 : 10.0.1.12 255.255.255.0
PC2> ping 10.0.1.11
84 bytes from 10.0.1.11 icmp_seq=1 ttl=64 time=1.082 ms
84 bytes from 10.0.1.11 icmp_seq=2 ttl=64 time=1.075 ms
84 bytes from 10.0.1.11 icmp_seq=3 ttl=64 time=1.078 ms
84 bytes from 10.0.1.11 icmp_seq=4 ttl=64 time=0.675 ms
84 bytes from 10.0.1.11 icmp_seq=5 ttl=64 time=1.120 ms
```

FIGURE 2 –

We set PC1 address : 10.0.1.11/24 and PC2 address : 10.0.1.12/24. We can see with our 2 pings that PC1 and PC2 communicate.

Part 3

Now we want to create a network of 4 VPC with a hub between them with the following IP addresses :

VMs	IP addresses
PC1	10.0.1.11/24
PC2	10.0.1.12/24
PC3	10.0.1.13/24
PC4	10.0.1.14/24

TABLE 1 – IP addresses part 3

After having created the network, we can verify if everything goes well by sending pings between the 4 VPCs :

```

jimmypan — PC1 — 55x24
~ — PC1
Trying ::1...
Connected to localhost.
Escape character is '^]'.
PC1> ip 10.0.1.11
Checking for duplicate address...
PC1 : 10.0.1.11 255.255.255.0
PC1> ping 10.0.1.12 -c 1
84 bytes from 10.0.1.12 icmp_seq=1 ttl=64 time=0.327 ms
PC1> ping 10.0.1.13 -c 1
84 bytes from 10.0.1.13 icmp_seq=1 ttl=64 time=1.144 ms
PC1> ping 10.0.1.12 -c 1
84 bytes from 10.0.1.12 icmp_seq=1 ttl=64 time=1.089 ms
PC1> ping 10.0.1.14 -c 1
84 bytes from 10.0.1.14 icmp_seq=1 ttl=64 time=1.235 ms
PC1>

jimmypan — PC2 — telnet localhost 5007 — 5...
~ — PC2 — telnet localhost 5007
Trying ::1...
Connected to localhost.
Escape character is '^]'.
PC2> ip 10.0.1.12
Checking for duplicate address...
PC2 : 10.0.1.12 255.255.255.0
PC2> ping 10.0.1.11 -c 1
84 bytes from 10.0.1.11 icmp_seq=1 ttl=64 time=1.169 ms
PC2> ping 10.0.1.13 -c 1
84 bytes from 10.0.1.13 icmp_seq=1 ttl=64 time=1.172 ms
PC2> ping 10.0.1.14 -c 1
84 bytes from 10.0.1.14 icmp_seq=1 ttl=64 time=1.141 ms
PC2>

jimmypan — PC3 — telnet localhost 5009 — 56...
~ — PC3 — telnet localhost 5009
Trying ::1...
Connected to localhost.
Escape character is '^]'.
PC3> ip 10.0.1.13
Checking for duplicate address...
PC3 : 10.0.1.13 255.255.255.0
PC3> ping 10.0.1.11 -c 1
84 bytes from 10.0.1.11 icmp_seq=1 ttl=64 time=1.415 ms
PC3> ping 10.0.1.12 -c 1
84 bytes from 10.0.1.12 icmp_seq=1 ttl=64 time=1.409 ms
PC3> ping 10.0.1.14 -c 1
84 bytes from 10.0.1.14 icmp_seq=1 ttl=64 time=1.003 ms
PC3>

jimmypan — PC4 — telnet localhost 5011 — 58...
~ — PC4 — telnet localhost 5011
Trying ::1...
Connected to localhost.
Escape character is '^]'.
PC4> ip 10.0.1.14
Checking for duplicate address...
PC4 : 10.0.1.14 255.255.255.0
PC4> ping 10.0.1.11 -c 1
84 bytes from 10.0.1.11 icmp_seq=1 ttl=64 time=0.464 ms
PC4> ping 10.0.1.12 -c 1
84 bytes from 10.0.1.12 icmp_seq=1 ttl=64 time=0.970 ms
PC4> ping 10.0.1.13 -c 1
84 bytes from 10.0.1.13 icmp_seq=1 ttl=64 time=1.230 ms
PC4> ping 10.0.1.14 -c 1
10.0.1.14 icmp_seq=1 ttl=64 time=0.001 ms
PC4>

```

FIGURE 3 –

We deduce that all VPCs communicate well.

Part 4

Now, we use Wireshark to analyze the packets sent between VPCs. Actually, we will focus on the packets sent and received by PC1. To do so, we will filtering the capture by entering in the filter the command : " ip.addr==10.0.1.11 " which is the IP address of VPC1. We get the following capture after sending packets from PC1 to PC2 (PC1% ping 10.0.1.12 -c) :

No.	Time	Source	Destination	Protocol	Length	Info
7	19.810573	10.0.1.11	10.0.1.12	ICMP	98	Echo (ping) request id=0x3ce2, seq=1/256, ttl=64 (reply in 8)
8	19.810958	10.0.1.12	10.0.1.11	ICMP	98	Echo (ping) reply id=0x3ce2, seq=1/256, ttl=64 (request in 7)
9	20.812271	10.0.1.11	10.0.1.12	ICMP	98	Echo (ping) request id=0x3de2, seq=2/512, ttl=64 (reply in 10)
10	20.812941	10.0.1.12	10.0.1.11	ICMP	98	Echo (ping) reply id=0x3de2, seq=2/512, ttl=64 (request in 9)
11	21.815350	10.0.1.11	10.0.1.12	ICMP	98	Echo (ping) request id=0x3ee2, seq=3/768, ttl=64 (reply in 12)
12	21.815944	10.0.1.12	10.0.1.11	ICMP	98	Echo (ping) reply id=0x3ee2, seq=3/768, ttl=64 (request in 11)
13	22.818788	10.0.1.11	10.0.1.12	ICMP	98	Echo (ping) request id=0x3fe2, seq=4/1024, ttl=64 (reply in 14)
14	22.819649	10.0.1.12	10.0.1.11	ICMP	98	Echo (ping) reply id=0x3fe2, seq=4/1024, ttl=64 (request in 13)
15	23.821633	10.0.1.11	10.0.1.12	ICMP	98	Echo (ping) request id=0x40e2, seq=5/1280, ttl=64 (reply in 16)
16	23.822320	10.0.1.12	10.0.1.11	ICMP	98	Echo (ping) reply id=0x40e2, seq=5/1280, ttl=64 (request in 15)

FIGURE 4 –

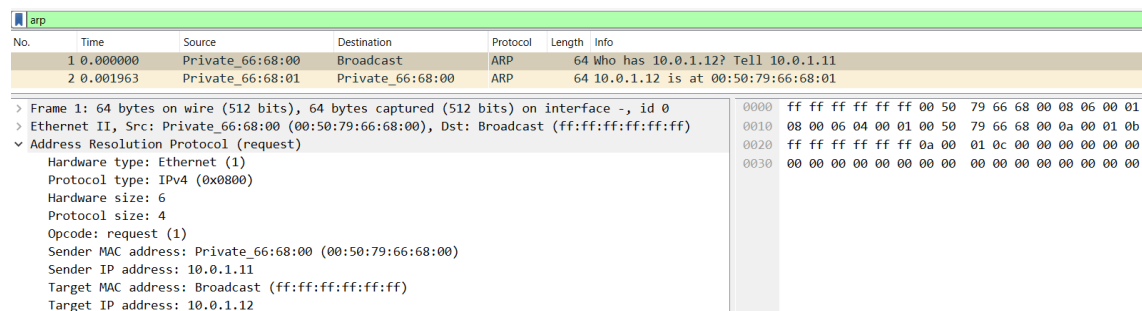
We can see in odd lines that VPC1 is the sender (request) and VPC2 is the receiver (reply).

Part 5

Exercise 5(A). A simple experiment with ARP

We first watch the ARP cache of the PC1 by writing the command `arp` and we see that it's empty. We then ping the computer PC2 (by sending 2 ICMP packet) which has the IP address 10.0.1.12 by typing `ping 10.0.1.12 -c 2`. On Wireshark we can see that there ICMP frames and ARP frames (explained after with the questions). Then when we watch the ARP cache of PC1, we can see that there is the information of PC2 saved.

What is the destination MAC address of an ARP Request packet ?



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Private_66:68:00	Broadcast	ARP	64	Who has 10.0.1.12? Tell 10.0.1.11
2	0.001963	Private_66:68:01	Private_66:68:00	ARP	64	10.0.1.12 is at 00:50:79:66:68:01

> Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface -, id 0
 > Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 > Address Resolution Protocol (request)
 Hardware type: Ethernet (1)
 Protocol type: IPv4 (0x0800)
 Hardware size: 6
 Protocol size: 4
 Opcode: request (1)
 Sender MAC address: Private_66:68:00 (00:50:79:66:68:00)
 Sender IP address: 10.0.1.11
 Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
 Target IP address: 10.0.1.12

FIGURE 5 – Wireshark's screenshot of destination MAC address

As we can see on the picture the MAC address of an ARP Request packet is the Broadcast one so it's `ff :ff :ff :ff :ff :ff`.

What are the different Type Field values in the Ethernet headers that you observed ?

The different Type Field values in the Ethernet headers that we can observe are : the MAC address destination, MAC address source, the type of protocol (here ARP).

Use the captured data to analyze the process by which ARP acquires the MAC address for IP address 10.0.1.12.

To acquire the MAC address for IP address 10.0.1.12 the ARP send first a broadcast request to ask who has this address. Then the device with this address answer back to the computer who requested that he's there.

Exercise 5(B). Matching IP addresses and MAC addresses

We can get the Mac address of each VPCs by issuing a ping command from that host to every other host on the network :

VMs	IP Address of eth0	MAC address of eth0
PC1	10.0.1.11 / 24	00:50:79:66:68:00
PC2	10.0.1.12 / 24	00:50:79:66:68:01
PC3	10.0.1.13 / 24	00:50:79:66:68:02
PC4	10.0.1.14 / 24	00:50:79:66:68:03

FIGURE 6 – Mac Addresses of VPCs

Exercise 5(C). ARP requests for a non-existing address

Here we will observe what happens when an ARP Request is issued for an IP address that does not exist. We watch on Wireshark packets coming from IP address of PC1. We ping from PC1 to 10.0.1.22 (an address that does not exist in the network configuration) and then observe the output.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Private_66:68:00	Broadcast	ARP	64	Who has 10.0.1.22? Tell 10.0.1.11
2	1.011933	Private_66:68:00	Broadcast	ARP	64	Who has 10.0.1.22? Tell 10.0.1.11
3	2.027184	Private_66:68:00	Broadcast	ARP	64	Who has 10.0.1.22? Tell 10.0.1.11

> Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface -, id 0 > Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff) > Address Resolution Protocol (request) Hardware type: Ethernet (1) Protocol type: IPv4 (0x0800) Hardware size: 6 Protocol size: 4 Opcode: request (1) Sender MAC address: Private_66:68:00 (00:50:79:66:68:00) Sender IP address: 10.0.1.11 Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff) Target IP address: 10.0.1.22		0000 ff ff ff ff ff ff 00 50 79 66 68 00 08 06 00 01 0010 08 00 06 04 00 01 00 50 79 66 68 00 0a 00 01 0b 0020 ff ff ff ff ff ff 0a 00 01 16 00 00 00 00 00 00 0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
---	--	---

FIGURE 7 – Wireshark's screenshot of unsuccessful ARP Request

Using the saved output, describe the time interval between each ARP Request packet issued by PC1. Observe the method used by ARP to determine the time between retransmissions of an unsuccessful ARP Request.

The time interval between each ARP request packet is about 1.01 seconds.

In your opinion, why are ARP Request packets not transmitted (i.e., not encapsulated) as IP packets? (Tip : In your answer consider for the layer the ARP protocol sits on.)

ARP Request are not transmitted as IP packets because IP protocol is at the network layer (3rd layer in OSI model). The ARP protocol is oftenly seen as a 2nd and half layer because it allows to make the link between the IP protocol which uses IP addresses to construct its packets and Ethernet frames (data link layer, 2nd layer in OSI model) which use MAC address.

Part 6

We set up a new network configuration as following :

VMs	IP addresses	Network Mask
PC1	10.0.1.100/24	255.255.255.0
PC2	10.0.1.101/28	255.255.255.240
PC3	10.0.1.120/24	255.255.255.0
PC4	10.0.1.121/28	255.255.255.240

TABLE 2 – IP addresses part 6

Now, we execute the following commands :

- From PC1 ping PC3 : PC1% ping 10.0.1.120 -c 1
- From PC1 ping PC2 : PC1% ping 10.0.1.101 -c 1
- From PC1 ping PC4 : PC1% ping 10.0.1.121 -c 1
- From PC4 ping PC1 : PC4% ping 10.0.1.100 -c 1
- From PC2 ping PC4 : PC2% ping 10.0.1.121 -c 1
- From PC2 ping PC3 : PC2% ping 10.0.1.120 -c 1

VMs	IP addresses	Range
PC1	10.0.1.100/24	10.0.1.1 to 10.0.1.255
PC2	10.0.1.101/28	10.0.1.97 to 10.0.1.111
PC3	10.0.1.120/24	10.0.1.1 to 10.0.1.255
PC4	10.0.1.121/28	10.0.1.113 to 10.0.1.127

TABLE 3 – IP addresses part 6 and their ranges

After executing the commands, we have these results :

- From PC1 ping PC3 : there is a connexion, PC3 is in the range of PC1.
- From PC1 ping PC2 : there is a connexion, PC2 is in the range of PC1.
- From PC1 ping PC4 : there is a connexion, PC2 is in the range of PC1.
- From PC4 ping PC1 : there is no connexion, PC1 is not in the range of PC4
- From PC2 ping PC4 : there is no connexion, PC1 is not in the range of PC4
- From PC2 ping PC3 : there is a connexion, PC3 is in the range of PC2.