Institut Universitaire de Technologie
Aix*Marseille Université
Département Informatique, Aix

C. Pain-Barre

UE31 - M3102 : Services Réseaux

Corrigé du TP 2

Fragmentation/Réassemblage de datagrammes IP

-

Analyse/Production de trames

Table des matières

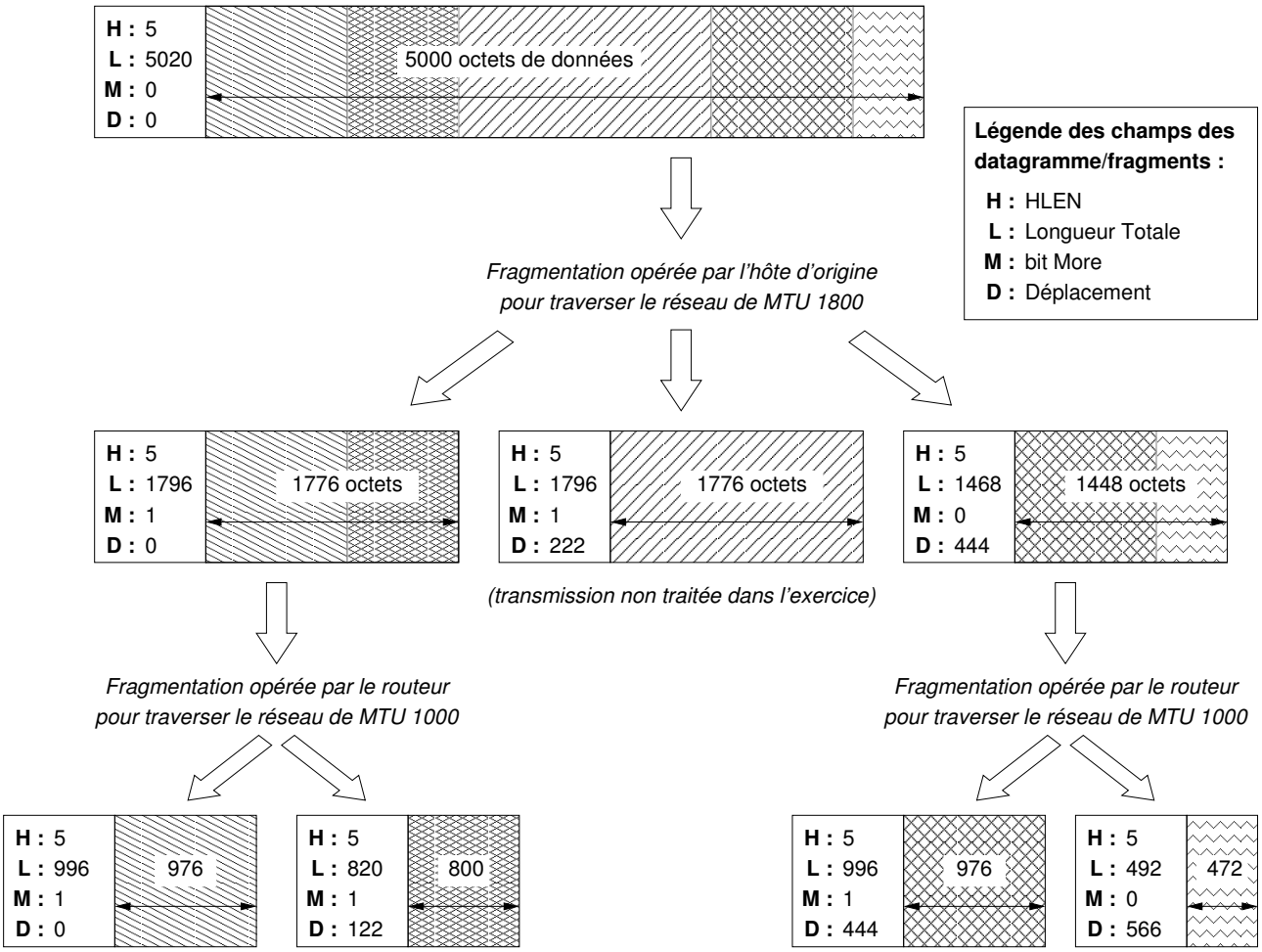


FIGURE 1 – *Fragmentations du datagramme de l'exercice 1*

1 Fragmentation et réassemblage de datagrammes IPv4

1.1 Technique de fragmentation

Corrigé de l'exercice 1 (Fragmentation de datagrammes)

[Consulter l'énoncé]

Les fragmentations sont illustrées par la figure 1. En haut, le datagramme d'origine se trouve fragmenté en 3 fragments par l'hôte émetteur. Ces fragments suivent chacun une route propre, éventuellement différente. Le premier et le dernier ont été reçus par un routeur (pas forcément le même) qui les fragmente aussi.

1.2 Technique de réassemblage

Corrigé de l'exercice 2 (Réassemblage des fragments de datagrammes)

[Consulter l'énoncé]

1. On s'aperçoit en effet qu'ils ont tous soit le *bit More* à 1, soit un champ *Déplacement* non nul

2. Il faut regrouper les datagrammes qui possèdent le même quadruplet (Protocole, Identification, Adresse IP Source, Adresse IP Destination). Il y a trois datagrammes d'origine :

   - datagramme origine 1 : `datagramme_01.txt`, `datagramme_06.txt` et `datagramme_10.txt`
   - datagramme origine 2 : `datagramme_02.txt`, `datagramme_05.txt`, `datagramme_11.txt` et `datagramme_12.txt`
   - datagramme origine 3 : `datagramme_03.txt`, `datagramme_04.txt`, `datagramme_07.txt`, `datagramme_08.txt` et `datagramme_09.txt`

   Ensuite, il faut ordonner les fragments de chaque datagramme d'origine en fonction du champ Déplacement, vérifier qu'il y a bien le premier (Déplacement à 0) et le dernier fragment (bit More à 0) et qu'il ne manque aucun fragment en se basant sur le champ Déplacement et la taille des données contenues dans les fragments :

   - datagramme origine 1 :
     l'ordre est `datagramme_10.txt`, `datagramme_01.txt` puis `datagramme_06.txt`.
     `datagramme_10.txt` est bien le premier fragment et `datagramme_06.txt` est le dernier. De plus, il n'y a pas de trous car $300 = 2400/8$ et $600 = 300 + (2400/8)$. Tous les fragments de ce datagramme ont donc été reçus et son en-tête est :

HLEN	LT	Proto	Ident	More	Dépl.	IP Source	IP Dest.
5	6180	17	12345	0	0	139.124.187.2	139.124.187.4

     où sa *Longueur Totale* est calculée comme $20 + 2400 + 2400 + 1360$
   - datagramme origine 2 :
     l'ordre est `datagramme_12.txt`, `datagramme_02.txt`, `datagramme_11.txt` puis `datagramme_05.txt`.
     `datagramme_12.txt` est bien le premier fragment et `datagramme_05.txt` est le dernier. Cependant, il manque au moins un fragment car `datagramme_11.txt` a un Déplacement valant 900 et une longueur de données de 2400, et $900 + (2400/8) \neq 1800$ alors que 1800 est le Déplacement annoncé par `datagramme_05.txt`. On ne peut donc reconstituer ce datagramme.
   - datagramme origine 3 :
     l'ordre est `datagramme_08.txt`, `datagramme_03.txt`, `datagramme_04.txt`, `datagramme_09.txt` puis `datagramme_07.txt`.
     `datagramme_08.txt` est bien le premier fragment et `datagramme_07.txt` est le dernier. De plus, il n'y a pas de trous car $300 = 2400/8$, $600 = 300 + 2400/8$, $900 = 600 + 2400/8$ et $1200 = 900 + 2400/8$. L'en-tête de ce datagramme entièrement reconstitué est :

HLEN	LT	Proto	Ident	More	Dépl.	IP Source	IP Dest.
5	9690	6	12345	0	0	158.159.160.161	139.124.187.4

     où sa *Longueur Totale* est calculée comme $20 + 2400 + 2400 + 2400 + 2400 + 70$

3. Il n'y a pas de corrigé pour cette question.

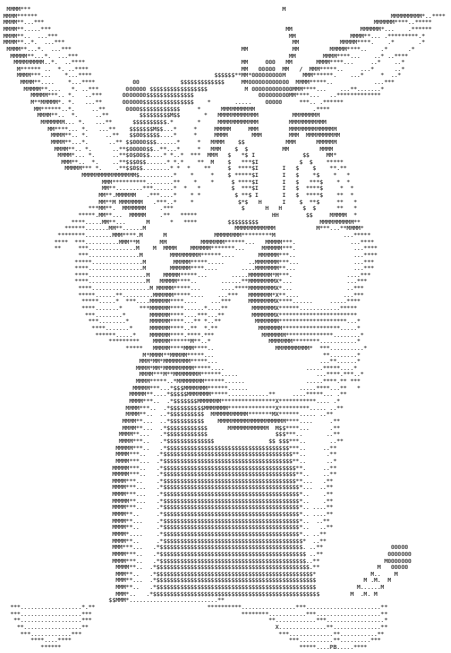
4. Visualisation des images contenues dans les datagrammes d'origine :
   - datagramme origine 1 : l'image représente une chanteuse. Pour l'afficher, il faut taper (en adaptant le chemin de **dataglue** et des datagrammes) :

     $ **dataglue datagramme_10.txt datagramme_01.txt datagramme_06.txt**

➪ ce sont bien des caractères ASCII, mais dans une très petite taille et en reverse vidéo.

- datagramme origine 2 : incomplet mais si on tente de visualiser ses données partielles, on obtient :

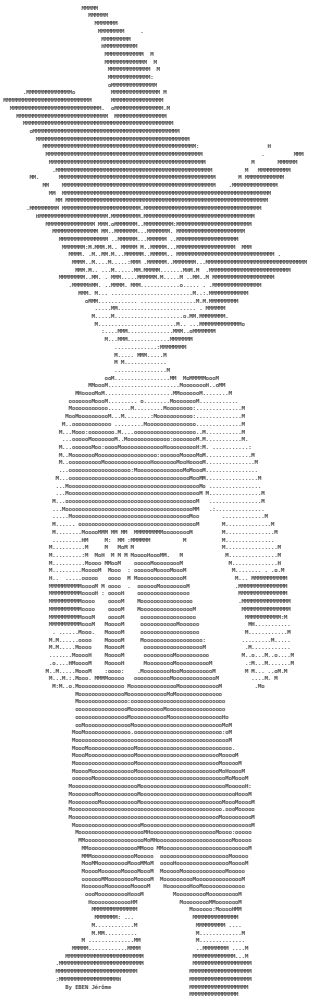
```
$ dataglue datagramme_12.txt datagramme_02.txt datagramme_11.txt \
          datagramme_05.txt
```



➪ Il semble bien qu'il manque quelque chose ! Manifestement, ce datagramme est bien incomplet...

- datagramme origine 3 :

```
$ dataglue  datagramme_08.txt datagramme_03.txt datagramme_04.txt \
          datagramme_09.txt datagramme_07.txt
```



2 Analyse et production de trames

2.1 Analyse de trames

Corrigé de l'exercice 3 (Téléchargement des fichiers de trames)

[Consulter l'énoncé]

Il n'y pas de corrigé pour cet exercice.

Corrigé de l'exercice 4 (Analyse (automatisée) de la première trame)

[Consulter l'énoncé]

1. *... pas de corrigé pour cette question...*
2. *... pas de corrigé pour cette question...*
3. Il s'agit d'une **requête ARP** émise en diffusion par l'hôte d'adresse IP `125.5.48.10` via sa carte Ethernet d'adresse MAC `09:ab:14:d8:05:48`. La requête demande quelle est l'adresse MAC de l'hôte ayant l'adresse IP `125.18.110.3`.
4. Comme le fait remarquer **Wireshark**, il y a un loup car cette trame ne devrait pas avoir une adresse multicast comme adresse source !
   À part ça, tout le reste semble correct : tous les champs du datagramme ARP sont cohérents et conformes à une requête ARP, et la trame qui l'encapsule a bien été émise en diffusion à partir de la carte qui correspond à l'adresse physique source mentionnée dans le datagramme ARP.

Corrigé de l'exercice 5 (Analyse automatisée de la deuxième trame)

[Consulter l'énoncé]

1. *. . . pas de corrigé pour cette question. . .*
2. Le protocole de plus haut niveau est ICMP. La figure 2 montre l'affichage détaillé du message ICMP par **Wireshark**. Il s'agit d'un message de type ECHO-REPLY envoyé par l'hôte `139.124.187.4`, normalement en réponse à une demande d'ECHO (message ICMP ECHO-REQUEST) qu'a faite la station `172.16.203.109`.
   Le Numéro de séquence laisse à penser qu'il y a déjà eu auparavant une demande (et, s'il n'y a pas eu d'erreur, une réponse). Il s'agit probablement d'une trame appartenant à un trafic provoqué par la commande **ping**.
3. Les hôtes impliqués dans ce dialogue ICMP sont `172.16.203.109` et `139.124.187.4`. Ils ne se trouvent manifestement pas sur le même réseau et la trame a été transmise par un routeur (le dernier traversé par le datagramme). C'est donc l'adresse MAC de l'interface de ce routeur sur le réseau de `172.16.203.109`.

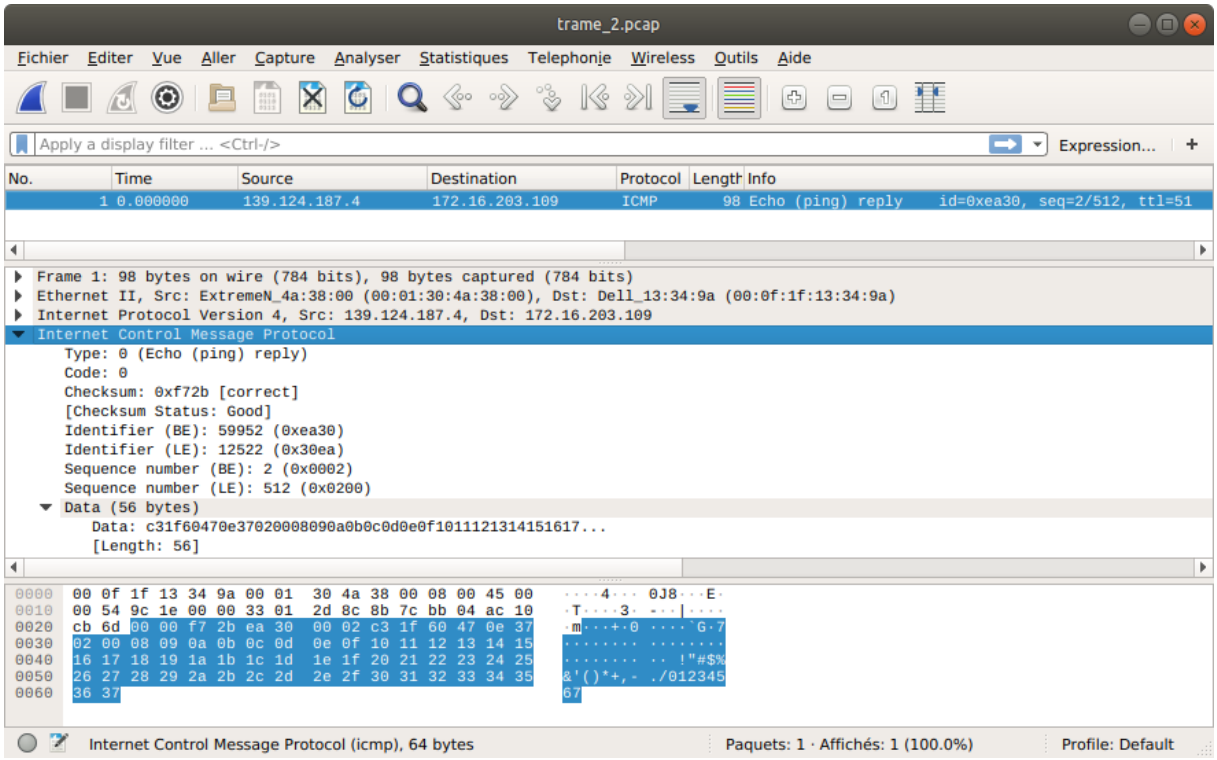


FIGURE 2 – *Vue détaillée de la trame 2 et des champs du message ICMP* ECHO-REPLY *dans* **Wireshark**.

2.2 Encapsulation de messages et production de trames

Corrigé de l'exercice 6 (Production automatisée de la réponse à la première trame)

[Consulter l'énoncé]

La figure 3 présente la valeur des champs pour cette réponse ARP. La figure 4 montre les octets générés pour cette trame après avoir cliqué sur *Gen-b*.

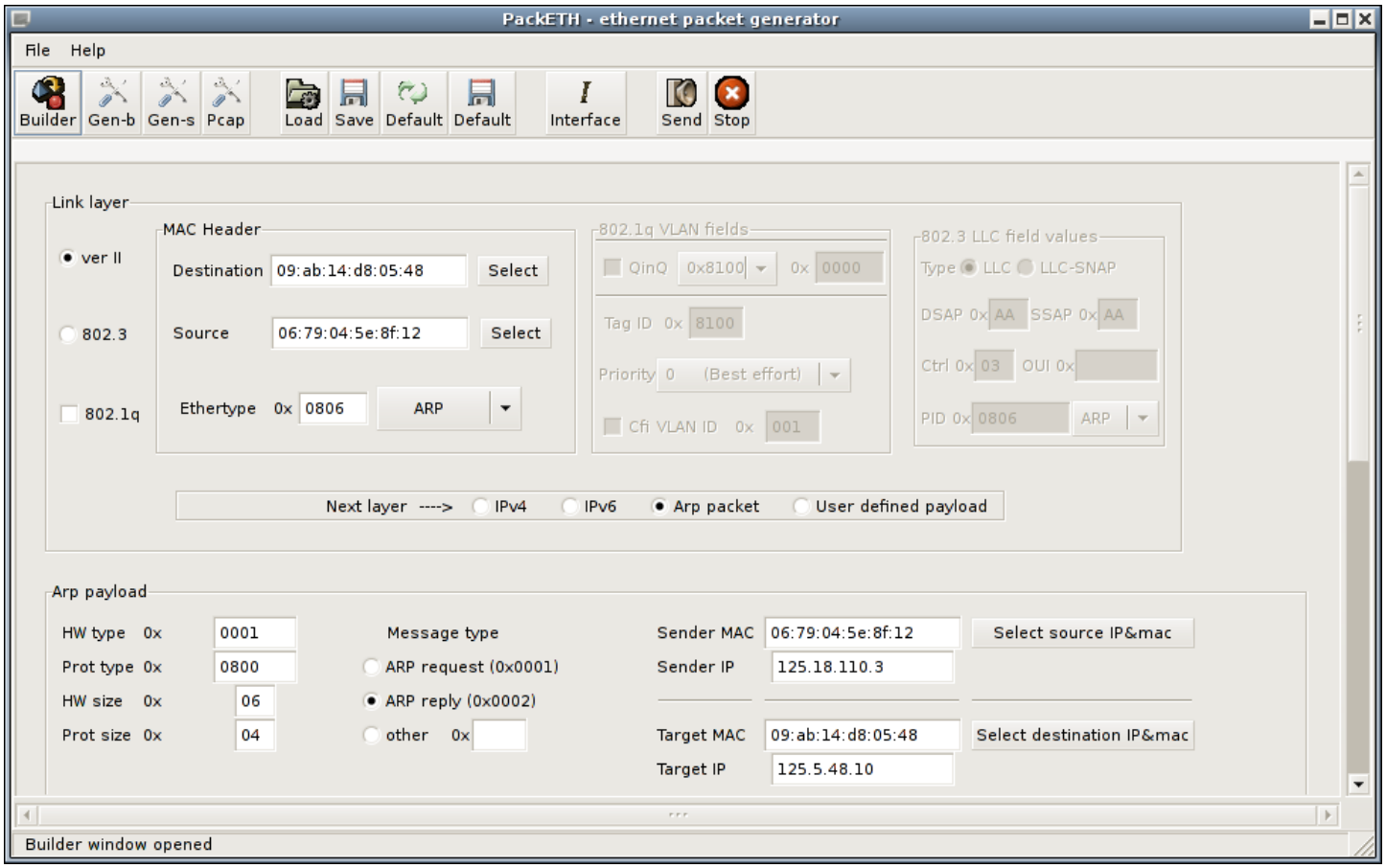


FIGURE 3 – *Saisie de la réponse ARP dans* **packeth** *pour l'exercice 6*

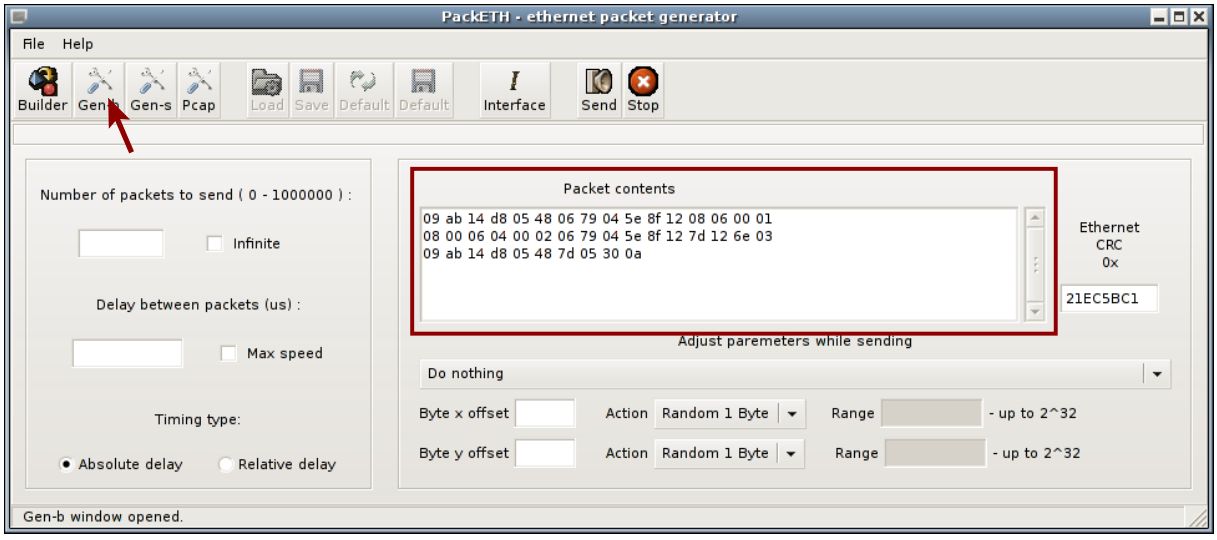


FIGURE 4 – *Génération des octets de la trame de la réponse ARP avec le bouton* `Gen-b` *de* **packeth**

Corrigé de l'exercice 7 (Deuxième trame à produire automatiquement)

[Consulter l'énoncé]

La figure 5 présente les informations à remplir pour produire la trame avec **packeth**.

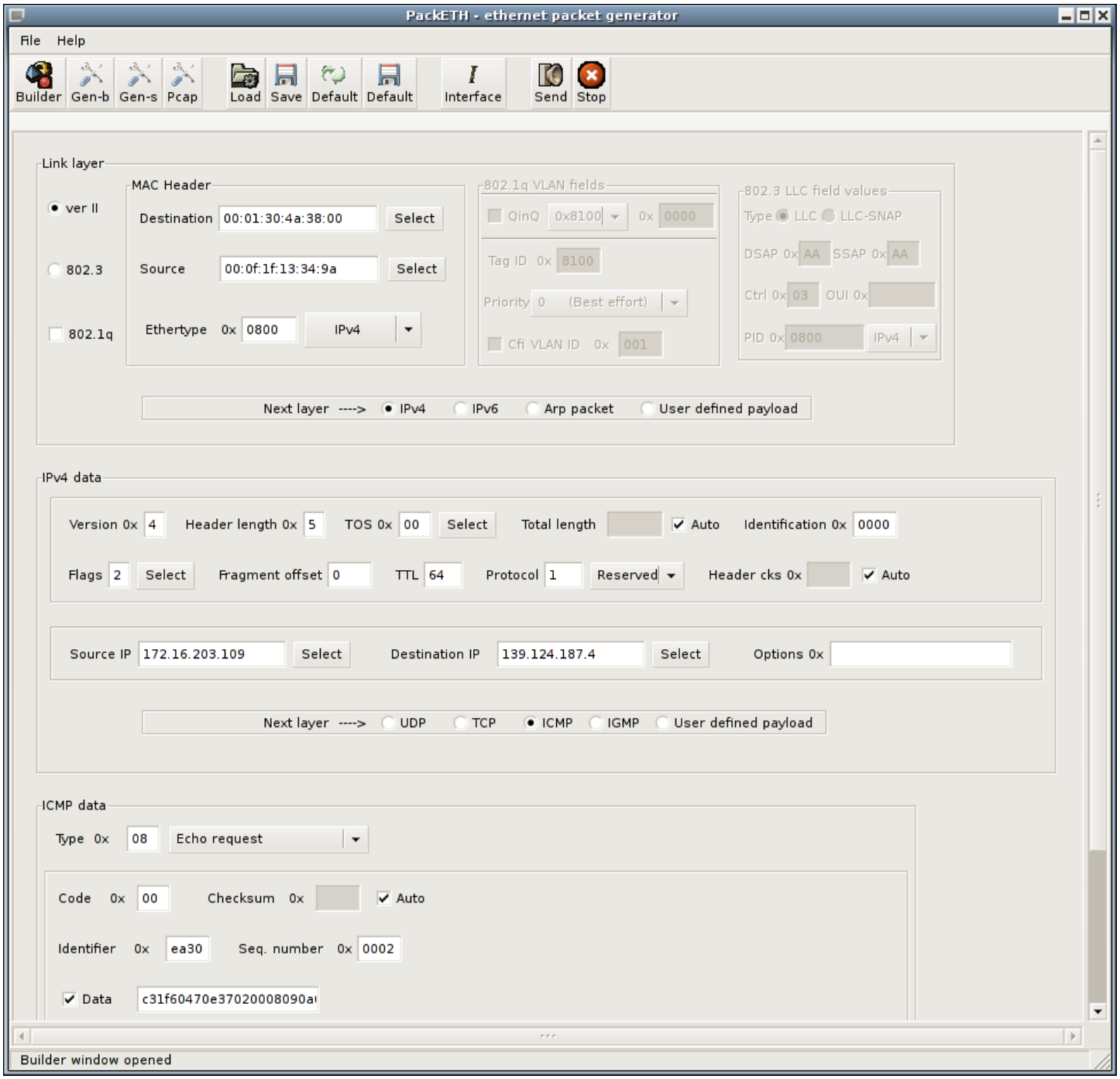


FIGURE 5 – *Production de la trame 2 (ICMP* ECHO REQUEST*) avec* **packeth**

2.3 Exercices complémentaires

2.3.1 Analyses manuelles de trames

Corrigé de l'exercice 8 (Analyse manuelle de la première trame capturée)

[Consulter l'énoncé]

1. On commence par extraire les champs de la trame Ethernet et le champ *EtherType* indique le type de PDU contenu dans les *Données* :
   - **Trame Ethernet :**

     On commence par extraire les champs de la trame *Ethernet*, cela donne :

Champs de la trame Ethernet	
Adresse Ethernet Destination	`ff:ff:ff:ff:ff:ff` (diffusion Ethernet)
Adresse Ethernet Source	`09:ab:14:d8:05:48`
EtherType	0x`0806` (ARP)
Données	`0001 0800 0604 0001 09ab 14d8 0548 7d05 300a 0000 0000 0000 7d12 6e03`

     ➪ Le champ EtherType indique que les données forment un datagramme ARP.
     La trame a été émise en **diffusion**. Toutes les machines du réseau vont la recevoir et la traiter.

   - **Datagramme ARP :**

     On extrait les champs du datagramme ARP à partir des données *Ethernet* :

Champs du datagramme ARP	
Type de réseau	`1` (donc Ethernet)
Protocole	0x`0800` (donc IP)
L. @phy	`6` (octets pour adresses Ethernet)
L. @pro	`4` (octets pour adresses IP)
Opération	`1` (donc requête ARP)
Adresse physique source	`09:ab:14:d8:05:48`
Adresse proto source	`125.5.48.10`
Adresse physique destination	`00:00:00:00:00:00` (inconnue)
Adresse proto destination	`125.18.110.3`

2. Il s'agit d'une **requête ARP** émise en diffusion par l'hôte d'adresse IP `125.5.48.10` via sa carte Ethernet d'adresse MAC `09:ab:14:d8:05:48`. La requête demande quelle est l'adresse MAC de l'hôte ayant l'adresse IP `125.18.110.3`.
   Tout semble correct : tous les champs du datagramme ARP sont cohérents et conformes à une requête ARP, et la trame qui l'encapsule a bien été émise en diffusion à partir de la carte qui correspond à l'adresse physique source mentionnée dans le datagramme ARP.

3. Le champ *Données* ne faisant que 28 octets, il manque le champ *Remplissage* (*Padding*) qui devait faire 18 octets, afin d'arriver aux 64 octets de la trame minimale. Ce champ ne comporte rien d'intéressant, et n'est généralement pas conservé par les utilitaires qui capturent les trames.

Corrigé de l'exercice 9 (Analyse d'une capture)

[Consulter l'énoncé]

1. . . . *pas de corrigé pour cette question. . .*
2. Il faut utiliser le filtre d'affichage : **`tcp.port == 21`**
3. L'établissement de la connexion se fait avec des messages TCP ayant le bit SYN positionné. On peut ensuite vérifier les numéros de séquence des données échangées et les accusés de réception correspondants. On peut remarquer que tous les messages applicatifs de FTP (commandes FTP et réponses FTP) sont émis avec le bit PUSH positionné. Enfin, la connexion est terminée lorsque les deux côtés ont envoyé (et acquitté la réception d') un segment avec le bit FIN positionné.
4. On voit clairement que le nom de l'utilisateur qui veut se connecter au serveur FTP ainsi que son mot de passe sont transmis sans cryptage. Un utilisateur malveillant (ou *attaquant*) qui se trouve sur le trajet du datagramme IP et qui peut l'intercepter (ou écouter une liaison sur laquelle il est transmis), obtiendra donc facilement ces informations. Si en plus cet utilisateur utilise sur plusieurs stations le même couple login/mot de passe, alors l'attaquant pourra facilement s'y connecter. . .

2.3.2 Productions manuelles de trames

Corrigé de l'exercice 10 (Première trame à produire manuellement)

[Consulter l'énoncé]

Il s'agit donc de la réponse ARP émise par la machine d'adresse IP `125.18.110.3` pour indiquer que l'adresse de sa carte Ethernet associée à `125.18.110.3` est `06:79:04:5e:8f:12`. Cette réponse est envoyée **en unicast** à la carte `09:ab:14:d8:05:48` de la machine `125.5.48.10`, qui avait posé la question.

Pour fabriquer la trame, on commence par fabriquer le datagramme ARP :

- **Datagramme ARP :**

Champs du datagramme ARP	
Type de réseau	0x**0001** (Ethernet)
Protocole	0x**0800** (IP)
L. @phy	0x**06** (6)
L. @pro	0x**04** (4)
Opération	0x**0002** (réponse ARP)
Adresse physique source	0x**0679045e8f12** (06:79:04:5e:8f:12)
Adresse proto source	0x**7d126e03** (125.18.110.3)
Adresse physique destination	0x**09ab14d80548** (09:ab:14:d8:05:48)
Adresse proto destination	0x**7d05300a** (125.5.48.10)

➪ Le datagramme ARP, de 28 octets, est finalement :

```
0001 0800 0604 0002 0679 045e 8f12
7d12 6e03 09ab 14d8 0548 7d05 300a
```

- **Trame Ethernet :**

Les champs de la trame Ethernet véhiculant ce datagramme sont :

Champs de la trame Ethernet	
Adresse Ethernet Destination	0x**09ab14d80548** (09:ab:14:d8:05:48)
Adresse Ethernet Source	0x**0679045e8f12** (06:79:04:5e:8f:12)
EtherType	0x**0806** (ARP)
Données	0001 0800 0604 0002 0679 045e 8f12 7d12 6e03 09ab 14d8 0548 7d05 300a

➪ La trame Ethernet, de 42 octets, est finalement :

```
09ab 14d8 0548 0679 045e 8f12 0806 0001
0800 0604 0002 0679 045e 8f12 7d12 6e03
09ab 14d8 0548 7d05 300a
```

ⓘ Cette trame ne fait que 42 octets. Or la taille minimale d'une trame Ethernet doit être de 64 octets, en comptant le CRC, mais pas le Préambule.
Il manque donc $64 - 42 - 4 = 18$ octets de bourrage, qu'il faut ajouter aux précédents (champ *Remplissage*), suivis du CRC.

Corrigé de l'exercice 11 (Production manuelle de la deuxième trame)

[Consulter l'énoncé]

Pour fabriquer la trame, on commence par fabriquer le message ICMP puis le datagramme IP qui l'encapsule :

- **Message ICMP :**

Champs du message ICMP	
Type	0x**08** (demande d'ECHO)
Code	0x**00**
Total de Contrôle)	0x**ef2b**
Identificateur	0x**ea30**
Numéro de séquence	0x**0002**

Données	`c31f 6047 0e37 0200 0809 0a0b 0c0d 0e0f`
	`1011 1213 1415 1617 1819 1a1b 1c1d 1e1f`
	`2021 2223 2425 2627 2829 2a2b 2c2d 2e2f`
	`3031 3233 3435 3637`

➪ Le message ICMP, faisant au total 64 octets, est finalement :

```
0800 ef2b ea30 0002 c31f 6047 0e37 0200
0809 0a0b 0c0d 0e0f 1011 1213 1415 1617
1819 1a1b 1c1d 1e1f 2021 2223 2425 2627
2829 2a2b 2c2d 2e2f 3031 3233 3435 3637
```

- **Datagramme IP :**

Champs du datagramme IP	
Version	0x**4** (version 4)
IHL	0x**5** (20 octets d'en-tête car pas d'option)
Type of Service (TOS)	0x**00** soit 000 0 0 0 00 en binaire car : Priorité 000 (routine) bit D 0 car pas de souhait de faible délai bit T 0 car pas de souhait de gros débit bit R 0 car pas de souhait pour la fiabilité Réservé 00
Longueur Totale	0x**0054** (84 octets = 64 données + 20 en-tête)
Identification	0x**0000**
Partie Flags et Déplacement : 0x**4000** soit 0 1 0 0000000000000 en binaire car : Bit 0 0 Bit Don't Fragment 1 car le datagramme ne doit pas être fragmenté Bit More 0 car pas de fragment qui suit ce datagramme Déplacement (Offset) 0 car pas de fragment qui précède ce datagramme	
Time To Live	0x**40** (64)
Proto	0x**01** (ICMP)
Checksum	0x**7caa**
Adresse IP source	0x**ac10cb6d** (172.16.203.109)
Adresse IP destination	0x**8b7cbb04** (139.124.187.4)
Données	0800 ef2b ea30 0002 c31f 6047 0e37 0200 0809 0a0b 0c0d 0e0f 1011 1213 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 3637

➪ Le datagramme IP, de 84 octets, est le suivant :

```
4500 0054 0000 4000 4001 7caa ac10 cb6d
8b7c bb04 0800 ef2b ea30 0002 c31f 6047
0e37 0200 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
3435 3637
```

- **Trame Ethernet :**

Champs de la trame Ethernet	
Adresse Ethernet Destination	0x**0001304a3800** (00:01:30:4a:38:00)
Adresse Ethernet Source	0x**000f1f13349a** (00:0f:1f:13:34:9a)
EtherType	0x**0800** (IP)
Données	4500 0054 0000 4000 4001 7caa ac10 cb6d 8b7c bb04 0800 ef2b ea30 0002 c31f 6047 0e37 0200 0809 0a0b 0c0d 0e0f 1011 1213 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 3637

➪ La trame Ethernet, de 98 octets, est la suivante :

```
0001 304a 3800 000f 1f13 349a 0800 4500
0054 0000 4000 4001 7caa ac10 cb6d 8b7c
bb04 0800 ef2b ea30 0002 c31f 6047 0e37
0200 0809 0a0b 0c0d 0e0f 1011 1213 1415
1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
3637
```