e TCP/IP

Dans ce TP, nous proposons d'observer et de comprendre certains principes de fonctionnement des réseaux et plus particulièrement d'un réseau TCP/IP. Nous verrons en particulier :

- l'encapsulation des données et le modèle en couches
- le routage et l'interconnexion des réseaux
- les protocoles ARP, HTTP, DNS
- le protocole de connexion de TCP
- ...

Certains faits nouveaux, dont je n'aurais pas eu connaissance, peuvent gêner le bon déroulement de ce TP. Si vous constatez que des manipulations ne fonctionnent pas ou plus, merci de me le signaler.

Afin de faciliter la correction de ce travail, merci d'indiquer clairement les numéros des questions dans votre compte-rendu.

1 Encapsulation des données

1.1 Capture d'un Ping

Pour découvrir le fonctionnement du système en couches et de l'encapsulation, nous allons utiliser un logiciel nommé Wireshark qui permet de capturer des trames réseau.

L'exemple qui suit a été obtenu en utilisant le programme ping. La machine source (1) a pour adresse 10.16.90.17 et la machine distante (2) pour adresse 10.16.90.70. Depuis la machine 1, nous *pinguons* la machine 2 :

ping 10.16.90.70

nous utilisons le programme ping (couche application) dont le but est d'envoyer des messages ICMP (couche transport/internet) de type *Echo Request*. Lorsque la machine distante reçoit un tel message, elle doit y répondre en envoyant à son tour via le protocole ICMP des messages de type *Echo Reply*. Ceci permet par exemple de tester si la connexion est bonne entre deux machines, au moins au niveau de la couche internet.

Sur la machine 1, le logiciel Wireshark a été lancé, pendant le ping, et nous donne pour résultats :

No. Time Source Destination Protocol Info 9 0.000168 10.16.30.70 10.16.30.70 10.16.30.70 10.46.30.71 10.16.30.70 10.46.30.71 5 0.395805 10.16.30.70 10.16.30.70 10.16.30.70 10.46.30.71 10.46.30.71 10.46.30.71 10.46.30.71 10.46.30.71 10.46.30.70 10.46.30.71 10.46.30.70 10.46.30.71 10.46.30.70 10.46.30.70 10.46.30.71	<u>File Edit Capture Display Tools</u>					Help		
3 0.000066 10.16.30.7 10.16.30.7 10.46.30.7 10.46.30.7 5 0.399805 10.16.30.7 10.16.30.7 10.46.30.7 10.46.30.7 5 0.399805 10.16.30.7 10.16.30.7 10.46.30.7 10.46.30.7 6 0.399703 10.16.30.7 10.46.30.7 10.46.30.7 10.46.30.7 7 2.000393 10.16.30.7 10.16.30.7 10.46.30.7 10.46.30.7 8 2.00130 11.46.30.7 10.46.30.7 10.46.30.7 10.46.30.7 8 2.00130 11.66.30.7 10.46.30.7 10.46.30.7 10.46.30.7 9 2.00130 10.46.30.7 10.46.30.7 10.46.30.7 10.46.30.7 9 2.00130 10.46.30.7 10.46.30.7 10.46.30.7 10.46.30.7 9 2.00130 10.46.30.7 10.46.30.7 10.46.30.7 10.46.30.7 9 2.00130 10.46.30.7 10.46.30.7 10.46.30.7 10.46.30.7 9 2.00130 10.46.7 10.46.20.7 10.46.20.7 10.46.20.7 9 10.60.7 10.46.20.7	No. Time Source	Destination	Protocol	Info				
4 0.000188 10.16.90.70 10.16.90.71 ICMP Echo (ping) reply 5 0.95963 10.16.90.77 10.16.90.70 ICMP Echo (ping) reply 7 2.00099 10.16.90.77 10.16.90.70 ICMP Echo (ping) reply 7 2.00099 10.16.90.77 10.16.90.70 ICMP Echo (ping) reply 7 2.00099 10.16.90.74 Cobured) Arrival Time: Jan 16, 2004 11:14:17.1509 Time dalta from previous packet: 0.000133 seconds Frame Number: 3 Packet Length: 74 bytes Capture Length: 74 bytes Capture Length: 74 bytes Capture Length: 74 bytes Capture 10:00:00:07:00:a8:19 (00:00:07:00:a8:19) Source: 00:50:05a:a7:01:6a (00:50:0b:a7:01:a8:19) Source: 00:50:05a:a7:01:6a (00:50:0b:a7:01:a8:19) Source: 00:50:05a:a7:01:6a (00:50:0b:a7:01:a8:19) Source: 00:50:05a:a7:01:6a (00:50:0b:a7:01:a8:19) Source: 00:50:00 (None) 000 = Precedence: routine (0) = 0.e Stormal = 0.e St	3 0.000168 10.16.90.17	10.16.90.70	ICMP	Echo (ping) request			
3 0.388883 10.16.80.17 10.16.30.17 11.0P Etho (ping) request 6 0.389703 10.16.80.17 10.16.30.17 11.0P Etho (ping) request 7 2.000393 10.16.80.17 10.16.30.17 11.0P Etho (ping) request 8 2.001101 10.16.80.17 10.16.30.17 11.0P Etho (ping) request Arrival Time: Jan 16.2004 111:14:17.1509 Time dalta fram previous packet: 0.000133 seconds Frame Number: 3 Packet Length: 74 bytes Capture Length: 74 bytes Capture Length: 74 bytes Distination: 00.60:b7:00:a0:10:01:a0:10:16:a0) Type: IP (0x0800) Internet Protocol Version: 4 Header Length: 20 bytes Dype of service: 0x00 (None) 000 = Precedence: routine (0) = Throughuit: Normal = Thoroughuit: Normal = Thoroughuit: Normal = One fragments: Not set = One fragments: Not set = One (bing) request) Code: 10.0P (0x01) Header Endextum: 0b+10 (correct) Source: 0.10.16.00.70 (10.16.30.70) Destination: 10.16.30.70 (10.16.30.70) Destination: 10.1	4 0.000188 10.16.90.70	10.16.90.17	ICMP	Echo (ping)) reply	- 11		
7 2:000999 t0.16.90.17 10.16.90.70 ICMP Echo (ping) request 8 2.001130 10.16.90.70 10.16.90.70 TCMP Echo (ping) request Arrival Time: 12 no 16, 2004 11:14:17.1509 Time delta from previous packet: 0.000139 seconds Frame Number: 3 Packet Length: 74 bytes Cature Length: 74 bytes Cature Length: 74 bytes Cature Length: 74 bytes Cature Length: 74 bytes Ethernet I Destination: 00:db:b7:0a:a8:19 (00:d0:b7:0a:a8:19) Source: 00:50:ba:af:01:6a (00:50:ba:af:01:6a) Type: 1P (0x0800) © Internet Protocol Wesder Length: 20 bytes El Type of Service: 0x00 (None) 000 = Precedence: routine (0) 0 = Reliability: Normal 0 = Reliability: Normal 0 = Role fisher Normal 0 = Nore fragment: Not set # More fragment: Not set 0 = Nore (15.00.70 (10.16.90.70) Bestination: 10.16.90.70 (10.16.90.70) Bestination: 10.16.90.70 (10.16.90.70) Destination: 10.16.90.70 (10.16.90.70) Bestine: 00455C Identifier: 0x0200	5 0.333685 10.16.30.17	10.16.90.70	TCMP	Echo (ping) request	н		
8 2.000030 10.16.90.70 10.16.90.12 TCMP Fcho (nind) really 4 Fries 3 (74 on wire, 74 captured) Arrival Time: Jan 16: 2004 11:14:17.1509 Time delta from previous packet: 0.000139 seconds Frame Number: 3 Packat Length: 74 bytes Capture Length: 74 bytes Capture Length: 74 bytes Destination: 00:db:70:ex8:19 (00:db:7:0a:a8:19) Source: 00:50:ba:af:01:6a (00:50:ba:af:01:6a) Type: 17 (0x8000) Elternet Frotocol Version: 4 Header Tength: 20 bytes Eltype of service: 0x00 (None) 000 = Precedence: routine (0) 0 = Reliability: Normal 0 = cost: Normal 0 = ofragments: Not set 0 = ofragments: Not set 0 = Nore fragment: Not set 0 = Nore fragment: Not set 0 = Nore fragment: Not set 0 = Nore fragments: Not set Fragment of fiel: 0 Time to live: 1200 Destination: 10.16.90.70 (10.16.90.70) Header Checksum: 0x465c Identifier: 0x0200 Sequence number: 1200 Data (32 bytes)	7 2.000999 10.16.90.17	10.16.90.70	ICMP	Echo (ping) request	- 11		
P Freene 3 (24 on wire, 74 captured) Arrival Time: Jan 16, 2004 11:14:17.1509 Time delta from previous packet: 0.000139 seconds Frame Number: 3 Packet Length: 74 bytes Capture Length: 74 bytes Ethernet II Destination: 00:d0:b7:0a:a8:19 (00:d0:b7:0a:a8:19) Source: 00:50:ba:af:01:6a (00:50:ba:af:01:6a) Type: IP (0x0800) E Internet Protocol Wersion: 4 Header length: 20 bytes E Type of service: 0x00 (Mone) 000 = Percedence: routine (0) 0. = Reliability: Normal 0. = Reliability: Normal 0. = Reliability: Normal 0. = Rois Normal 0. = Nore fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (Ox01) Header checksum: 0x4610 (correct) Source: 10.15.90.70 (10.16.90.70) E Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Other Kesum: 0x465C Identifier: 0x0200 Sequence number: 1200 Data (32 bytes) 00000	8 2.001030 10.16.90.70	10.16.90.17	TCMP	Fchn (nina) renly	IZ		
Arrival Time: Jan 16, 2004 11:14:17.1509 Time delta from previous packat: 0.000133 seconds Frame Number: 3 Packat Length: 74 bytes Capture Length: 74 bytes Capture Length: 74 bytes Capture Length: 74 bytes Destination: 00:06:b7:0a:a8:19 (00:d0:b7:0a:a8:19) Source: 00:50:ba:af:01:6a (00:50:ba:af:01:6a) Type: IP (0x0800) I Internet Protocol Version: 4 Header length: 20 bytes I Type of service: 0x00 (None) 000 = Precedence: routine (0) 0 = OHay: Normal 0 = OHay: Normal 0 = OHay: Normal 0 = OHay: Normal 0 = Cost: Normal Total Length: 60 Identification: 0x27a I Fragment: Not set 0. = More fragment: Not set 0 0 0. (0. 16, 90, 17 (0. 16, 90, 70) I Internet Control Message Protocol Type: 8 (Ccho (ping) request) Code: 0 0000 00 db 17 09 as 19 00 50 ba af 01 69 00 10 01 04 10 10 ac 05 00 10 10 10 32 (2 2 38 00 00 00 01 45 10 04 05 01 04 01 05 32 10 00 00 01 45 00 07 01 05 10 65 65 65 67 68 65 0 0 0 0 0 0 0	■ Frame 3 (74 on wire, 74 captured	0						
Time delta from previous packet: 0.000139 seconds Freme Number: 3 Packet Length: 74 bytes Gapture Length: 74 bytes Ethernet II Destination: 00:00:b7:0a:a8:19 (00:d0:b7:0a:a8:19) Source: 00:50:ba:af:01:6a (00:50:ba:af:01:6a) Type: IP (0x0800) Internet Protocol Version: 4 Header length: 20 bytes IType of service: 0x00 (None) 000 = Delay: Normal 0 = Polay: Normal 0 = Throughput: Normal 0 = Cost: Normal Total Length: 60 Identification: 0x273a I Flags: 0x00 Time to live: 128 Protocol ICMP (0x01) Header checksum: 0x4610 (correct) Source: 10.16.90.17 (10.16.90.70) Internet Control Messae Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) Dotto 0 00 do b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Sequence number: 1280 Data (32 bytes) Dotto 0 0 col 57 3a 00 00 80 01 44 h1 0 0a 10 5a 11 ba 10 Sequence number: 1280 Data (32 bytes) Dotto 0 0 col 57 3a 00 00 80 01 44 h1 0 0a 10 5a 11 ba 10 Sequence number: 1280 Data (32 bytes) Dotto 0 0 col 57 3a 00 00 80 01 44 h1 0 0a 10 5a 11 ba 10 Sequence number: 1280 Data (32 bytes) Dotto 0 0 col 56 66 67 68 69 Vabcdefg h1 Eimer	Arrival Time: Jan 16, 2004 11	:14:17.1509						
Frame Number: 3 Packet Length: 74 bytes Elthernet II Destination: 00:d0:b7:0a:a8:19 (00:d0:b7:0a:a8:19) Source: 00:50:ba:af:01:5a (00:50:ba:af:01:5a) Type: IP (0x0800) Elnternet Protocol Version: 4 Header length: 20 bytes El Type of service: 0x00 (None) 000 = Precedence: routine (0) 0 = Fliability: Normal 0 = cliability: Normal	Time delta from previous pack	et: 0.000139 seconds						
Packet Length: 74 bytes Capture Length: 74 bytes Ethernet II Destination: 00:d0:b7:0a:a8:19 (00:d0:b7:0a:a8:19) Source: 00:50:ba:af:01:6a (00:50:ba:af:01:6a) Type: IP (0x0800) E Internet Protocol Wersion: 4 Header length: 20 bytes E Type of service: 0x00 (None) 000 = Precedence: routine (0) = Delay: Normal 0 = Throughput: Normal 0 = Reliability: Normal 0 = Reliability: Normal 0 = Reliability: Normal 0 = Reliability: Normal 0 = Cost: Normal Total Length: 60 Identification: 0x27a E Flags: 0x00 .0 = Don't fragment: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.77 (10.16.90.77) Destination: 10.16.90.70 (10.16.90.70) E Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Othecksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) D000 00 db 70 aa 88 19 00 50 ba af 01 6a 08 00 45 00 Sequence number: 1280 Data (52 bytes) D000 00 db 70 aa 88 19 00 50 ba af 01 6a 08 00 45 00 Sequence number: 1280 Data (52 bytes) D000 07 61 62 63 64 65 66 67 68 63 Vabcdefg hi Emerl	Frame Number: 3							
Capture Length: /4 bytes Ethernet II Destination: 00:d0:b7:0a:a0:19 (00:d0:b7:0a:a0:19) Source: 00:50:ba:af:01:5a (00:50:ba:af:01:5a) Type: IP (0x08000) El Internet Protocol Version: 4 Header length: 20 bytes El Type of service: 0x00 (None) 000 = Percedence: routine (0) 0 = Delay: Normal 0 = Cost:	Packet Length: 74 bytes							
Destination: 00:d0:b7:0a:a8:19 (00:d0:b7:0a:a8:19) Source: 00:50:ba:af:01:6a (00:50:ba:af:01:6a) Type: 1P (0x0800) E Internet Protocol Version: 4 Header length: 20 bytes E Type of service: 0x00 (None) 000 = Precedence: routine (0) = Delay: Normal 0 = Delay: Normal 0 = Delay: Normal 0 = Roughput: Normal 0 = Cost: Normal Total Length: 60 Identification: 0x273a E Flags: 0x00 .0 = Nore fragment: Not set 0. = More fragment: Not set 0. = Nore fragment: Not set 	Capture Length: 74 bytes							
Source: 00:50:ba:af:01:5a (00:50:ba:af:01:5a) Type: IP (0x0000) Uersion: 4 Header length: 20 bytes □ Type of service: 0x00 (None) 000. = Precedence: routine (0) 0	Destination: 00:d0:b7:0a:a8:1	9 (00·d0·b7·00·09·19)						
Type: IP (0x0800) I Internet Protocol Version: 4 Header length: 20 bytes I Type of Service: 8x00 (None) 000 = Precedence: routine (0) 0 = Delay: Normal 0 = Delay: Normal 0 = Reliability: Normal 0 = Cost: Normal Total Length: 60 Identification: 0x273a I Flags: 0x00 .0. = Nore fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.17 (10.16.90.70) I Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x455c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 000 db b7 Da as 19 00 50 ba af 01 Ga 08 00 45 00 Pj.E. 010 00 8: 27 30 00 08 00 14 010 08 11 08 10 Pj.E. 010 00 77 61 62 63 64 65 66 77 071 72 73 74 75 76 yeight/mine papersturv vabcdefg hi	Source: 00:50:ba:af:01:6a (00	:50:ha:af:01:6a)						
<pre> Internet Protocol Version: 4 Header length: 20 bytes IType of service: 0x00 (None) 000 = Precedence: routine (0) = Delay: Normal 0 = Throughput: Normal 0 = Cost: Normal Total Length: 60 1 dentification: 0x273a IF Flags: 0x00 = Don't fragment: Not set = Nore fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.70 (10.16.90.70) Destination: 10.16.90.70 (10.16.90.70) II Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x020 Sequence number: 1280 Data (32 bytes) D000 00 do b7 De as 19 00 50 ba af 01 5a 08 00 45 00Pj.E. D010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10Pj.E. D020 Sa 44 68 00 45 56 65 66 67 06 86 9 Vabue Vab</pre>	Type: IP (0x0800)							
Version: 4 Header length: 20 bytes B Type of service: 0x00 (None) 000 = Precedence: routine (0) 0 = Chalay: Normal 0 = Reliability: Normal 0 = Cost: Normal Total Length: 60 Identification: 0x273a B Flags: 0x00 .0. = More fragments: Not set 0. = Nor' fragment: Not set 0. = Nor' fragment: Not set 0. = Nor' fragment: Not set Pragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.070 (10.16.90.70) Destination: 10.16.90.70 (10.16.90.70) B Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x485c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 db f7 0a a8 19 00 50 ba af 01 5a 08 00 45 00 0020 5a 4 68 00 46 5c 02 00 0 05 00 61 56 26 36 56 56 67 76 86 59 0030 67 68 69 6a 65 cc 20 00 05 00 61 56 26 36 45 56 65 77 77 72 73 74 75 76 shijklin opgrstuv vabcdefg hi 0040 0040 77 61 62 63 64 65 66 67 68 6	Internet Protocol							
Header length: 20 bytes □ Type of service: 0x00 (None) 000 = Precedence: routine (0) 0 = Delay: Normal 0 = Belay: Normal 0 = Reliability: Normal 0. = Cost: Normal Total Length: 60 Identification: 0x273a □ Flags: 0x00 .0. = Don't fragment: Not set 0. = More fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header Checksum: 0x465c Identifier: 0x200 Sequence number: 1280 Data (32 bytes) 0000 000 db b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Pj.E. 0010 00 db b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Pj.E. 0010 00 db b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Data (32 bytes) 0000 000 db b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Data (32 bytes) 0000 000 07 6a 65 65 66 67 68 69 Pj.E. 010 00 77 6i 62 63 64 65 66 67 68 69 Pj.E. 010 00 77 6i 62 63 64 65 66 67 68 69 P	Version: 4							
□ Type of service: 0x00 (None) 000 = Precedence: routine (0) 0.00 = Delay: Normal	Header length: 20 bytes							
000 = Precedence: routine (0) 0 = Chay: Normal 0 = Cost: Normal 0 = Cost: Normal Total Length: 60 Identification: 0x273a Eflags: 0x00 .0 = Don't fragment: Not set 0 = Cost: Normal Total Length: 60 Identification: 0x273a Eflags: 0x00	□ Type of service: 0x00 (None)							
0 = Delay: Normal 0 = Phroughput: Normal 0. = Cost: Normal Total Length: 60 Identification: 0x273a E Flags: 0x00 .0 = Don't fragment: Not set = More fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: 10.16.90.17 (10.16.90.17) Destination: 10.16.90.70 (10.16.90.70) E Internet Control Message Protocol Type: 0 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 do b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Source: 2,Pj.E. 0000 00 ac 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 Source: 2,	000 = Precedence: ro	utine (O)						
1 0 = Milability: Normal 0. = Cost: Normal Total Length: 60 Identification: 0x273a E Flags: 0x00 .0 = More fragment: Not set 0. = More fragment: Not set 0. = More fragment: Not set	= Delay: Normal	rmal						
1	0 = Reliability: N	in mail						
Total Length: 60 Identification: 0x273a E Flags: 0x00 .0. = Don't fragment: Not set = More fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.70 (10.16.90.70) E Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 77 61 62 63 64 65 66 67 68 69 vabcdef pi								
Identification: 0x273a □ Flags: 0x00 .0. = Don't fragment: Not set .0. = More fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4010 (correct) Source: 10.16.90.17 (10.16.90.70) □ Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a æ8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a æ8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a æ8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a æ8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a æ8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 77 61 62 63 64 65 66 67 68 69 Either	Total Length: 60							
□ Flags: 0x00 .0 Don't fragment: Not set	Identification: 0x273a							
.0. = Don't fragment: Not set 0. = More fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.70 (10.16.90.70) El Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x405c Identifier: 0x405c Identifier: 0x405c Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 77 61 62 63 64 65 66 67 68 69 Elle <contures 0<="" drops:="" td=""><td>⊟ Flags: 0x00</td><td></td><td></td><td></td><td></td><td></td></contures>	⊟ Flags: 0x00							
P More fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.77 (10.16.90.70) Elinternet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 50Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0000 77 61 62 63 64 65 66 67 68 69 vabcdef d1 77 75 76 vabcdef yabcdef yabcde	.O = Don't fragment: Not	set						
Fragment offset: 0 Time to live: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.17 (10.16.90.70) Destination: 10.16.90.70 (10.16.90.70) E Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .<':kabcdef	O. = More fragments: Not	set						
11me to 11ve: 128 Protocol: ICMP (0x01) Header checksum: 0x4b10 (correct) Source: 10.16.90.17 (10.16.90.17) Destination: 10.16.90.70 (10.16.90.70) Elinternet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 0000 02 c7 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 Pj.E. 0010 00 c2 73 a0 00 08 00 14 4b 10 0a 10 5a 11 0a 10 Pj.E. 0020 5a 46 08 00 46 5c 02 00 05 00 61 62 63 64 65 66 2F, Abcdef 0040 77 61 62 63 64 65 66 67 68 69 vabcdefg hi vabcdefg hi vabcdefg hi	Fragment offset: 0							
Header checksum: 0x4b10 (correct) Source: 10.16.90.17 (10.16.90.70) Binternet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x465c Identifier: 0x465c Joata (32 bytes) 0000 00 db 70 aa 81 90 05 0 ba af 01 6a 08 00 45 00 0000 00 db 70 aa 81 90 05 0 ba af 01 6a 08 00 45 00 0010 03 c27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 Pj.E. 0010 03 c27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 PjE. 0020 5a 46 0s 20 200 05 00 61 62 63 64 65 65 27 F: F. Nabcdef 0030 67 68 69 6a 6b 6c 6d 6e 7 67 17 27 37 47 57 6 0040 77 61 62 63 64 65 66 67 68 63 28 29 2040 205 206 207 208 209 200 200 200 201 202 203 203 204 204 205	Diretocol: TCMD (0v01)							
Source: 10.16.90.17 (10.16.90.72) Destination: 10.16.90.70 (10.16.90.70) Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequece number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 Pj.E. 0020 Sa 46 08 00 45 5c 02 00 05 00 16 22 63 64 65 66 2F.F.Fabcdef 0040 77 61 62 63 64 65 66 67 68 69 Verter Verter Either	Protocol: ICMP (0x01) Header checkman (N+H10 (correct)							
Destination: 10.16.90.70 (10.16.90.70) E Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10Pj.E. 0020 5a 46 08 00 46 5c 02 00 05 00 61 62 63 64 65 66 2FRabcdef 0030 67 68 69 6a 6b 6c 6f 6e 6f 70 71 72 73 74 75 76 ghijklam opgrstuv vabcdefg hi	Source: 10.16.90.17 (10.16.90.17)							
Internet Control Message Protocol Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .<'Pj.E. 0020 5a 46 08 00 46 5c 02 00 05 00 61 62 63 64 65 66 27Rabcdef ghijklem opgrstuv vabcdefg hi Filter Eiler < contures Drops 0	Destination: 10.16.90.70 (10.16.90.70)							
Type: 8 (Echo (ping) request) Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10Pj.E. 0020 5a 46 08 00 46 5c 02 00 05 00 61 62 63 64 65 66 27b.c.abcdef 0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklm opgrstuw vabcdefg hi Either	Internet Control Message Protocol							
Code: 0 Checksum: 0x465c Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .<':KZ 0020 5a 46 08 00 46 5c 02 00 05 00 61 62 63 64 65 66 2FFabcdef 0040 77 61 62 63 64 65 66 67 68 69 Either	Type: 8 (Echo (ping) request)							
Checksum: 0x465c Identifier: 0x405c Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10Pj.E. 0020 5a 46 08 00 45 5c 02 00 05 00 61 62 63 64 65 66 2F.F.Nabcdef 0030 67 58 69 6a 6b 6c 6f 66 7 07 17 72 73 74 75 76 shijklinn opqrstuv vabcdefg hi Either	Code: 0							
Identifier: 0x0200 Sequence number: 1280 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .<'	Checksum: 0x465c							
Sequence number: 1200 Data (32 bytes) 0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Pj.E. 0010 00 ac 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .<': KZ	Identifier: 0x0200							
0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00 Pj.E. 0010 00 a2 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .<':Pj.E.	Data (32 hutes)							
0000 00 d0 b7 0a a8 19 00 50 ba af 01 6a 08 00 45 00Pj.E. 0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .<':Pj.E. 0020 5a 46 08 00 46 5c 02 00 05 00 61 62 63 64 65 66 27.F. Aabcdef 0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijkim opqrstuv 0040 77 61 62 63 64 65 66 67 68 69 Either	bata (52 bytes)							
0010 00 3c 27 3a 00 00 80 01 4b 10 0a 10 5a 11 0a 10 .c ² : KZ 0020 5a 46 08 00 46 5c 02 00 05 00 61 62 63 64 65 66 27 KZ 0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklim opqrstuv 0040 77 61 62 63 64 65 66 67 68 69 vabcdefg hi	0000 00 d0 b7 0a a8 19 00 50 ba	af 01 6a 08 00 45 00	P	jE.		4		
0020 67 68 69 6a 60 40 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklim opgrstuv 0040 77 61 62 63 64 65 66 67 68 69	0010 00 3c 27 3a 00 00 80 01 4b	10 0a 10 5a 11 0a 10 00 61 62 63 64 65 66	75 5	KZ		- 11		
0040 77 51 52 53 54 55 55 57 58 59 wabcdefg hi	0030 67 68 69 6a 6b 6c 6d 6e 6f	70 71 72 73 74 75 76	ghijklmn	opgrstuv		- 11		
Filter	0040 77 61 62 63 64 65 66 67 68	69	vabcdefg	hi		- 11		
Filter								
Filter								
Filter						H		
Theorem in the second of the s	Filter			7 Reset	File: <capture> Drops: 0</capture>			

Dans la partie du haut figurent les trames capturées (elles sont numérotées). Voici comment interprêter ces résultats : La trame 3 est issue de la machine source 10.16.90.17 vers la destination 10.16.90.70, elle utilise le protocole ICMP, et elle est de type *Echo Request*. La trame 4 est la réponse de la machine 10.16.90.70 à la machine 10.16.90.17. Les trames 5,6 puis 7,8 et 9,10 correspondent aux trois autres ping envoyés (pour un total de 4).

Vous allez reproduire cette manipulation.

Si vous êtes gênés par les pings des autres groupes, notez que dans la case Filtre de Wireshark, vous pouvez entrer un filtre pour les trames qui seront capturées. Par exemple : host 10.16.90.70 ne capturera que ce qui concerne 10.16.90.70, src 10.16.90.70 ne capturera que les trames pour lesquelles 10.16.90.70 est émetteur, et dst 10.16.90.70 ne capturera que les trames pour lesquelles 10.16.90.70 est récepteur. La syntaxe pour les filtres de capture est différente de celle des filtres d'affichage. Vous pouvez par exemple tout capturer puis ne filtrer que certaines trames à afficher. Vous indiquerez par ip.addr==10.16.90.70 que vous ne voulez afficher que ce qui concerne 10.16.90.70 (consultez l'aide et les exemples)

Question 1-1

Reproduisez cette manipulation. Pour cela :

- Recherchez l'IP de votre machine et de la machine que vous «pinguerez» (une machine de la salle)
- «Pinguez» la machine cible et capturez les trames sur votre machine (vous pouvez utiliser des filtres (voir ci-dessous))
- Indiquez sur votre rapport votre IP, l'IP cible, une trame echo request et la trame echo reply correspondante (copie d'écran de Wireshark, montrant bien les trames en question).

Rappelez à quoi sert la commande ping.

1.2 Contenu du message ICMP

Question 1-2

Grâce à la capture de Wireshark, pour une message *Echo Request*, identifiez, dans le message ICMP, les informations qui déterminent que le message est justement un *Echo Request*.

Quels sont les autres champs contenus dans le message ICMP ? À quoi servent-ils ?

En ce qui concerne les octets du message ICMP, certains proviennent du programme ping (couche application) et d'autres sont ajoutés par le protocole ICMP.

Question 1-3

Indiquez quelles sont les données ajoutées par ICMP et quelles sont les données produites par le programme ping. Autrement dit, quelle est la partie du message ICMP liée au protocole, et quelle est celle liée au programme ping. Justifiez.

Question 1-4 Demandez à l'encadrant à pouvoir intercepter des messages *Echo Request* provenant d'une machine sous Linux et capturez les trames. Voyez-vous un moyen d'identifier si la machine émettant les messages *Echo Request* fonctionne sous Linux ou bien sous Windows?

1.3 Encapsulation dans un datagramme IP

Le message ICMP est ensuite encapsulé dans un datagramme IP.

Question 1-5

Faites apparaître, en surbrillance, sur une copie d'écran, le contenu de l'en-tête IP ajouté au message ICMP.

Question 1-6

Qu'est ce qui, dans l'en tête IP, permet de savoir que le message encapsulé est un message ICMP ?

1.4 Encapsulation dans une trame Ethernet

Le datagramme IP est ensuite lui même encapsulé dans une trame Ethernet.

Question 1-7

Faites apparaître, en surbrillance, sur une copie d'écran, le contenu de l'en-tête Ethernet ajouté au datagramme IP. Quelle est sa longueur en octets ?

Question 1-8

Déduisez de l'analyse de l'entête Ethernet l'adresse MAC de votre machine et l'adresse MAC de la

machine que vous avez pingué. Présentez ces résultats de manière claire (votre IP - votre @MAC, l'@IP pinguée, l'@MAC de la machine pinguée)

Comment connaître la marque des cartes réseaux utilisées dans la salle ? (<u>Standards.ieee.org</u> ou <u>Mac Find</u>)

1.5 Réponse au message

La trame Ethernet analysée est maintenant *lachée* sur le réseau, récupérée par la machine cible, qui va y répondre par un *Echo Reply*

Question 1-9

Donnez une copie d'écran de la trame Ethernet contenant le message *Echo Reply* **qui correspond** au message *Echo Request* que vous venez d'analyser. Identifiez les différences et les similitudes entre les deux messages *Echo Request* et *Echo Reply* et expliquez-les.

Voici le cheminement des informations entre les deux machines :

Machine 1



Parmi les informations qui s'affichent lors d'un ping figure le temps écoulé entre l'émission d'un *Echo Request* et la réception d'un *Echo Reply* (1 ms ou moins dans l'exemple qui suit).



Question 1-10

Sachant que les réponses n'arrivent pas forcément dans l'ordre d'émission des requêtes et sachant que sur une seule et même machine, on peut pinguer plusieurs fois et en même temps la même machine cible, comment pensez-vous que le délai entre émission et réception puisse être calculé ? Illustrez vos propos avec des exemples recueillis lors d'un test réel.

2 Principes du routage

Nous venons de voir comment les informations transitaient d'une machine 1 vers une machine 2 directement connectées. Que se passe-t-il sur un réseau comme Internet ? Connaître l'adresse IP de la machine destination ne suffit sans doute pas pour la retrouver quelque part dans le monde. C'est là qu'intervient le routage.

Vous pouvez regarder l'IP de votre machine et son masque de sous-réseau à partir d'une console en faisant : ipconfig (Windows, essayez l'option /all) ou /sbin/ifconfig (Linux) Vous devez obtenir quelque chose comme :

```
Adresse IP ..... : 192.168.81.17
Masque de sous-réseau : 255.255.255.0
```

Question 2-1

- Donnez les informations relatives à votre poste (IP, Masque, Classe de réseau, adresse de réseau, adresse de diffusion et passerelle).
- Donnez la plage d'adresses qui peut être utilisée par les machines du même sous-réseau que vous.

Les explication qui suivent sont données pour une topologie de réseau qui n'est pas la vôtre. Vous devez comprendre les explications qui suivent et les refaire (voyez les questions à la fin) dans votre réseau.

Depuis une machine de votre réseau (par exemple la votre), comment joindre la machine 10.16.83.102 ? Il se peut que, comme dans notre exemple les deux réseaux 10.16.80.0 et 192.168.8X.0 soient reliés par l'intermédiaire d'un routeur. Mais peut-être pas... Il se peut que le premier réseau soit relié à un deuxième, qui lui même est relié à un troisième qui lui même est relié à la destination. La force du protocole IP réside en partie dans cette faculté de trouver sa route «tout seul» ou presque. Il suffit que votre machine connaisse l'@ du premier intermédiaire (routeur) et lui envoie le datagramme. La suite n'est plus de son ressort.

Voici comment **pourraient** être connectés les postes (les schémas qui suivent illustrent une ancienne configuration du réseau, qui n'est plus d'actualité. Ce sera à vous, dans la prochaine question, de déterminer la manière dont sont actuellement connectées les machines et les réseaux entre eux):



Sur ce schéma, le réseau 192.168.81.0 est en bleu et le réseau 10.16.80.0 est en rouge. La machine 192.168.81.17 appartient bien au réseau bleu et la machine 10.16.83.102 au réseau rouge. La passerelle du réseau bleu (la machine qui permet au réseau de «sortir») est 192.168.81.1 et elle appartient aussi au réseau rouge.

Une partie de cette structure est emmagasinée dans les machines sous la forme de tables de routage. La table de routage de 192.168.81.17 contient essentiellement comme informations (tapez route print (Windows) ou /sbin/route -n (Linux) pour les voir) le fait que :

- Le réseau 192.168.81.0/255.255.255.0 est accessible directement par l'interface 1
- Pour les autres réseaux, s'adresser à 192.168.81.1

La machine n'ayant qu'une interface (une seule adresse IP), nous lui avons donné le nom 1.

De son côté, la machine 192.168.81.1 a deux IP (elle a aussi 10.16.83.24) et donc deux interfaces. Disons que la première est celle qui correspond à 192.168.81.1. La table de routage de la machine ressemblera à :

- Le réseau 192.168.81.0/255.255.255.0 est accessible directement par l'interface 1
- Le réseau 10.16.80.0/255.255.240.0 est accessible directement par l'interface 2
- Pour les autres réseaux, s'adresser à ...

Les points de suspension signifient que le réseau 10.16.80.0 a sans doute aussi sa passerelle pour sortir vers l'extérieur, mais nous ne l'avons pas faite figurer sur le schéma.

C'est ainsi que la communication se fera de 192.168.81.17 vers 10.16.83.102 :

- 192.168.81.17 consulte sa table de routage et voit que 10.16.83.102 ne fait pas partie d'un réseau accessible directement. Elle envoie le tout à sa passerelle : 192.168.81.1
- La passerelle réceptionne et regarde de son côté si 10.16.83.102 fait partie d'un réseau directement accessible : c'est le cas (interface 2) et elle transmet le tout à 10.16.83.102.

Imaginons, pour terminer cet exemple que 192.168.81.17 veuille communiquer avec 194.254.43.242. Reprenons notre schéma et ajoutons les maillons manquants:



Sur ce nouveau schéma la passerelle de 10.16.80.0 apparaît. Il s'agit de 10.16.81.254, qui appartient aussi au réseau 194.254.43.0/255.255.255.0 avec l'adresse 194.254.43.1. Les numéros des interface figurent sur le schéma (lorsqu'il n'est pas indiqué on parlera de l'interface 1). Le passage des données de 192.168.81.17 vers 194.254.43.242 se fera donc ainsi :

- 192.168.81.17 regarde si 194.254.43.242 appartient à un réseau de sa table de routage. Non. Donc elle transmet à la passerelle 192.168.81.1 sur son interface 1 (en lui indiquant le destinataire final).
- 192.168.81.1 doit transmettre à 194.254.43.242. Cette machine n'appartient pas à un réseau qui figure dans sa table de routage (elle n'a que 192.168.81.0/255.255.255.0 et 10.16.80.0/255.255.240.0). Donc elle transmet à sa passerelle par défaut 10.16.81.254 en indiquant le destinataire final et l'expéditeur. Elle utilise l'interface 2 pour transmettre puisque le réseau de 10.16.81.254 est sur son interface 2.
- 10.16.81.254 doit transmettre à 194.254.43.242. Cette machine appartient à un réseau qui figure dans sa table de routage. Elle lui envoie donc les informations directement par l'interface 2 puisque c'est l'interface qui correspond au réseau de destination.

Question 2-2

Donnez la table de routage de votre machine et commentez (ligne correspondant au réseau local et passerelle par défaut).

Question 2-3

Vous pouvez observer le trajet suivi par les paquets IP en utilisant le programme tracert -d (Windows) ou traceroute -n (Linux). En utilisant ce programme, proposez le schéma réel de la portion de réseau que vous pouvez examiner. Les machines suivantes

- 192.168.8X.XX (au choix)
- 10.16.83.102
- 194.254.43.242

doivent impérativement figurer dans vote schéma.

Question 2-4

Essayez un «traceroute» (et un ping) vers google. Que se passe-t-il et pourquoi ?

Question 2-5 Pinguez la machine 194.254.43.242 depuis votre poste et réalisez une capture avec Wireshark. Que pouvez-vous dire de l'adresse MAC de 194.254.43.242 ?

3 Protocole ARP

Au niveau de la couche accès au réseau la correspondance entre l'adresse IP du premier routeur à joindre et son adresse MAC devait être effectuée. D'où provient cette information ? Il y a deux possibilités :

- soit la machine émettrice connaît déjà l'adresse MAC de la destination car elle a déjà dialogué avec elle (elle la conserve dans un cache)
- soit la machine émettrice ne la connaît pas. Dans ce cas, le protocole ARP (Address Request Protocol) est utilisé pour demander quelle adresse MAC correspond à une certaine IP.

Vous pouvez connaître la liste des adresses MAC que conserve votre machine dans son cache en utilisant la commande arp -a (Windows) ou arp -n (Linux).

Question 3-1 Donnez la liste des machines dans le cache ARP de votre machine. Pinguez une machine qui n'est pas actuellement dans votre cache (et qui est allumée....) et constatez son ajout dans le cache ARP (donnez-des copies d'écran montrant le contenu de votre cache ARP avant et après). Pourquoi la machine 10.16.83.102 n'apparaît-elle pas dans le cache même si vous la pinguez ?

Question 3-2

Vous pouvez utiliser Wireshark pour ne capturer que le protocole ARP en entrant dans la case Filtre : arp Essayez et constatez le nombre de messages concernant ARP et leur provenance. Comment expliquez vous une telle affluence ? En particulier, vous voyez peut être des messages concernant d'autres salles, voire d'autres réseaux.

Question 3-3

Ce système de résolution des adresses MAC fait partie des principes utilisés à des fins malveillantes : Que se passe-t-il à votre avis si lorsque la machine A (son IP est A et son adresse MAC est a) veut communiquer avec la machine C (son IP est C et son adresse MAC est c) et qu'une certaine machine B (son IP est B et son adresse MAC est b) envoie sans cesse des paquets de type ARP Reply indiquant à A que la machine qui a l'ip C, c'est elle-même, avec l'adresse MAC b...?

4 Ports

Certaines manips peuvent ne pas fonctionner correctement. Prévenez l'encadrant lorsque vous en êtes à cette partie du TP

Les numéros de port TCP permettent de cibler telle ou telle application sur une machine serveur. Ces numéros sont plus ou moins normalisés. Le port pour le web (HTTP) est 80, celui pour lire ses mails via POP3 est 110, celui pour envoyer des mails via SMTP est 25. La machine A lorsqu'elle communique **via TCP** pour lire une page web s'adressera au port 80 de la machine B. Sur la machine B, on dira que le serveur web **écoute** sur le port 80.

Sur une machine de type serveur, il y a donc en permanence des programmes (couche application) qui écoutent sur certains ports, en attendant les connexions. Par exemple, le programme Apache, qui est un serveur web écoute généralement le port 80, et postfix qui est un agent de transport de mail écoute le port 25.

Il y a de nombreux autres programmes qui tournent ainsi en permanence et écoutent certains ports, même sur des machines qui ne sont pas des serveurs. Pour connaître la correspondance entre un numéro de port et le service qui est généralement associé, on pourra consulter : http://www.iana.org/assignments/port-numbers.

Un programme qui permet, depuis une machine A de savoir quels sont les ports ouverts sur une machine B s'appelle un scanneur de ports. L'utilisation d'un tel logiciel est soumise à condition. En dehors d'exercices pédagogiques (!) seul l'administrateur d'un réseau peut scanner des ports, et uniquement ceux de ses propres machines.

Question 4-1

À l'aide de nmap déterminez quels sont les ports ouverts sur les postes :

- 192.168.8X.X (choisissez un poste dans la salle)
- 192.168.80.200
- 192.168.81.3

Identifiez les fonctions associées à ces ports en utilisant le site web donné plus haut. Vérifiez, pour la machine 192.168.80.200 que les services qui tournent sont bien ceux que vous avez identifié. Quel service/fonctionnalité avez-vous découvert ?

Question 4-2

Il y a une imprimantes dans votre réseau. Les postes communiquent avec l'imprimante en se connectant généralement sur les ports printer spooler ou jetdirect. Recherchez à quels numéros correspondent ces services, puis repérez l'imprimante en recherchant ces ports ouverts.

Attention de ne pas faire un scan «sauvage» de tout le réseau. Demandez avant de lancer votre commande.

Question 4-3

Nous venons de voir que le programme PuTTY permet de se connecter en ssh sur une machine distante: Essayez de vous connecter en ssh sur les machines suivantes :

- 192.168.80.123
- 192.168.80.200
- 10.16.83.102

Qu'est ce qui marche/ne marche pas et pourquoi selon vous ?

5 Protocole HTTP

Le protocole HTTP est celui utilisé sur le web. Il permet de récupérer le contenu d'URL. Une page Web ordinaire est un texte. Dans ce texte peuvent figurer des références à des images, des sons, des animations Flash, des applets Java... Dans un navigateur Web, entrez l'URL suivante : <u>http://deptinfo-ensip.univ-poitiers.fr/demo/page2.html</u>

La page qui apparaît contient texte et image. Vous pouvez voir ce qui a été réceptionné par votre navigateur en affichant le *source* de la page (du texte, balisé en HTML)

Question 5-1

À votre avis, combien de requêtes HTTP (GET) le navigateur a-t-il envoyé au serveur (et pour obtenir quoi ?) lors de l'affichage de la page http://deptinfo-ensip.univ-poitiers.fr/demo/page2.html pour la première fois ? Vous pouvez le déduire de la page ou le mesurer avec Wireshark. Attention toutefois à prendre en compte le fait que le navigateur a un *cache*.

Nous allons maintenant *simuler* un navigateur Web et nous mettre à la place d'un client HTTP. Le programme PuTTY (Windows) ou la commande telnet (Linux) permettent de se connecter en mode texte vers une machine distante. Connectez vous sur le port 80 de deptinfo-ensip.univ-poitiers.fr en utilisant telnet :

telnet deptinfo-ensip.univ-poitiers.fr 80

ou bien en utilisant Putty (mieux) (connexion : Raw, Close Windows on exit : never). Rien ne se passe... À présent entrez les lignes suivantes (sans fautes):

GET /demo/page2.html HTTP/1.0

puis validez deux fois... Le serveur Web répond et ce que vous voyez est bien le source de la page tel qu'il était dans le navigateur.

Question 5-2 Expliquez la ligne que vous avez entré :

GET ...

Question 5-3 Comment faire pour récupérer, par le même principe, l'image qui figure dans la page ? Faites le et expliquez ce que vous obtenez.

Lorsque le serveur Web répond, il donne, en tout début de réponse, des informations supplémentaires (qu'on ne retrouve pas dans le source de la page et qui constituent le *header* HTTP).

Question 5-4 Quel logiciel (et en quelle version) est utilisé actuellement comme serveur Web sur deptinfo-ensip.univ-poitiers.fr. Quelle est la taille en octets de l'image que vous avez essayé de récupérer ?

Question 5-5 Essayez de déterminer le type de logiciel serveur qui sert les pages de <u>http://www.univ-poitiers.fr</u> et de <u>http://ensip.univ-poitiers.fr</u>

Question 5.6 (ex 6-2) Dans sa requêtes, votre navigateur transmet des informations sur sa nature. Quelles sont ces informations ? Donnez aussi une copie d'écran qui permet de vérifier ce que vous avancez.

Question 5.7 (ex 6-4) À présent, capturez les trames lorsque vous consultez la page : <u>https://deptinfo-ensip.univ-poitiers.fr/</u> Quelles informations arrivez-vous à tirer de la capture de trame ? Expliquez.

6 Trames TCP

Les premières applications que nous avons observées à bas niveau n'expédiaient qu'une seule trame... Qu'en est-il des applications réseau plus conviviales ? Ce qui suit a été obtenu de la façon suivante : Une première machine (10.16.90.70) utilise un navigateur Web pour regarder le contenu de la page (qui n'existe plus maintenant) http://wwwesip2.univ-

poitiers.fr/accueil.php. Une autre machine, non loin de là, utilise Wireshark et capture toutes les trames dont la source ou la destination est 10.16.90.70 :

<u>F</u> ile	<u>E</u> dit <u>C</u> a	apture <u>D</u> isplay	<u>T</u> ools		<u>H</u> elp
No.	Time	Source	Destination	Protocol	Info
1	0.000000	10.16.90.70	194.254.43.241	ТСР	56198 > 80 [SYN] Seg=57887727 Ack=0 W
2	0.000208	194.254.43.241	10.16.90.70	тср	80 > 56198 [SYN, ACK] Seq=3833848996
3	0.000231	10.16.90.70	194.254.43.241	тср	56198 > 80 [ACK] Seq=57887728 Ack=383
4	0.000303	10.16.90.70	194.254.43.241	HTTP	GET /accueil.php HTTP/1.1\r\n
5	0.000661	194.254.43.241	10.16.90.70	тср	80 > 56198 [ACK] Seq=3833848997 Ack=5
6	0.003354	194.254.43.241	10.16.90.70	HTTP	HTTP/1.1 200 OK\r\n
7	0.003391	10.16.90.70	194.254.43.241	тср	56198 > 80 [ACK] Seq=57888233 Ack=383
8	0.023443	10.16.90.70	194.254.43.241	HTTP	GET /style.css HTTP/1.1\r\n
9	0.024396	194.254.43.241	10.16.90.70	HTTP	HTTP/1.1 200 OK\r\n
10	0.042586	10.16.90.70	194.254.43.241	HTTP	GET /images/batiment.png HTTP/1.1\r\n
11	0.043743	194.254.43.241	10.16.90.70	HTTP	HTTP/1.1 200 OK\r\n
12	0.043858	194.254.43.241	10.16.90.70	HTTP	Continuation
13	0.043875	10.16.90.70	194.254.43.241	тср	56198 > 80 [ACK] Seq=57889171 Ack=383
14	0.043980	194.254.43.241	10.16.90.70	HTTP	Continuation
15	0.044093	10.16.90.70	194.254.43.241	тср	56199 > 80 [SYN] Seq=64457318 Ack=0 W
16	0.044566	194.254.43.241	10.16.90.70	HTTP	Continuation
17	0.044591	10.16.90.70	194.254.43.241	тср	56198 > 80 [ACK] Seq=57889171 Ack=383
18	0.044683	194.254.43.241	10.16.90.70	HTTP	Continuation
<u>10</u>	0 044000	104 354 43 344	40 40 00 70	11776	Continuation
<u> -</u>					×.
E Er	ome 1 (74	1 on wire 74 c	antured)		21
	hornot Ti	т он тне, тт с Г	apculeu,		1
	tornot D	rotocol			
	iternet Pi				IZ
0000	00 e0 b	1 48 96 ff 00 c	0 b7 0a a8 19 08 00 45 00	H	E. 14
0010	00 3c 9	3 81 40 00 40 0	6 53 f5 0a 10 5a 46 c2 fe	.<.@.@.	SZF
0020	2b f1 d	b 86 00 50 03 7	3 4b ef 00 00 00 00 a0 02	+P.s	К
0030	16 d0 d	6 a5 00 00 02 0	4 05 b4 04 02 08 0a 03 a4		
0040	ав ье О	0 00 00 00 01 U	3 03 00	·n	
		Filter:			Reset File: capture Drops: 0

Nous avons capturé... 70 trames pour un simple affichage de page web.... En effet, une seule trame ne peut pas contenir toutes ces informations. C'est là qu'intervient le protocole TCP.... Le navigateur web (application) est basé sur le protocole HTTP (HyperText Transfert Protocol), lui même basé sur TCP (Transmission Control Protocol), lui même basé sur IP (Internet Protocol)... Le but de TCP est d'établir une connexion (TCP est un protocole orienté connexion) "courtoise" avec une machine distante, de découper les informations provenant du protocole supérieur en morceaux, qu'il donnera à IP pour qu'il les transmette. De plus, TCP vérifie que chaque morceau a été correctement transmis et reçu. Il s'occupe aussi de remettre les segments dans l'ordre lors d'une réception (en effet, les informations, étant transmises «par morceaux», peuvent ne pas suivre la même route et il se peut que leur ordre d'arrivée ne soit pas leur ordre de départ....).

Après avoir vidé le cache de votre navigateur, lancez Wireshark, réglez correctement les filtres et

consultez la page suivante ; <u>http://deptinfo-ensip.univ-poitiers.fr/demo/champernowne/index.html</u>. Vous allez devoir analyser le résultat obtenu dans Wireshark.

Question 6-1 Repérez un *three way handshake* TCP, mettez le en surbrillance et donnez une copie d'écran. Durant cet échange, client et serveur synchronisent leurs numéros de séquence respectifs. Indiquez les numéros de séquence initiaux *réels* du client et du serveur. Faites en sorte qu'on puisse vérifier votre résultat à partir de copies d'écran que vous fournirez.

Question 6-2 Donnez une copie d'écran qui montre (surbrillance) la fin d'une connexion TCP.

7 Résolution de noms

Lorsque vous consultez une page sur deptinfo-ensip.univ-poitiers.fr avec un navigateur Web, les datagrammes IP émis ne contiennent pas le nom de la machine, mais son adresse IP.

Question 7-1 Quelle est l'adresse IP de deptinfo-ensip.univ-poitiers.fr? Comment votre navigateur l'a-t-il obtenue ? Donnez des copies d'écran montrant ce mécanisme.

Vous pouvez voir le serveur DNS que votre machine utilise en entrant ipconfig /all (Windows) ou cat /etc/resolv.conf (Linux).

Notons que le protocole DNS fait partie de la couche application et qu'il n'est pas basé sur TCP comme HTTP, mais sur UDP. UDP a la même fonction que TCP mais ne s'occupe pas, en particulier, de vérifier que la transmission s'est faite correctement ou que tout ce qui a été envoyé a été reçu.

Question 7-2 Identifiez votre DNS par défaut (copie d'écran) et réalisez une capture d'une requête. Vous prendrez comme exemple une requete concernant www.google.fr. Analysez ensuite la reponse du DNS concernant www.google.fr.www.google.fr a-t-il d'autres noms ? Il semble avoir plusieurs adresses IP... Essayez de pinguer www.google.fr plusieurs fois. Que constatez vous ? Expliquez.

8 Bonus

Question 8-1

Indiquez à l'encadrant que vous en êtes à la question Bonus. Essayez de consulter quelques pages Webs, puis déterminez pourquoi votre machine ne fonctionne plus correctement.

9 Archives

Putty

Pour scanner les ports de nos machines, vous devrez vous connecter sur une machine Linux (192.168.81.21 en SSH) en utilisant le logiciel PuTTY. Puis vous vous connecterez à la machine Linux (les identifiants (*username* et *password*) vous seront données en TP) et utiliserez le logiciel nmap.



Lorsque vous êtes dans la fenêtre de terminal de Putty (la fenêtre noire), vous pouvez, sous Windows, y copier ce qui est dans le presse-papier. Pour cela, il sufit de cliquer sur le bouton droit de la souris.