Reti di Calcolatori

Canonico Roberto

a.a. 2023-2024

Author
Alessio Romano

February 18, 2025

Contents

1 HTTP **3**
1.1 Richieste HTTP . . . 4
1.2 Risposte HTTP . . . 4
1.3 Header . . . 5
1.4 Cookies . . . 6

2 FTP **6**
2.1 Funzionamento . . . 6
2.1.1 Controllo . . . 6
2.2 FTP Attivo/Passivo . . . 7

3 SMTP **7**

4 IPv4 **8**
4.1 Header IPv4 . . . 8
4.1.1 Identification, Flags, Fragment offset . . . 9
4.1.2 Frammentazione e riassemblaggio . . . 9
4.1.3 Opzioni dell'header IPv4 . . . 9

5 ARP **11**

6 Ethernet **12**

7 IPv6 **12**
7.1 Options . . . 13

8 UDP **13**

9 TCP **14**

1 HTTP

L'**hyperText Transfer Protocol (HTTP)** è un protocollo di livello *applicazione*, per la comunicazione tra due entità definite **client** e **server**. Si tratta di un protocollo basato su **TCP** a livello trasporto.

- A livello applicativo si genera una richiesta o una risposta HTTP
- A livello di trasporto si instaura una connessione TCP tra *client* e *server*
- Il livello di trasporto si occupa della trasmissione affidabile dei dati

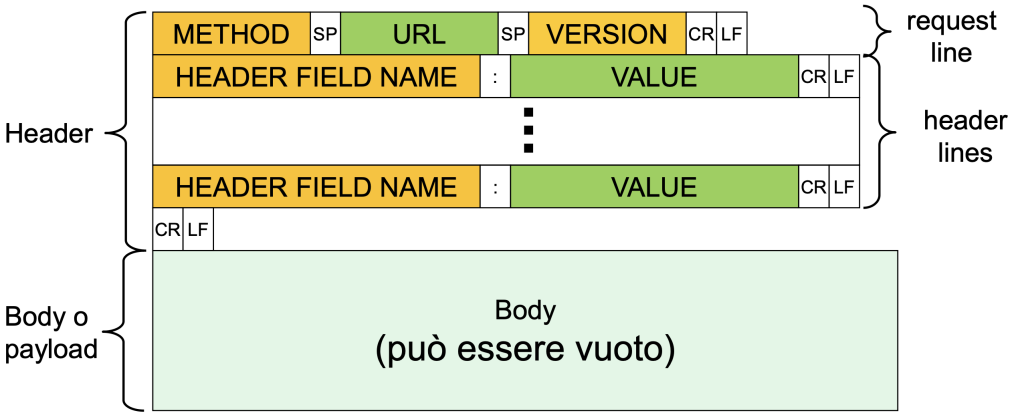
I tipi di connessione instaurabili sono 3:

1. **Non Persistente**: Si apre e si chiude una connessione TCP tra client e server per ogni **oggetto** richiesto
2. **Persistente**: Si apre una connessione TCP tra client e server, il primo invia varie richieste attendendo la risposta da parte del server tra una richiesta e l'altra, al termine dello scambio si chiude la connessione
3. **Persistente con Pipelining**: Si apre una connessione TCP tra client e server, il client invia più richieste senza attendere le risposte, il server risponde alle richieste, al termine dello scambio si chiude la connessione

1.1 Richieste HTTP

Una richiesta HTTP è formata da due sezioni principali:

- **Riga di richiesta**: la prima riga, contenente i campi
  - **Method**: il metodo della richiesta
    - HEAD - GET - PUT - POST - DELETE - TRACE - OPTION
    - I metodi si classificano come **sicuri** se non modificano lo stato del server, e **idempotenti** se l'effetto di un'unica richiesta è identico all'effetto di più richieste
  - **URL**: l'identificatore della risorsa sul server
  - **Version**: versione del protocollo tra 1.0 e 1.1
- **Righe di intestazione**: da 0 ad n righe riservate per informazioni aggiuntive, ne sono esempi
  - **Host**: specifica l'host in cui risiede l'oggetto (per la cache del proxy)
  - **Connection**: specifica il tipo di connessione (persistente, non persistente) ecc

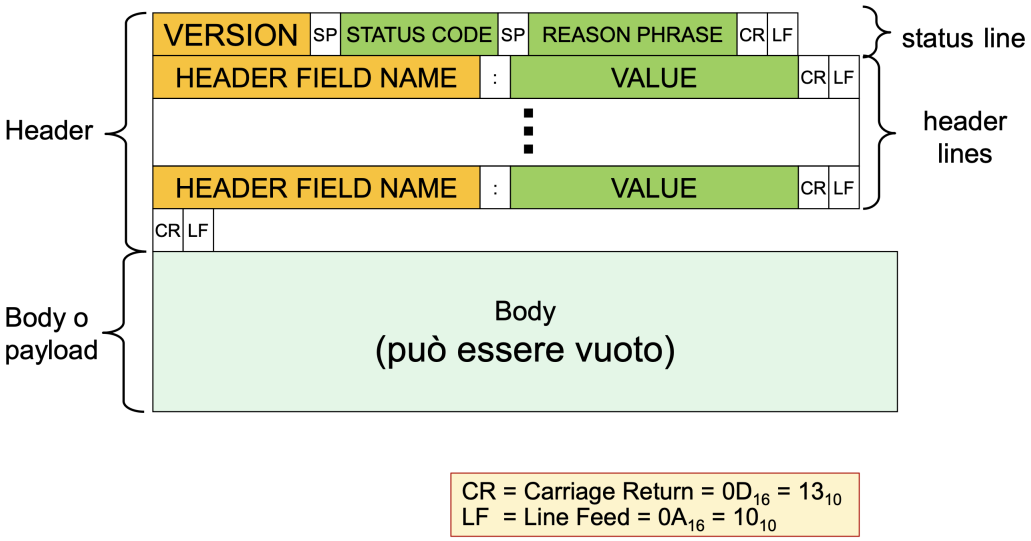


1.2 Risposte HTTP

Una risposta HTTP è formata da tre sezioni principali:

- **Riga di stato**: la prima riga, contiene i campi
  - **Version**: versione del protocollo tra 1.0 e 1.1
  - **Status Code**: codice di stato
    - 1xx: informational
    - 2xx: successful
    - 3xx: redirection
    - 4xx: client error
    - 5xx: server error

  - **Reason Phrase**: messaggio di stato (OK su 200 successfull)
- **Righe di intestazione**: da 0 ad n righe riservate per informazioni aggiuntive, ne sono esempi:
  - **Content-Type**: tipo di contenuto nel messaggio
  - **Last Modified**: data di ultima modifica
- **Body**: Il dato effettivo da trasportare dal server al client



1.3 Header

Gli header in HTTP si suddividono in 3 tipi:

- **Header Generali**: si applicano sia alle richieste che alle risposte e sono ad esempio
  - **MIME-Version**: la versione MIME che si sta utilizzando
  - **Transfer-Encoding**: la codifica della trasmissione
  - **Cache-Control**: tipo di meccanismo di caching suggerito o richiesto per la risorsa
  - **Connection**: tipo di connessione da utilizzare
- **Header risposta**: header specifici della risposta posti dal server per specificare informazioni sulla risposta e su se stesso al client
  - **Server**: una stringa che descrive il server
  - **Accept-Range**: il tipo di range che può accettare
- **Header Identità**: specificano informazioni sul contenuto del body o della risorsa inviata
  - **Content-Type**: Oggetto/Formato
  - **Content-Length**: lunghezza in byte del body
  - ...

1.4 Cookies

Il protocollo HTTP non è in grado di mantenere uno stato, dunque per mantenere brevi informazioni scambiate tra server e client, si utilizzano i **cookie**. Si hanno due nuovi header possibili, uno per la richiesta uno per la risposta

- **Set-Cookie (Risposta)**: il server permette al client di memorizzare cookie e rispedirli alla richiesta successiva
- **Cookie (Richiesta)**: il client decide se spedire i cookie in base al documento, l'età del cookie ecc

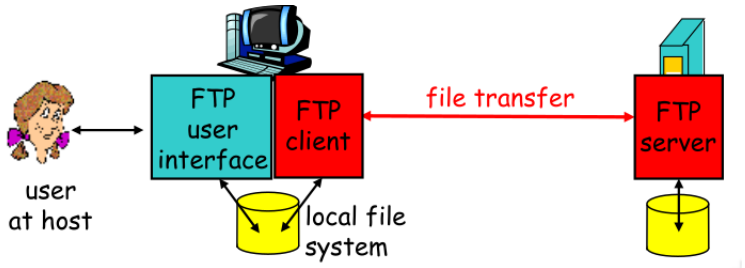
I cookie contengono campi che ne stabiliscono la durata e lo scope, detti **max-age** e **expires**

2 FTP

Il protocollo FTP (**file transfer protocol**) permette il trasferimento di uno o più file qualsiasi tra due macchine. Lavora utilizzando due connessioni, una dati e una di controllo (out of band)

2.1 Funzionamento

FTP si basa sul modello client/server, in cui un client FTP trasferisce file da o verso una macchina remota (server FTP)



Il client FTP contatta il server FTP alla porta 21, e vengono aperte due connessioni parallele

- **Dati**: file che fluiscono dal client al server o viceversa
- **Controllo**: scambio di comandi, messaggi di risposta tra il client e il server

Il server FTP mantiene uno stato, ad esempio la directory corrente, i dati dell'autenticazione etc.

2.1.1 Controllo

Nella connessione di controllo, vengono inviati comandi e ricevute risposte come testo ASCII. Esempi di comandi sono:

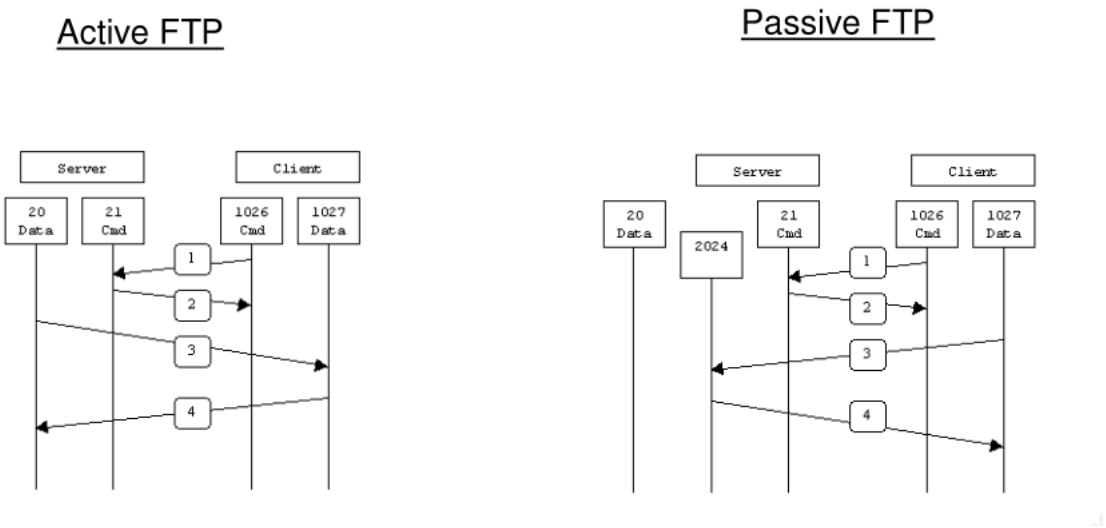
- **USER** username
- **PASS** password
- **LIST**: restituisce i files presenti nella directory
- **GET** filename

Il tipo di paradigma client/server presuppone che il server venga configurato per accettare connessioni FTP, mentre i client FTP sono invece disponibili pressochè su tutti i sistemi operativi

2.2 FTP Attivo/Passivo

L'interazione tra client e server FTP può essere di 2 tipi:

- **Modalità attiva**: Il client informa il server su quale porta può ricevere i dati, il server su connette al client per trasferire i dati
- **Modalità passiva**: Il server informa il client su quale porta è in ascolto per il trasferimento dei dati, il client si connette per trasferire i dati



3 SMTP

Il **Simple Mail Transfer Protocol (SMTP)**, è il protocollo utilizzato nella comunicazione tramite mail. La comunicazione tramite mail si basa su 3 attori principali:

- **User Agent**: consentono agli utenti di leggere, rispondere, inoltrare, salvare ecc... le mail (Outlook, Gmail, ...)
- **Mail Server**: ciascun destinatario ha una **Mail Box** su un mail server, che si occupa di gestire i messaggi a lui inviati e da lui inviati
- **SMTP**: Il protocollo con cui comunicano i mail server

Si noti che il processo di invio di una mail, passa attraverso i **mail server**, lo user agent mittente carica la mail da inviare sul suo mail server, che si occuperà di inoltrarla al mail server destinatario, da cui lo user agent destinatario potrà recuperarla attraverso protocolli quali IMAP e POP3. Questa comunicazione avviene tramite **SMTP**. SMTP è costituito nel seguente modo

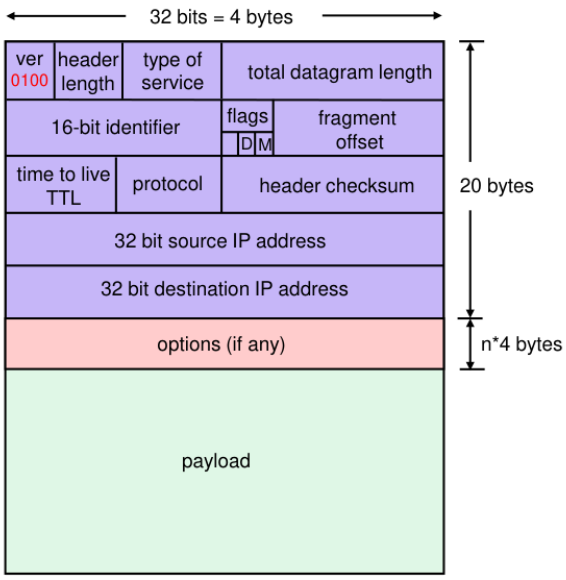
- **Header**: linee di intestazione quali: To, From, Subject, Data...
- **Body**: Dati in caratteri ASCII

Per ovviare alla limitazione dell'invio di dati in ASCII, è stato creato il MIME che consente di gestire diversi tipi di dati attraverso una codifica di quest'ultimi (**Content-Transfer-Encoding**) e un header aggiuntivo (**Content-Type**). Si possono gestire file MP3, HTML, MP4, JPEG, PNG...

4 IPv4

Come accennato in precedenza, il protocollo IPv4 opera sul Layer 3 (rete). Vediamo di cosa si occupa nello specifico.

- IP gestisce indirizzamento, frammentazione, ri-assemblaggio e multiplexing dei datagram
- È implementato negli end terminal e nei router e si occupa dell'instradamento dei pacchetti.
- Un datagram IPv4 può avere una dimensione massima di 65535 byte ($2^16 - 1$)
- È composto da un header e un payload, l'header è costituito da una struttura fissa (20 byte) ed una opzionale e il payload è il dato creato dal protocollo di trasporto TCP o UDP



Il protocollo IPv4 è un servizio **best effort**, il termine indica che la rete non discrimina un pacchetto rispetto ad altri, tuttavia non garantisce di prevenire:

- Pacchetti duplicati
- Consegna ritardata o fuori ordine
- Corruzione di dati
- Perdita di pacchetti

Il ruolo di rendere la consegna affidabile, viene spostato sui meccanismi di controllo realizzati nei protocolli di livello superiore.

4.1 Header IPv4

In IPv4 l'header è costituito da una struttura fissa (20 byte) ed una opzionale di lunghezza multipla di 4 byte. I campi che compongono l'header sono

- **IP header length (4 bit)**: lunghezza dell'header in multipli di 32 bit (max 60 byte)
- **Type-of-service (8 bit)**: specifica il tipo di servizio che si richiede alla rete
- **Total length (16 bit)**: indica la lunghezza in byte del pacchetto (header + payload)
- **Protocol (8 bit)**: indica il protocollo di livello superiore associato al payload (6 TCP, 17 UDP)
- **Header checksum (16 bit)**: serve a verificare l'integrità dell'header IP
- **Source IP Address (32 bit)**: indirizzo IP del mittente
- **Destination IP Address (32 bit)**: indirizzo IP del destinatario
- **Identification (16 bit), Flags (3 bit), Fragment Offset (13 bit)**: sono usati in caso di frammentazione del pacchetto da parte di un router

4.1.1 Identification, Flags, Fragment offset

Questi campi servono a gestire la frammentazione dei pacchetti IPv4. Un pacchetto IPv4 può essere frammentato da un router in una sequenza di pacchetti che viaggiano singolarmente verso il destinatario, una volta arrivati a destinazione il livello IP del destinatario si occupa di riassemblari prima di di consegnarlo allo strato superiore.

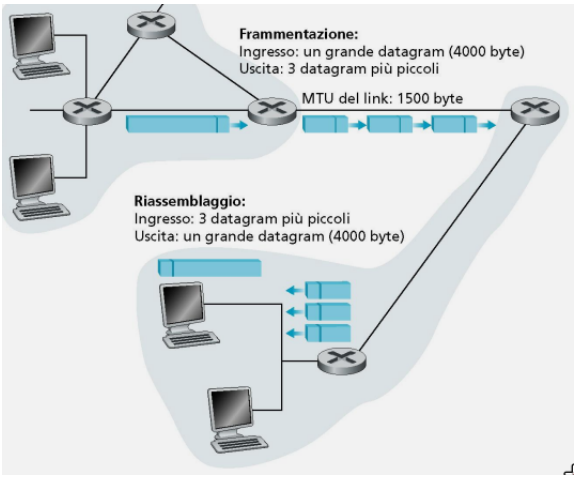
- Un pacchetto può essere frammentato anche più volte lungo il percorso
- La necessità di frammentare un pacchetto si presenta quando la dimensione del pacchetto supera la **MTU (Maximum Transmissable Unit)** sul link di uscita
- Il valore della MTU dipende dalla tecnologia usata al livello 2 (es. ethernet 1500 byte)

Nello specifico, ciascuno di questi campi dell'header si occupano di

- **Identification**: è un identificativo del datagram e serve ad associare diversi frammenti ad un unico pacchetto originario
- **Flags**: Il bit D (don't fragment) indica se il pacchetto può essere frammentato, mentre il bit M (more fragments) indica se il pacchetto è l'ultimo frammento
- **Fragment Offset**: identifica la posizione del frammento all'interno del pacchetto

4.1.2 Frammentazione e riassemblaggio

- Se un pacchetto di dimenione N arriva ad un router e deve essere trasmesso su un link di uscita con MTU M<N, il pacchetto è frammentato
- Ogni frammento è trasmesso come singolo pacchetto IP
- La dimensione del payload di ogni frammento è un multiplo di 8 byte
- Tutti i frammenti hanno lo stesso ID number



Ci sono tuttavia alcuni problemi relativi all'operazione di frammentazione: il compito di riassemblaggio è oneroso, il destinatario deve collezionare tutti i frammenti del parcchetto originario prima di consegnare il payload al livello superiore, se non li riceve entro un determinato tempo, i frammenti arrivati sono scartati. Un metodo per evitare la frammentazione dei pacchetti lungo il percorso, talvolta si effettua un **Path MTU Discovery**, cioè si determina la più piccola MTU lungo il percorso da un host A ad un host B.

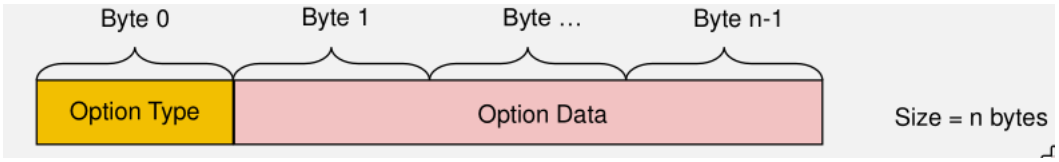
4.1.3 Opzioni dell'header IPv4

L'header può contenere campi Opzione mediante le quali si richiede una elaborazione "speciale" da parte dei router.

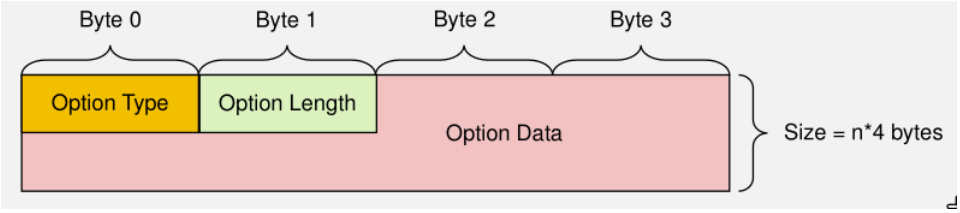
- Security
- Source Routing
- Route Recording
- Stream Identification
- Timestamping

Per la presenza delle opzioni, l'header IP può essere di lunghezza variabile, da questo deriva la presenza del campo Header Length. Ci sono due formati per segnalare le opzioni

- **Single byte**: Le opzioni di questo formato hanno una lunghezza di n byte definita implicitamente dal valore di Option Type



- **Multiple bytes**: le opzioni di questo formato hanno una lunghezza multipla di 4 byte, esplicitamente indicata nel campo Option Length

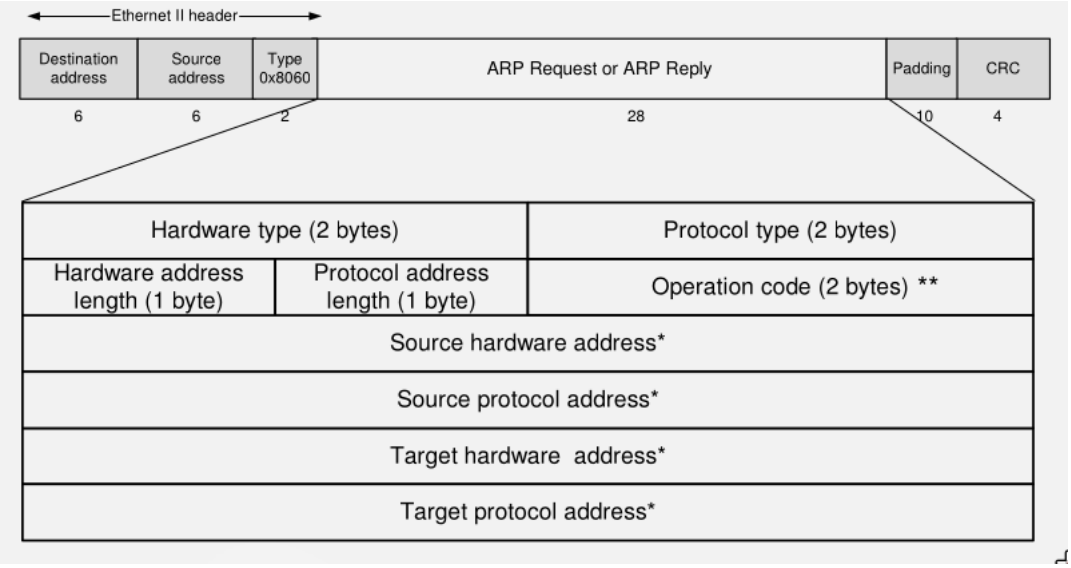


5 ARP

L'**address resolution protocol (ARP)**, si occupa della mappatura tra indirizzi IP e indirizzi MAC delle interfacce di rete.Specificamente ARP associa ad un indirizzo IP il corrispondente indirizzo MAC e RARP associa ad un indirizzo MAC il corrispondente indirizzo IP. Ci sono due scenari di utilizzo per il protocollo ARP

- A vuole trasmettere un datagram IP a B, nella stessa LAN
  1. A invia una richiesta ARP in broadcast in una *frame ethernet*
  2. B riceve la richiesta ed invia una risposta ARP direttamente ad A
- A vuole trasmettere un datagram IP a B, al di fuori della LAN
  1. A determina l'indirizzo IP del gateway R1 dalla sua tabella di routing
  2. A invia una richiesta ARP in broadcast in una *frame ethernet*
  3. R1 risponde inviando il suo MAC address ad A

Il formato di una richiesta ARP è il seguente



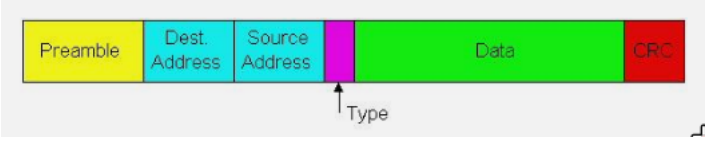
dove i campi sono rispettivamente

- **Protocol Type (16 bit)**: Specifica il protocollo di rete utilizzato (0x0800 per IPv4)
- **Hardware Type (16 bit)**: Specifica il tipo di hardware utilizzato (1 per MAC)
- **Hardware address length (8 bit)**: Lunghezza dell'indirizzo MAC
- **Protocol address length (8 bit)**: Lunghezza indirizzo IP
- **Op Code (16 bit)**: se OpCode = 0x0001 ARP request, se OpCode = 0x0002 ARP reply
- **Source/Target hardware address (48 bit)**: Indirizzo MAC mittente e destinatario
- **Source/Target protocol address (32 bit)**: Indirizzo IP mittente e destinatario

Gli host sulla rete, mantengono una tabella con indirizzi MAC e rispettivi indirizzi IP, chiamata **ARP table**. I messaggi ARP possono essere anche *unsolicited*, e inviati da un host per richiedere agli altri host sulla rete di modificare la propria ARP table, ne sono esempi i **Gratuitos ARP (GARP)**, e l'**ARP announcement**

6 Ethernet

Il protocollo **Ethernet** a livello data-link, si occupa di incapsulare datagrammi IP in *frame* ethernet. Una frame è costituita nel seguente modo

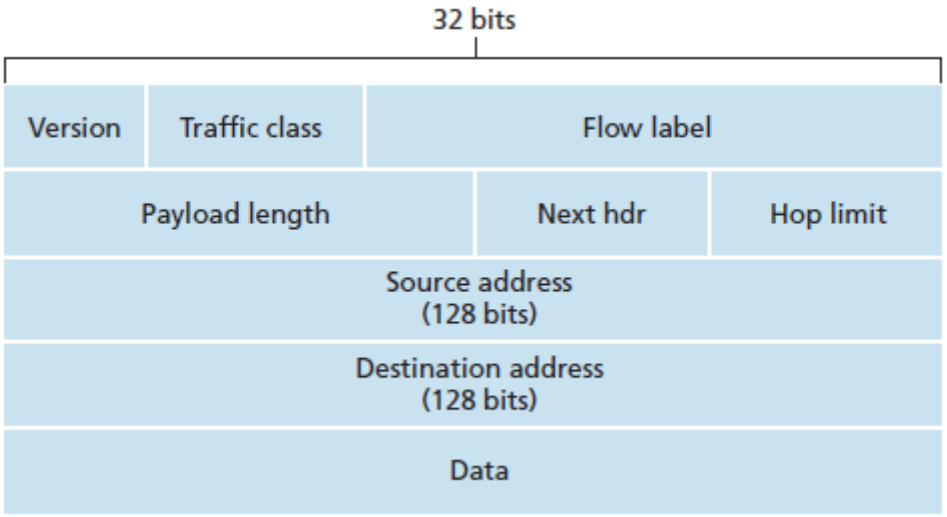


È composta dai seguenti campi:

- **Preambolo (8 byte)**: è composto da 7 byte 10101010 e l'ultimo byte 10101011, serve a sincronizzare le schede di rete e notificare che è in arrivo la "parte importante" della frame
- **Source Address (6 byte)**: indirizzo MAC del mittente
- **Destination Address (6 byte)**: indirizzo MAC del destinatario
- **Type (2 byte)**: indica il protocollo di rete sovrastante (solitamente IP)
- **CRC (4 byte)**: consente alla scheda di rete di rilevare errori nella frame

7 IPv6

L'**Internet Protocol v6 (IPv6)** è stata progettata per risolvere problemi di IPv4 quali lo spazio di indirizzamento in esaurimento, la sicurezza ecc... Un datagram IPv6 segue la seguente struttura



L'header IPv4 è a lunghezza fissa di 40 byte, e formato dai seguenti campi

- **Version (4 bit)**: campo che identifica il numero di versione IP
- **Traffic Class (8 bit)**: permette di attribuire priorità a determinati datagrammi di un flusso o proveninenti da specifiche applicazioni
- **Flow Label (20 bit)**: permette di identificare un flusso di datagrammi
- **Payload length (16 bit)**: è trattato come un intero senza segno e indica il numero di byte nel datagramma IPv6 che seguono l'intestazione

- **Hop Limit (8 bit)**: è decrementato di 1 ogni volta che il datagramma viene inoltrato da un router, quando il valore raggiunge 0 viene eliminato
- **Indirizzi sorgente e destinazione**: Indirizzi IPv6 del mittente e del destinatario
- **Next Header (8 bit)**: campo che identifica il protocollo a cui verranno consegnati i contenuti del datagramma, per esempio TCP o UDP
- **Dati**: Il payload associato al datagramma

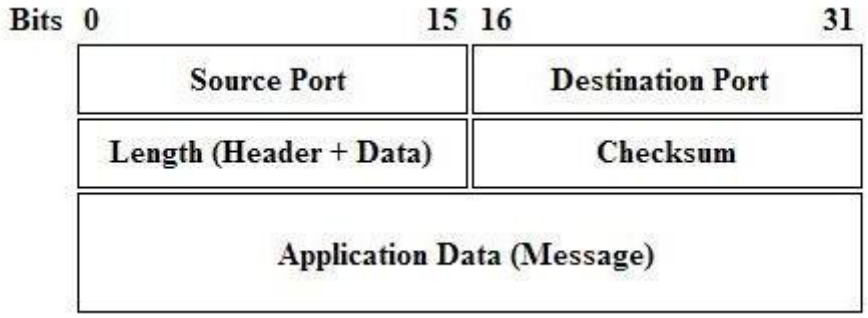
Si noti che IPv6 non supporta **frammentazione/riassemblaggio** nei sistemi intermedi, ma solo negli end system in quanto ritenuta una pratica troppo onerosa. In caso di un pacchetto troppo grande, un router IPv6 invia un messaggio ICMP "Packet too big", sarà compito dell'end system mittente ristrutturare il datagramma in base all'MTU minima dei router sul suo percorso

7.1 Options

Il campo options non è completamente scomparso, infatti uno dei possibili valori di **next header** permette di puntare ad un header aggiuntivo contenente le opzioni. Ciò permette di avere un header principale sempre a lunghezza fissa di 40 byte.

8 UDP

Il protocollo **UDP** fornisce un servizio best effort, rapido ma non sicuro quanto TCP. Infatti UDP implementa il minimo richiesto da un protocollo di trasporto, si occupa di aggiungere i campi port al segmento ed effettuare operazioni di multiplexing, demultiplexing. La struttura di un segmento UDP è la seguente



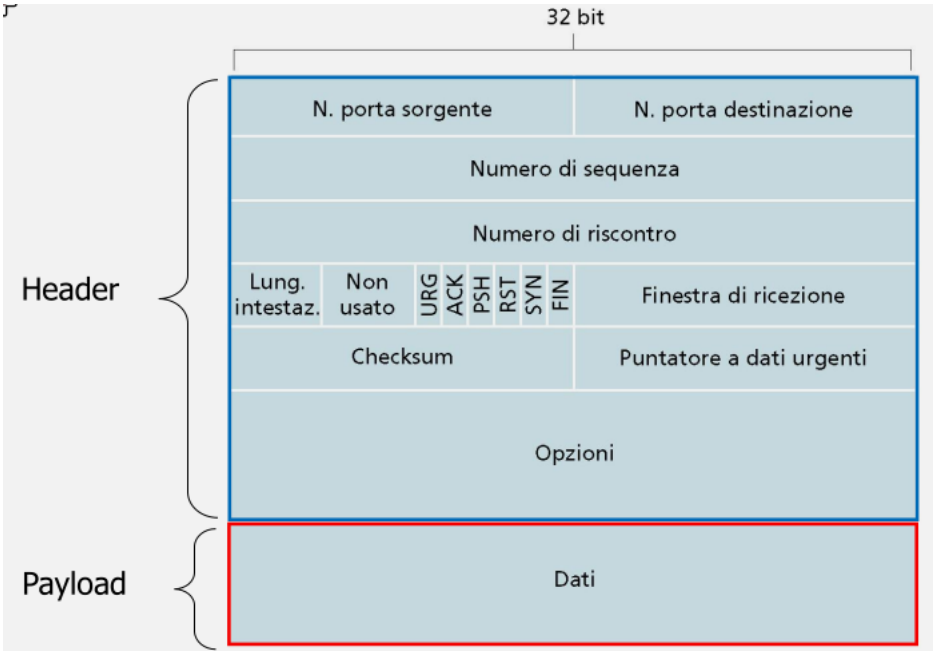
È composto da i campi

- **Source Port (16 bit)**: Numero porta di origine
- **Dest Port (16 bit)**: Numero porta destinatario
- **Length (16 bit)**: Lunghezza dell'header + data
- **Checksum (16 bit)**: Campo per il rilevamento degli errori

Il campo checksum in UDP, si basa sulla somma in complemento ad 1 di tutte le parole da 16 bit nel segmento, e l'eventuale riporto finale viene sommato al primo bit. Il valore risultante viene inserito nel campo checksum e in ricezione si sommano di nuovo tutte le parole da 16 bit nel segmento e successivamente si somma il valore del campo checksum, se tutti e 16 bit risultanti sono ad 1non ci sono errori, altrimenti ne è stato introdotto almeno 1

9 TCP

Il protocollo TCP è un ulteriore protocollo di livello trasporto *connection oriented*, la struttura di un segmento TCP è la seguente



- **Numero di sequenza (32 bit)**: è il numero di byte dopo lo 0 a cui fa riferimento il segmento. Se supponiamo di voler trasferire 100.000 byte con MSS 1000, i segmenti saranno 100 con il primo segmento avente sequence number 0, il secondo 1000, il terzo 2000...
- **Numero di riscontro (32 bit)**: è il numero di byte relativi al segmento che il ricevente si aspetta di ricevere. Supponiamo di trasferire 100.000 byte con MSS 1000, il ricevente in seguito alla ricezione del segmento con sequenza 1000, scriverà nel Ack Number 1001, in quanto è da dove si aspetta di ricevere i dati nel prossimo segmento
- **Rcv Window (16 bit)**: viene utilizzato per il controllo del flusso, e fornisce lo spazio libero nel buffer del destinatario
- **Lunghezza dell'intestazione (4 bit)**: specifica la lunghezza dell'intestazione TCP in multipli di 32 bit.
- **Opzioni (lunghezza variabile)**: Permette di concordare la dimensione massima del segmento ed altri valori
- **Flag (6 bit)**: Contiene alcune flag attivabili quali il bit URG per i dati urgenti ACK per indicare che l'ACK number è valido, RST, SYN, FIN per stabilire e chiudere la connesione