

# Address Resolution Protocol (ARP)

**Réalisé par :** DIOURI Mehdi

**Encadré par :** Pr. Mallouli Wissam

## 1- Description du protocole :

Le protocole ARP a été défini afin d'obtenir la résolution des adresses IPv4 dans les adresses MAC. L'ARP permet la transmission de données dans les réseaux Ethernet car les trames individuelles d'un paquet IP ne peuvent être envoyées aux hôtes cible souhaités qu'à l'aide de l'adresse matérielle, et le protocole Internet ne peut se référer à ces adresses physiques indépendamment. D'autre part, le protocole IPv4 n'est pas en mesure de stocker les adresses des périphériques en raison de sa longueur limitée. (1)

## 2- Les champs protocolaires d'ARP :

Le protocole ARP utilise un format de message simple pour envoyer des requêtes ou des réponses. Bien qu'il soit à l'origine destiné aux adresses IPv4 et MAC, l'utilisation d'autres protocoles réseau est théoriquement possible, d'où l'existence de champs pour le type et la taille de l'adresse matérielle et du protocole. Le tableau suivant montre les composantes d'un tel paquet d'informations : (1)

	Bits 0 – 7	Bits 8 – 15	Bits 16 – 23	Bits 24 – 31		
0	Type de matériel ( <i>Hardware type</i> )		Type de protocole ( <i>Protocol type</i> )			
32	Longueur de l'adresse physique ( <i>Hardware Address Length</i> )	Longueur de l'adresse logique ( <i>Protocol Address Length</i> )	Operation			
64	Adresse physique de l'émetteur ( <i>Sender Hardware Address</i> ) – Adresse MAC source					
96						
112	Adresse réseau de l'émetteur ( <i>Sender Protocol Address</i> ) – Adresse IP de source					
144	Adresse physique du destinataire ( <i>Target Hardware Address</i> ) – Adresse MAC destination					
176						
192	Adresse réseau du destinataire ( <i>Target Protocol Address</i> ) – Adresse IP de destination					

*Figure 1: Composant d'un paquet ARP*

(1) <https://www.ionos.fr/digitalguide/serveur/know-how/protocole-de-resolution-dadresse-quest-ce-que-larp/>

### 3- Logiciel d'extraction des différents champs protocolaire d'ARP

#### *Démarches suivi pour la réalisation du projet :*

- Choix du protocole : ARP.
- Documentation sur le protocole et la constitution d'un paquet ARP.
- Téléchargement du fichier pcap source adéquat avec le protocole choisi.
- Choix du langage de programmation : Python
- Recherche de la bibliothèque permettant de lire le fichier : scapy.
- Lecture de la documentation de la bibliothèque scapy pour découvrir les exemples d'utilisation.
- Développement et debug de la fonction de parsing du fichier pcap.
- Conteneurisation du logiciel à l'aide de Docker
- Ajout d'une API pour faciliter l'utilisation du logiciel.
- Rédaction du manuel d'installation et d'utilisation.

#### *Etapas pour la compilation du logiciel :*

J'ai utilisé Docker pour permettre au logiciel d'avoir toutes les dépendances nécessaires pour l'exécution du code

- **Cloner le projet à partir du dépôt du git :**

**La commande :** `git clone https://github.com/mehdi-deve/Mini-projet-Not--DIOURI-MEHDI.git`

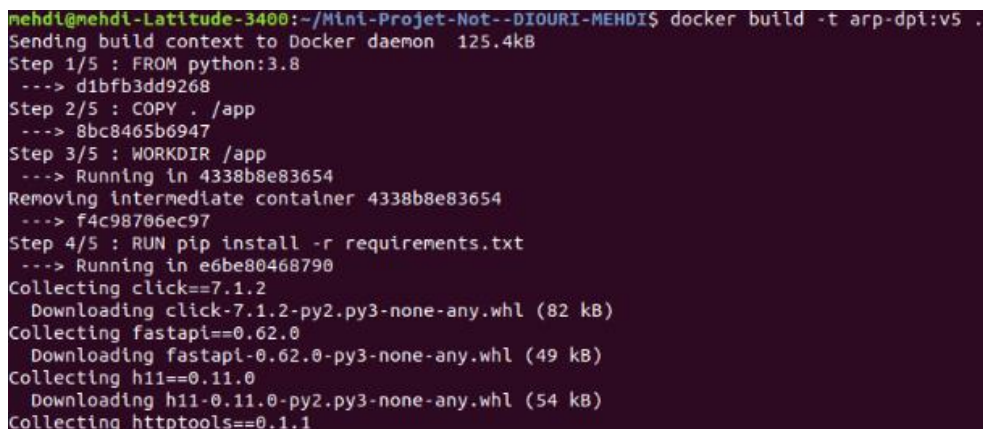


```
mehdi@mehdi-Latitude-3400:~$ git clone https://github.com/mehdi-deve/Mini-Projet-Not--DIOURI-MEHDI.git
Clonage dans 'Mini-Projet-Not--DIOURI-MEHDI'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Dépaquetage des objets: 100% (8/8), fait.
mehdi@mehdi-Latitude-3400:~$ cd Mini-Projet-Not--DIOURI-MEHDI/
mehdi@mehdi-Latitude-3400:~/Mini-Projet-Not--DIOURI-MEHDI$
```

Figure 2: Clone du dépôt

- **Builder l'image Docker à partir du Dockerfile en lui donnant le nom 'arp-dpi:v5'**

**La commande :** `docker build -t arp-dpi:v5 .`



```
mehdi@mehdi-Latitude-3400:~/Mini-Projet-Not--DIOURI-MEHDI$ docker build -t arp-dpi:v5 .
Sending build context to Docker daemon 125.4kB
Step 1/5 : FROM python:3.8
--> d1bfb3dd9268
Step 2/5 : COPY . /app
--> 8bc8465b6947
Step 3/5 : WORKDIR /app
--> Running in 4338b8e83654
Removing intermediate container 4338b8e83654
--> f4c98706ec97
Step 4/5 : RUN pip install -r requirements.txt
--> Running in e6be80468790
Collecting click==7.1.2
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting fastapi==0.62.0
  Downloading fastapi-0.62.0-py3-none-any.whl (49 kB)
Collecting h11==0.11.0
  Downloading h11-0.11.0-py2.py3-none-any.whl (54 kB)
Collecting httpxtools==0.1.1
```

Figure 3: Build de l'image Docker

```
Successfully built python-multipart PyYAML scapy
Installing collected packages: starlette, six, pydantic, h11, click, websockets, watchdog, uvloop, uvicorn, typing-extensions, scapy, PyYAML, p
Successfully installed PyYAML-5.3.1 click-7.1.2 fastapi-0.62.0 h11-0.11.0 httptools-0.1.1 pydantic-1.7.3 python-dotenv-0.15.0 python-multipart-
ns-3.7.4.3 uvicorn-0.13.1 uvloop-0.14.0 watchdog-0.6 websockets-8.1
Removing intermediate container e6be80468790
--> 554f0ecb6ce1
Step 5/5 : CMD ["python","-u","main.py"]
--> Running in 008509aae707
Removing intermediate container 008509aae707
--> 3a2c83bbda6
Successfully built 3a2c83bbda6
Successfully tagged arp-dpi:v5
```

Figure 4: Confirmation du build de l'image Docker

- **Lancer le container :**

**La commande :** `docker run -p 80:5001 arp-dpi:v5`

```
mehdi@mehdi-Latitude-3400:~/Mini-Projet-Not--DIOURI-MEHDIS$ docker run -p 80:5001 arp-dpi:v5
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:5001 (Press CTRL+C to quit)
```

Figure 5: Lancement du container

- **Accéder à l'adresse <http://0.0.0.0:80> à partir d'un navigateur web :**

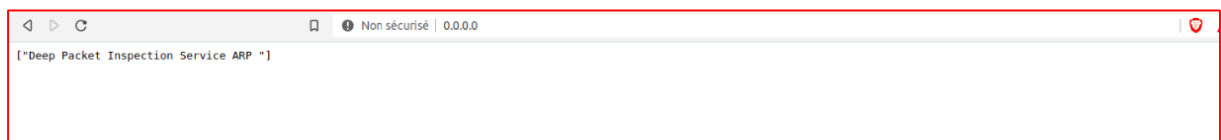


Figure 6: Première interface de l'API

- **Accéder à l'adresse <http://0.0.0.0:80/docs> à partir d'un navigateur web :**

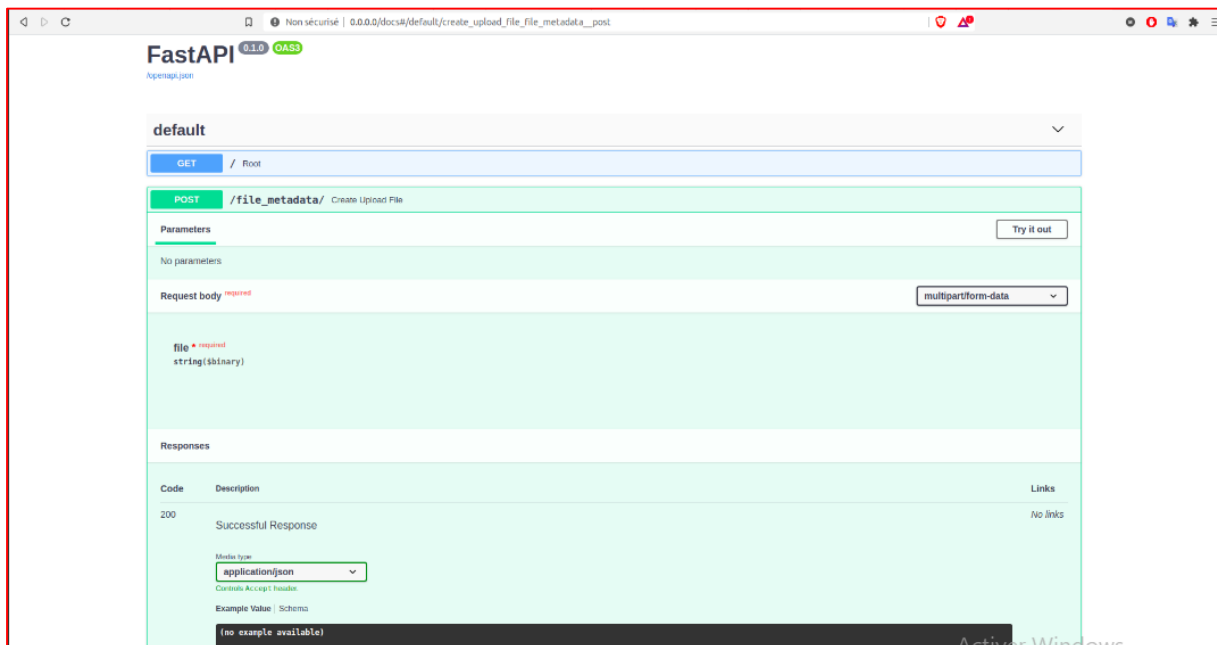


Figure 7: Interface de l'API

- Cliquer sur ‘Try it out’ pour charger le fichier pcap présent dans le dépôt du git puis cliquer sur Exécuter

FastAPI 0.1.0 OAS3

default

GET / Root

POST /file\_metadata/ Create Upload File

Parameters

No parameters

Request body **required** multipart/form-data

file **required** string(\$binary) Choisir un fichier Aucun fichier choisi

Execute

Responses

Activer Windows  
Accédez aux paramètres pour activer Windows.

- Affichage des valeurs des différents champs du protocole :

file **required** string(\$binary) Choisir un fichier arp-storm.pcap

Execute Clear

Responses

Curl

```
curl -X POST "http://0.0.0.0/file_metadata/" -H "accept: application/json" -H "Content-Type: multipart/form-data" -F "file=@arp-storm.pcap;type=application/vnd.tcpdump.pcap"
```

Request URL

http://0.0.0.0/file\_metadata/

Server response

Code	Details
200	<p>Response body</p> <pre>{   "Ethernet": {     "dst": "ff:ff:ff:ff:ff:ff",     "src": "00:07:0d:af:f4:54",     "type": "ARP"   },   "ARP": {     "hwtype": "0x1",     "ptype": "IPv4",     "hlen": "0",     "plen": "4",     "op": "who-has",     "hwsrc": "00:07:0d:af:f4:54",     "psrc": "24.166.172.1",     "hwdst": "00:00:00:00:00:00",     "pdst": "24.166.173.159"   },   "Padding": {     "Load": ""   } }</pre>

Figure 8: L'affichage des différents champs du protocole ARP