# Protocolo de Resolución de Direcciones (ARP)

# Introducción

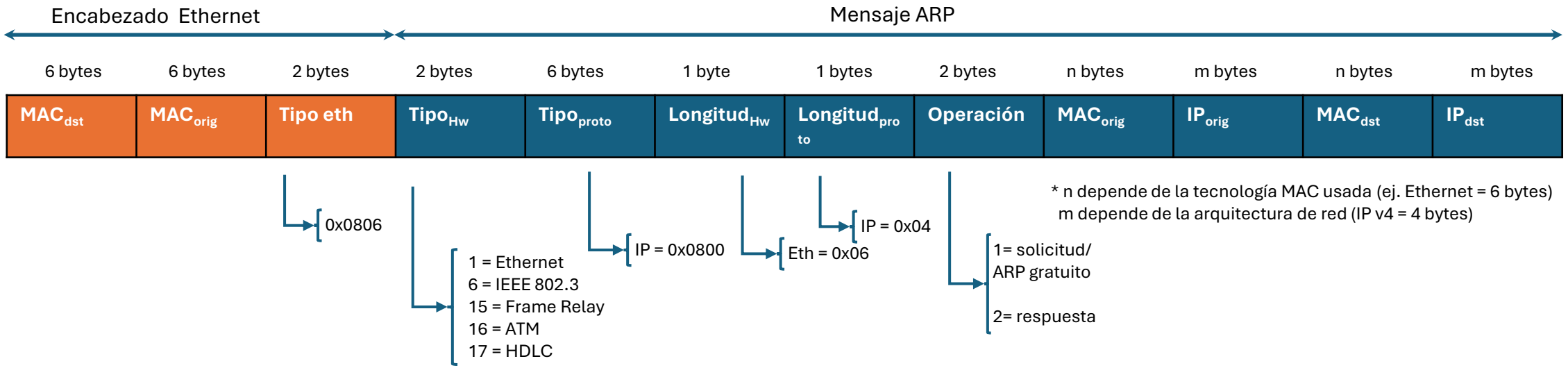
- Protocolo creado con la finalidad de traducir direcciones lógicas (direcciones IP) en direcciones físicas (direcciones MAC) en redes LAN.
- Una vez traducida una dirección IP en su correspondiente dirección MAC, ésta es almacenada en el caché de ARP, para que en el futuro cuando se desee transmitir una trama al mismo destinatario para el cual se hizo la resolución de dirección ya se cuente con la correspondiente dirección MAC sin tener que volver a solicitar la traducción de dirección.

# Introducción

- ARP fue publicado en el IETF como un estándar en 1982 en el RFC 826.
- Funciona sobre distintas tecnologías de control de acceso al medio como Ethernet, IEEE802.3, etc. Y el tamaño de mensaje de ARP depende de la tecnología de control de acceso al medio que se use (en Ethernet es de 30 bytes).
- Opera como uno de los servicios de la capa de red del modelo OSI (capa de Internet de TCP/IP) como el protocolo con identificador $(0x0806)_{16}$ o $(2054)_{10}$
- ARP solo permite hacer 1 resolución por cada mensaje

*Fuente: https://www.rfc-es.org/rfc/rfc0826-es.txt

# Formato de mensaje ARP




*Fuente: https://www.rfc-es.org/rfc/rfc0826-es.txt

# Funcionamiento

- ARP maneja diferentes mensajes:
  - **Solicitud ARP**: operación = 1. Se usa para preguntar por una dirección IP, esperando recibir una respuesta que incluya la dirección MAC asociada a dicha dirección IP.
  - **Respuesta ARP**: operación = 2. Se usa para devolver como respuesta a una solicitud ARP la dirección MAC que está asociada a la dirección IP en la solicitud. Solo una máquina debe responder la solicitud.

  * **ARP gratuito**: operación = 1.En este mensaje los campos $IP_{emisor}$ e $IP_{destino}$ contienen la misma dirección IP y se usa para validar que nadie más en el segmento de red esté usando la misma dirección IP que la máquina que envía la solicitud.

*Fuente: https://www.rfc-es.org/rfc/rfc0826-es.txt

# Funcionamiento

- Datos relevantes en las solicitudes ARP
  - Cada implementador del protocolo ARP especificará el temporizador (en segundos) de vigencia en caché de una entrada con el mapeo Dir. IP – Dir MAC.
  - Cuando una máquina genera una solicitud ARP, espera normalmente 60 segundos antes de declarar inalcanzable un host si es que no recibe respuesta

*Fuente1: https://www.rfc-es.org/rfc/rfc0826-es.txt
*Fuente 2: https://stackoverflow.com/questions/37341598/arp-timeout-in-broadcast-message

# Análisis del protocolo ARP en PCAP4J

- **Clase ArpPacket (org.pcap4j.packet.ArpPacket)**
- **Métodos:**
  - int length( )
  - byte[ ] getRawData( )
  - ArpHeader getHeader( )
  - static ArpPacket newPacket(byte[ ] datos, int desplazamiento, int tam)

*Fuente: https://www.javadoc.io/doc/org.pcap4j/pcap4j/1.7.3/index.html

# Análisis del protocolo ARP en PCAP4J

- **Clase ArpHeader (org.pcap4j.packet.ArpPacket.ArpHeader)**
- **Métodos:**
  - ArpHardwareType getHardwareType( )
  - EtherType getProtocolType( )
  - int getHardwareAddrLengthAsInt( )
  - int getProtocolAddrLengthAsInt( )
  - ArpOperation getOperation( )
  - MacAddress getSrcHardwareAddr( )
  - InetAddress getSrcProtocolAddr( )
  - MacAddress getDstHardwareAddr( )
  - InetAddress getDstProtocolAddr( )

***Fuente: https://www.javadoc.io/static/org.pcap4j/pcap4j/1.7.3/org/pcap4j/packet/ArpPacket.ArpHeader.html**

# Análisis del protocolo ARP en PCAP4J

```
while (true) {
    byte[ ] trama = h.getNextRawPacket( );
    if (trama == null) {
     continue;
    } else {
     if (trama.contains(ArpPacket.class)) {
            ArpPacket arp = trama.get(ArpPacket.class);
            if (arp.getHeader().getOperation().equals(ArpOperation.REQUEST)) {
            System.out.println("Operación: Solicitud ARP")
            } else if (arp.getHeader().getOperation().equals(ArpOperation.REPLY)) {
            System.out.println("Operación: Respuesta ARP")
            }
    }//else
• }//while
```

***Fuente: https://www.javadoc.io/static/org.pcap4j/pcap4j/1.7.3/org/pcap4j/packet/ArpPacket.ArpHeader.html**

# Análisis del protocolo ARP en NPCAP

```
typedef struct dir_MAC{
  u_char byte1;
  u_char byte2;
  u_char byte3;
  u_char byte4;
  u_char byte5;
  u_char byte6;
} dir_MAC

typedef struct dir_IP{
  u_char byte1;
  u_char byte2;
  u_char byte3;
  u_char byte4;
} dir_IP
```

***Fuente: https://npcap.com/guide/npcap-tutorial.html**

# Análisis del protocolo ARP en NPCAP

```
typedef struct arp_enc{
  u_short tipo_hw;
  u_short tipo_proto;
  u_char longitud_hw;
  u_char longitud_proto;
  u_short operación;
  dir_MAC mac_origen;
  dir_IP  ip_origen;
  dir_MAC mac_destino;
  dir_IP  ip_destino;
} arp_enc;
```

***Fuente: https://npcap.com/guide/npcap-tutorial.html**

# Análisis del protocolo ARP en NPCAP

```
void packet_handler(u_char *parametros, const struct pcap_pkthdr *enc, const u_char *datos) {
//. . .
unsigned short tipo = (datos[12]*256)+ datos[13];
 if (tipo==2054){  //arp
    printf("Paquete ARP..\n");
/* obtenemos la posición de inicio del mensaje ARP a partir de los datos del paquete */
 arp_enc * arp;,
 arp = (arp_enc *) (datos + 14);
/* imprimimos tipo de hw, tipo de protocolo,longitud de hw y longitud de protocolo */
 printf("tipo hw:%u, tipo protocolo: %u, longitud hw:%u, longitud proto: %u, operación:%u\n", arp->tipo_hw,
arp->tipo_proto, arp->longitud_hw, arp->longitud_proto, arp->operacion);
 if (arp->operacion==1){ //solicitud ARP
     printf("Solicitud ARP\n");
 }//if

//…
```

*Fuente: https://npcap.com/guide/npcap-tutorial.html