ANALISI DELLE RETI (2°parte)

Dopo aver visto quali sono i software da installare per effettuare l'analisi di una rete, facciamo una premessa prima di cominciare ad utilizzare qualche strumento messo a disposizione dal potente KALI Linux, che affonteremo successivamente.

In questa premessa cercheremo di conoscere ed approfondire i protocolli più interessanti, e descriveremo tutti i termini ed acronimi utilizzati, in modo che al termine di questa parte del tutorial sulle reti, avremo una maggiore consapevolezza di cosa avviene in una rete LAN o WAN.

Ricordiamoci che **LAN** sta per **L**ocal **A**rea **N**etwork e cioè una rete locale, mentre **WAN** sta per **W**ide **A**rea **N**etwork e cioè una rete dimensionamente più grande come ad esempio Internet, che è la più grande rete al mondo.

La precedente spiegazione è un esempio di come affronteremo gli argomenti nel tutorial, argomenti questi che richiederebbero tempo e studio, ma noi li tratteremo in maniera oserei dire "essenziale", aprendo delle brevi parentesi ogni qualvolta ci sia un termine nuovo da conoscere.

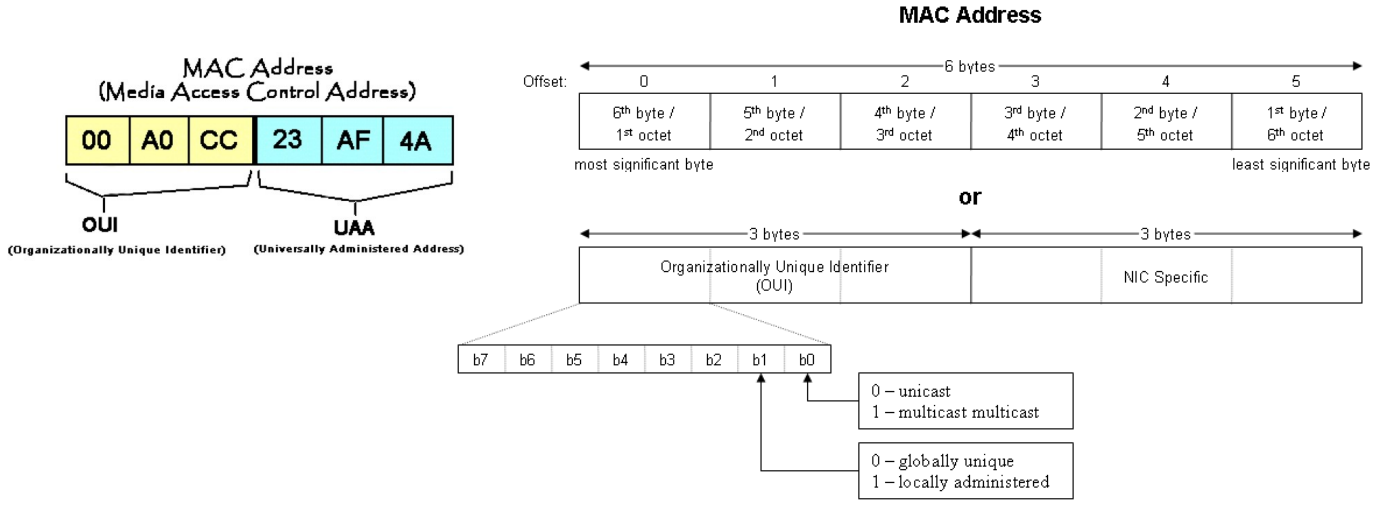
Probabilmente per molti saranno nozioni scontate, ma questo vuol essere un tutorial per tutti, o meglio per tutti quelli che comunque hanno un minimo di conoscenza tecnica su questi argomenti.

I punti principali che affronteremo sono i seguenti:

- *MAC Address*
- *Protocollo IPV4*
- *Indirizzamento IP*
- *Subnetting*
- *Protocollo IPV6*
- *Protocollo TCP*
- *Protocollo UDP*
- *Protocollo ICMP*
- *PORTE*
- *Socket*
- *Protocollo HTTP e HTTPS*
- *Protocollo ARP*
- *NAT*
- *Protocollo DHCP*

MAC Address (**M**edia **A**ccess **C**ontrol)

E' un codice di 48 bit assegnato in modo univoco dal produttore ad ogni scheda di rete o wireless prodotta a livello mondiale. Esso rappresenta un identificatico di ogni scheda ed è diviso in due parti, una che identifica il costruttore ed una che identifica il numero di serie della scheda



IPV4 (**I**nternet **P**rotocol **V**ersione **4)**

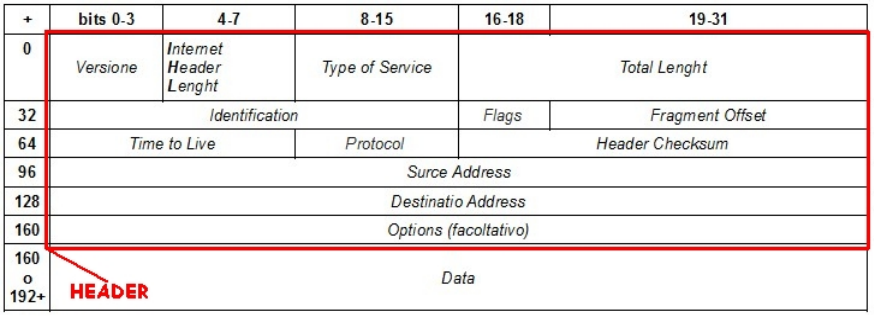
L'IPV4 fa parte della suite dei protocolli di internet, ed è quello attualmente in uso.

Secondo questo protocollo, i dati transitano in un pacchetto chiamato datagramma, composto da 13 campi oltre ad uno facoltativo.

+	bits 0-3	4-7	8-15	16-18	19-31
0	*Versione*	***I**nternet **H**eader **L**enght*	*Type of Service*	*Total Lenght*	
32	*Identification*			*Flags*	*Fragment Offset*
64	*Time to Live*		*Protocol*	*Header Checksum*	
96	*Surce Address*				
128	*Destinatio Address*				
160	*Options (facoltativo)*				
160 o 192+	*Data*				

Analizziamo ora i campi del datagramma IPV4.

- **<u>Versione</u>** - *in questo campo che occupa 4 bit, troviamo la versione, che nel caso dell'IPV4 vale 4.*
- **<u>IHL</u>** (Internet Header Lenght) - *anche questo campo è di 4 bit, e contiene la lunghezza in word (32 bit) dell'header del pacchetto, se non c'è il campo options troveremo in questo campo il valore 5 (cioè 20 bytes).*



- **<u>Type of service</u>** - *in questo campo da 8 bit, viene definita la modalità con cui trattare il datagramma, i possibili valori sono i seguenti:*
  - *bit 0-2 Precedenza*
  - *bit 3 Latenza (tempo tra l'arrivo e la ripartenza di un dato) 0=normale 1=bassa*
  - *bit 4 Throughput (effettiva capacità del canale trasmissivo in bit/s) 0=normale 1=alto*
  - *bit 5 Affidabilità 0=normale 1=alta*
  - *bit 6-7 Riservati per usi futuri*

- **<u>Total Lenght</u>** - *lunghezza in byte dell'intero pacchetto.*

- **Identification** - *è un valore assegnato dal mittente per assemblare i frammenti di un datagramma, tutti i frammenti dello stesso datagramma hanno lo stesso identificatore.*
  *Quando la dimensione dei dati e l'header supera la **MTU** (**M**aximum **T**rasmission **U**nit cioè le dimensioni massime del pacchetto dati) allora il datagramma viene **frammentato**, cioè diviso in diversi datagrammi che avranno lo stesso identificatore.*

- **Flags** - *3 bits con il seguente significato:*
  - ***bit 2*** *riservato sempre settato a 0*
  - ***bit 1*** ***DF** (**D**on't **F**ragment) Impone al router di non frammentare il datagramma in quanto il destinatario non può ricomporlo.*
    Con questo bit ad 1 ed il datagramma non può essere inviato senza essere frammentato allora viene scartato.
  - ***bit 0*** ***MF** (**M**ore **F**ragments) se vale 1 indica che il pacchetto fa parte di un datagramma, se invece vale 0 indica che è l'ultimo pacchetto del datagramma.*

- **Fragment Offset** - *Il campo occupa 13 bit ed indica la posizione (offset) del frammento nel datagramma corrente. Tutti i frammenti tranne l'ultimo, devono avere in questo campo un valore multiplo di 8 bit (dimensione del frammento elementare) il primo frammento ha offset 0.*

  In questo esempio un pacchetto di 4000 dati viene frammentato in 3 parti.

	Lenght=4000				.....

	Lenght=1500	*ID 0x01C4*	*Flag bit0=1*	*Fragment Offset=0*	.....
	Lenght=1500	*ID 0x01C4*	*Flag bit0=1*	*Fragment Offset=1480*	.....
	Lenght=1040	*ID 0x01C4*	*Flag bit0=0*	*Fragment Offset=2960*	.....

- **Time to Live** - *il campo occupa 8 bit ed è un contatore che limita la vita del pacchetto ad ogni passaggio tra Router, viene decrementato, quando arriva a zero il pacchetto viene scartato e viene inviato un avviso all'host sorgente.*

- **Protocol** - *il campo occupa 8 bit ed indica il protocollo utilizzato, ad esempio TCP ha il codice 6 mentre UDP ha codice 17, la **RFC 790** stabilisce i vari codici.*
  *Le **RFC** (**R**equest **F**or **C**omment) sono pubblicate nell' RFC Editor http://www.rfc-editor.org/retrieve/ e sono documenti pubblicati dall'organismo internazionale* **Internet Engineering Task Force**

- **Header Checksum** - *il campo occupa 16 bit e contiene il valore del Checksum per il controllo degli errori. Il Checksum viene calcolato ed associato al pacchetto trasmesso secondo determinati algoritmi utilizzati nel controllo degli errori. Questo valore viene ricalcolato ogni volta che il pacchetto entra in un Router e confrontato con quello presente nel campo, se non c'è corrispondenza il pacchetto viene scartato.*

- **Source Address e Destination Address** - *Sono due campi a 32 bit che contengono gli indirizzi dell'host mittente e destinatario.*

- .**Option** - *il campo opzionale può essere utilizzato per delle informazioni sui Router, lista e percorso effettuato, o per informazioni riguardo la sicurezza.*

Ecco come si presenta il contenuto di un datagramma IPV4

```
▲ Internet Protocol Version 4, Src: 192.168.1.109, Dst: 216.58.198.5
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x2ec4 (11972)
  ▷ Flags: 0x4000, Don't fragment
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x6bb6 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.109
    Destination: 216.58.198.5
```

Ed ecco i bit che realmente vengono inviati sulla rete, quello evidenziato è il datagramma IPV4.

```
01100100 11010001 01010100 10101010 10111100 00110010 11010000 01010011
01001001 01010100 01010100 01001001 00001000 00000000 01000101 00000000
00000000 00101000 00101110 11000100 01000000 00000000 10000000 00000110
01101011 10110110 11000000 10101000 00000001 01101101 11011000 00111010
11000110 00000101 11000001 11001111 00000001 10111011 00000000 11110101
01000111 10100000 10010010 00100100 00001110 11100100 01010000 00010000
00000101 10101001 10011100 10101101 00000000 00000000
```

INDIRIZZAMENTO IP

Abbiamo parlato sopra di indirizzi degli host (dispositivi collegati alla rete) ricordiamo ora come funziona l'indirizzamento di una rete secondo il protocollo IP.
Una rete viene identificata secondo un indirizzo IP ed una Subnet Mask, che occupano entrambe **4 bytes**. Ogni campo **può valere al massimo 255**.
Il campo della Subnet Mask contenente 255 indica che il corrispondente campo dell'indirizzo IP non può variare.
Nel seguente caso si identifica una rete che avrà l'ultimo campo variabile ed i possibili indirizzi **vanno da 192.168.1.1 a 192.168.1.254.**
L'indirizzo **192.168.1.0** non si può utilizzare in quanto **identifica la rete**, e l'indirizzo **192.168.1.255** non si può utilizzare per un host, in quanto identifica la comunicazione di tipo **Broadcast** (cioè diretta a tutti i dispositivi collegati alla rete).

	8 bit	**8 bit**	**8 bit**	**8 bit**
Indirizzo IP	192	168	1	0
Subnet Mask	255	255	255	0

La stessa rete può essere identificata con la seguente sintassi **192.168.1.0 / 24**
Il valore indicato dopo l'indirizzo rappresenta il **numero di bit ad 1 della Subnet Mask**.

Gli indirizzi possibili vengono comunque suddivisi in **5 classi di indirizzo** secondo il metodo **Classfull**.

	0	**8**	**16**	**24**	**INTERVALLO INDIRIZZI**
CLASSE A	**0** NETWORK	HOST			***1.0.0.0 127.255.255.255***
CLASSE B	**10** NETWORK		HOST		***128.0.0.0 191.255.255.255***
CLASSE C	**110** NETOWORK			HOST	***192.0.0.0 223.255.255.255***
CLASSE D	**1110** MULTICAST				***224.0.0.0 239.255.255.255***
CLASSE E	**11110** RESERVED				***240.0.0.0 255.255.255.255***

Tutti gli host che hanno un indirizzo IP con lo stesso indirizzo di rete (network) possono comunicare tra di loro, se hanno indirizzo di rete diverso non possono invece comunicare anche se sono fisicamente connessi.

Di seguito le caratteristiche delle 4 classi di indirizzamento.

- **Indirizzi di classe A: da 1.0.0.0 a 127.255.255.255**

  126 reti con 16.777214 host - subnet mask 255.0.0.0 o anche /8

  7 bit in rosso per le possibili reti 24 bit in blu per gli host di ogni rete

da	00000001 1	00000000 0	00000000 0	00000000 0
a	01111111 127	11111111 255	11111111 255	11111111 255

- **Indirizzi di classe B: da 128.0.0.0 a 191.255.255.255**

  16.382 reti con 65.534 host - subnet mask 255.255.0.0 o anche /16

  14 bit in rosso per le possibili reti 16 bit in blu per gli host di ogni rete

da	10000000 128	00000000 0	00000000 0	00000000 0
a	10111111 191	11111111 255	11111111 255	11111111 255

- **Indirizzi di classe C: da 192.0.0.0 a 223.255.255.255**

  2.097.150 reti con 254 host - subnet mask 255.255.255.0 o anche /24

  21 bit in rosso per le possibili reti 8 bit in blu per gli host di ogni rete

da	11000000 192	00000000 0	00000000 0	00000000 0
a	11011111 223	11111111 255	11111111 255	11111111 255

- **Indirizzi di classe E: da 224.0.0.0 a 239.255.255.255**

  riservate per il multicasting

da	11100000 224	00000000 0	00000000 0	00000000 0
a	11101111 239	11111111 255	11111111 255	11111111 255

- **Indirizzi di classe D: da 240.0.0.0 a 254.255.255.255**

  riservate per usi futuri

da	11110000 240	00000000 0	00000000 0	00000000 0
a	11111011 254	11111111 255	11111111 255	11111111 255

SUBNETTING

Per gestire in maniera efficiente le reti delle 3 classi utilizzabili A,B e C e per limitare il traffico globale, è opportuno creare delle sottoreti usando il metodo del subnetting, che consiste nel modificare il prefisso di rete, utilizzando per esso parte dei bit destinati agli host, estendendo in questo modo l'indirizzo di rete.

NETWORK	HOST

NETWORK	SUBNET	HOST

NETWORK ESTESA	HOST

Uno strumento utile per effettuare il subnetting viene dato dall'applicazioe online **IPSubnetCalc** http://www.subnet-calculator.com.

Supponiamo ad esempio di avere la necessità di applicare la tecnica del subnetting sul seguente indirizzo in classe C; **192.168.1.0/24.**

E supponiamo inoltre di voler ottenere 3 sottoreti per almeno 30 PC..

Per avere 32 indirizzi ho bisogno di almeno 5 bit perciò posso rubare 3 bit alla parte dedicata agli host per estendere l'indirizzo di rete:

192.168.1.xxxxxxxx diventerà **192.168.1.sssxxxxx**

in questo modo avrò 3 bit per definire 8 sottoreti, più che sufficienti per avere le mie 3 sottoreti richieste, che saranno:

192.168.1.000xxxxx	cioè	**192.168.1.0 / 27**
	o anche	**192.168.1.0 mask 255.255.255.224**
	30 host da	**192.168.1.1** a **192.168.1.30**
	indirizzo di broadcast	**192.168.1.31**
192.168.1.001xxxxx	cioè	**192.168.1.32 / 27**
	o anche	**192.168.1.32 mask 255.255.255.224**
	30 host da	**192.168.1.33** a **192.168.1.62**
	indirizzo di broadcast	**192.168.1.63**
192.168.1.010xxxxx	cioè	**192.168.1.64 / 27**
	o anche	**192.168.1.64 mask 255.255.255.224**
	30 host da	**192.168.1.65** a **192.168.1.94**
	indirizzo di broadcast	**192.168.1.95**

In rete si possono comunque trovare diversi esercizi per esercitarsi con a tecnica del subnetting.

IPV6 (**I**nternet **P**rotocol **V**ersione **6)**

La versione del protocollo IPV6 è stata introdotta per ovviare ai limiti dimensionali della IPV4.

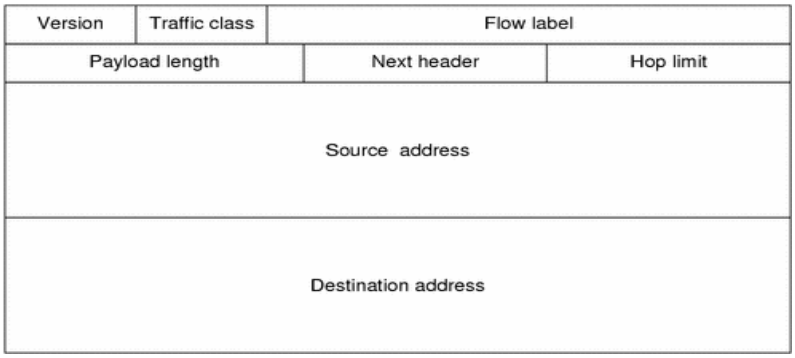
In questo caso vengono infatti riservati 128 bit agli indirizzi, contro i 32 che venivano utilizzati nell'IPV4.

Questi indirizzi sono rappresentati da 8 gruppi di 4 cifre esadecimali scritte in minuscolo:

ad esempio **2001:0db8:85a3:0000:1319:8a2e:0370:7344** è un indirizzo IPV6 valido.

Gli ultimi 32 bit possono comunque essere espressi in decimale.

La struttura dell'header dell'IPV6 è la seguente:



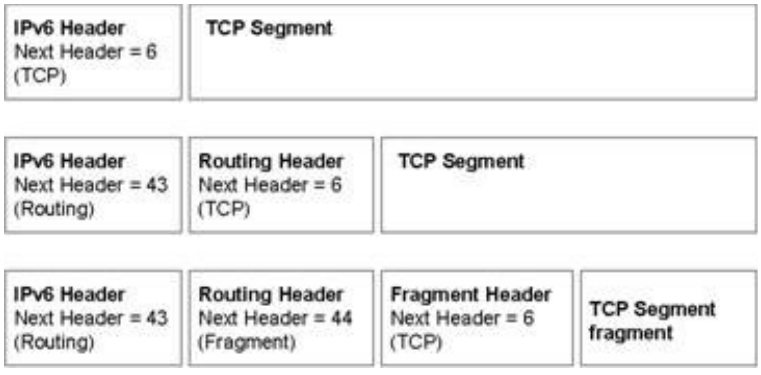
Rapidamente vediamo il significato di ogni campo:

- **Version** 4 bit — numero corrispondente alla versione dell'IP, in questo caso 6.
- **Traffic Class** 8 bit — tipo di traffico, tempo reale, voce…..
- **Flow Label** 24 bit — qualità del servizio
- **Payload Lenght** 16 bit — lunghezza del pacchetto in ottetti
- **Next Header** 8 bit — tipo di protocollo che segue l'intestazione dell'IP (TCP, UDP….)
- **Hop Limit** 8 bit — numero massimo di salti (tra router) che un pacchetto può fare
- **Source Address** 128 bit — indirizzo del sorgente
- **Destination Address** 128 bit — indirizzo del destinatario

A differenza dell'IPV4 non ci sono i campi per la frammentazione e l lunghezza minima del datagramma è passata da 576 a 1280 byte, se il router riceve un datagramma più grande restituisce un errore all'Host che dovrà frammentarlo in datagrammi più piccoli. Anche il campo di checksum è stato tolto.

L'IPV6 prevede inoltre un **Extension Header** per aggiungere informazioni al campo di header. Questi dati seguiranno l'intestazione standard.

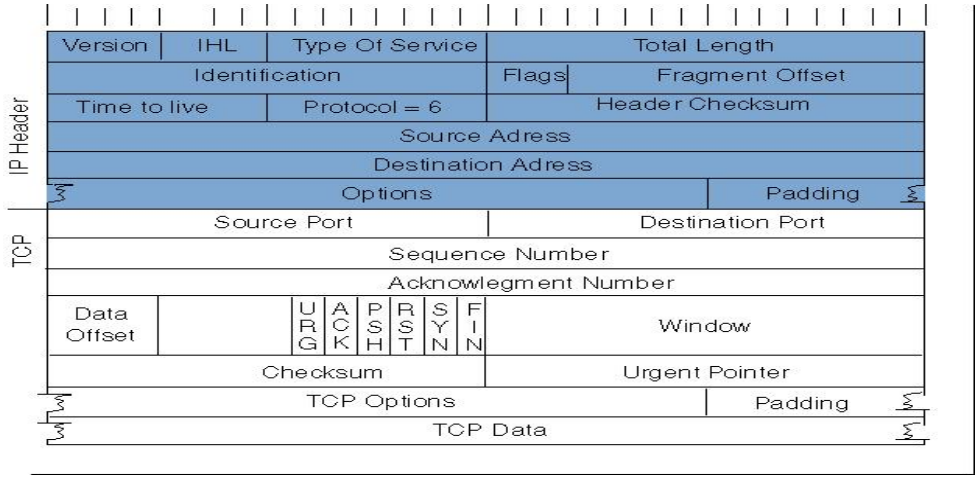
Ogni pacchetto può portare più intestazioni di estensioni successive, identificate dal campo Next Header.



TCP (**T**ransfer **P**rotocol **U**nit)

E' un protocollo di rete a pacchetto a livello di trasporto. Viiene utilizzato dalla rete internet insieme al protocollo IP. Il pacchetto dati utilizzato nel TCP, è il **PDU** (**P**rotocol **D**ata **U**nit).

Il PDU viene poi imbustato (cioè inserito) nel datagramma IP che aggiunge il suo header, cioè la sua intestazione.



Analizziamo ora il PDU del pacchetto TCP.

OFFSET		**0**								**1**								**2**								**3**							
bytes	bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source Port																Destination Port															
4	32	Sequence Number																															
8	64	Acknowledgement number																															
12	96	Data Offset				Reserved 0 0 0 0				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent Pointer															
20	160	Option																															
20/24	160/192	DATA																															

- **Source port e destination port** - 16 bit per ognuno, *sono i numeri della porta del PC sorgente e del PC destinazione.*
- **Sequence Number** - *un campo da 32 bit che contiene il numero del byte dell'intero stream. Nel TCP dati sono coniderati come un unico flusso dati (stream) diviso in segmenti, in questo campo si trova il il numero del primo segmento, ad esempo se abbiamo una suddivisione di segmenti in 1500 byte ciascuno, il primo segmento conterrà in questo campo 0, il secondo 1500 e così via.*
- **Acknlowledgment number** - *è il campo dove il ricevente inserira il numero del byte successivo al segmento ricevuto. Questo è attivo solo se il bit ACK è 1.*

- **Data Offset** - *occupa 4 bit ed indica la lunghezza dell'intestazione in parole da 32 bit.*

- **Reserved** - *occupa 4 bit ed è predisposto per usi futuri, attualmente non utilizzato.*

- **Flags** - *contiene le opziono di controllo del protocollo secondo 8 bit.*
  - ***CWR*** *(**C**ongestion **W**indow **R**educer) conferma la ricezione di un segmento on il bit ECE (successivo) ad 1.*
  - ***ECE*** *(***E**xplicit **C**ongestion notification **E**cho) se è ad 1 iindica che viene gestita la notifica della congestione. In pratica il meccanismo può prevenire la congestione dei dati con una notifica.*
  - ***URG*** (**URG**ente) *indica che nel flusso dati ci sono dati urgenti posizionati all'offset indicato nel capo Urgent Pointer.*
  - **ACK** (**ACK**nowledgment) *se è 1 indica che il campo Acknlowledgment number è valido*
  - **PSH** (**P**u**SH**) indica a chi riceve che i dati in arrivo devono essere passati direttamente al livello superiore.
  - **RST** (**R**e**S**e**T**) *inizializza la connessione TCP, questo può avvenire a causa di un errore, se utilizzato insieme al fag ACK può chiudere la connessione.*
  - **SYN** (**SYN**cronize) *indica che l'host mittente vuole aprire una connessione TCP con il destinatario, ed in Sequence Number mette il valore iniziale. La richiesta di connessione inizia con SYN=1 e ACK=0.*
  - **FIN** *viene usato per chiudere la connessione TCP, il mittente attende conferma con FIN+ACK.*

- **Window Size** - *occupa 16 bit ed indica il numero di byte che il mittente è in grado di ricevere partendo dall'acknowledgment number.*

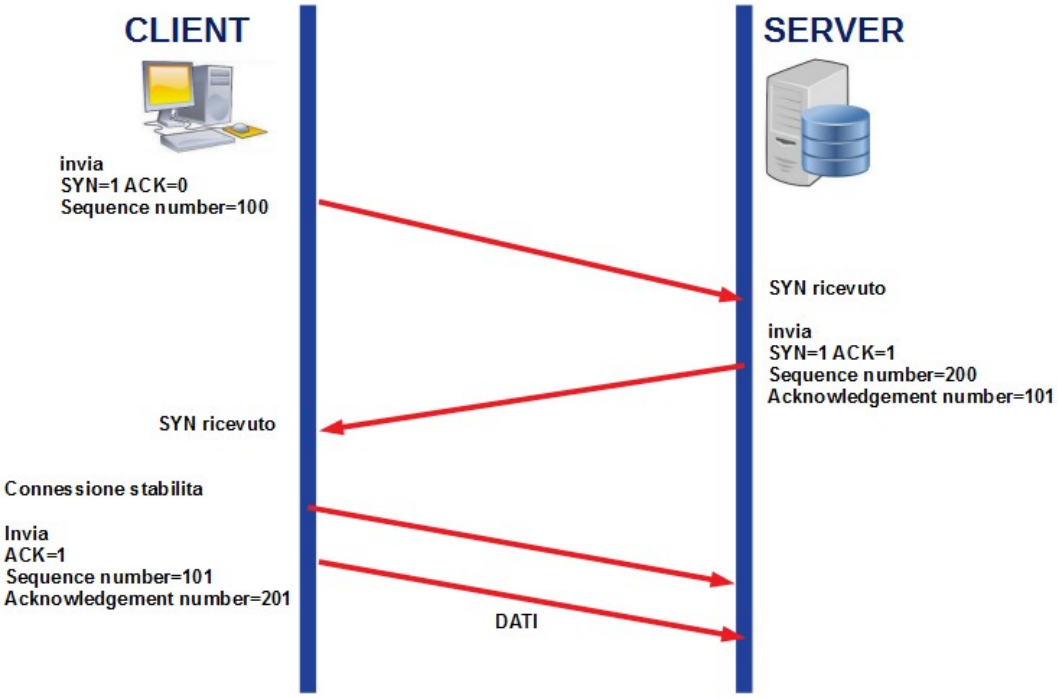
- **Checksum** - *occupa 16 bit ed è il campo per realizzare il controllo di errori.*

- **Urgent Pointer** - *occupa 16 bit ed è il puntatore ai dati urgenti indicato nella descrizione del flag URG.*

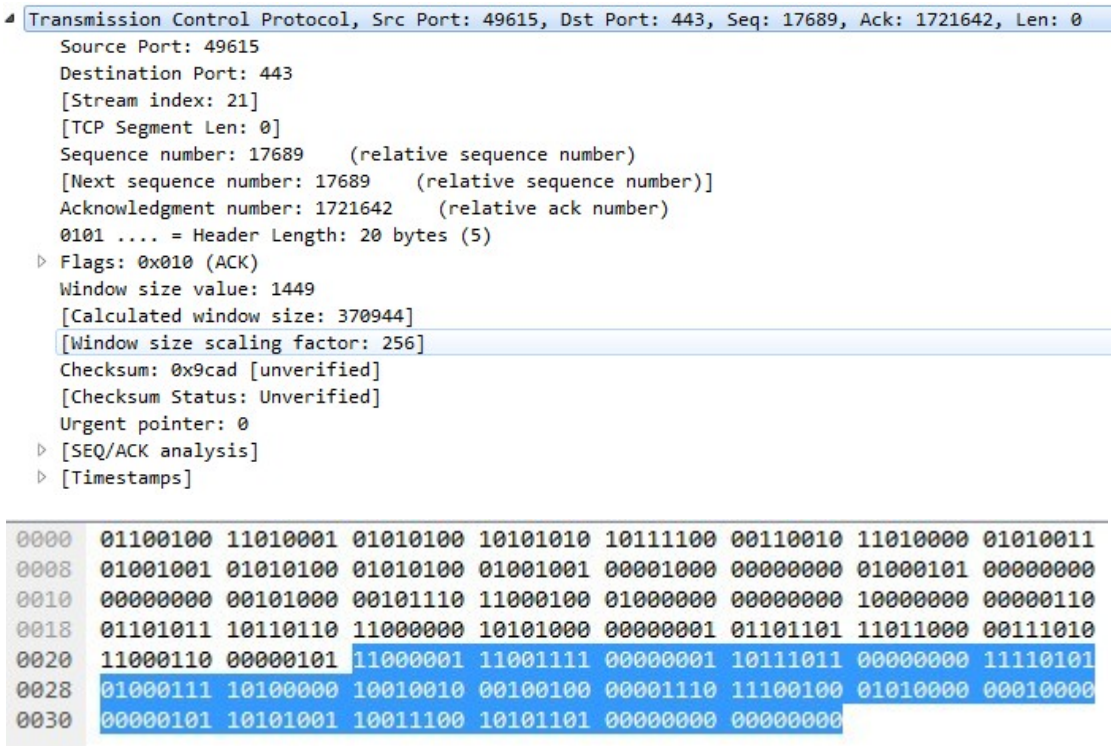
- **Options** - *campo facoltativo per usi del protocollo avanzati.*

- **Data** - *dati provenienti dal livello superiore.*

Una connessione TCP viene aperta con un metodo chiamato **three-way handshake** schematizzato come segue.

- 

Ecco come si presenta un PDU ed evidenziati i suoi relativi bytes



UDP (**U**ser **D**atagram **P**rotocol)

Oltre al protocollo TCP al livello trasporto troviamo anche il protocollo **UDP** definito in **RFC 768**.

A differenza del TCP, questo protocollo non prevede conferma dei segmenti spediti, non li riordina e non fornisce una connessione come nel caso del TCP. Viene utilizzato tutte le volte che si vuole privilegiare una connessione veloce senza considerare però la sua affidabiità. Si utilizza ad esempio nella trasmissioni audio e video real time, come anche nella risoluzione dell'indirizzo di un sito tramite comunicazione con server DNS.

Il pacchetto usato dal protocollo è molto povero, prevede la porta del mittente del destinatario, la lunghezza ed il valore di checksum per il controllo degli errori.

+	Bit 0-15	Bit 16-31
0	Source port	Destination port
32	Lenght	Checksum
64	Data	

ICMP (**I**nternet **C**ontrol **M**essage **P**rotocol)

Questo è un protocollo di servizio per le reti a pacchetto, viene utilizzato da diversi applicativi di rete, tra cui PING, l'applicazione che invia un pacchetto dati al fine di verificare il corretto collegamento e configurazione della rete, e TRACEROUTE, che invece si occupa di ricavare il percorso di un pacchetto nella rete, ricavando così l'indirizzo IP di ogni router attraversato per raggiungere il destinatario.

Di seguito due esempi dei comandi descritti:

Ping all'indirizzo 92.168.1.252 vengono inviati dei pacchetti ICMP e viene misurato il tempo di risposta.

```
PING 192.168.1.252 (192.168.1.252) 56(84) bytes of data.
64 bytes from 192.168.1.252: icmp_seq=1 ttl=64 time=4.86 ms
64 bytes from 192.168.1.252: icmp_seq=2 ttl=64 time=4.74 ms
64 bytes from 192.168.1.252: icmp_seq=3 ttl=64 time=28.5 ms
64 bytes from 192.168.1.252: icmp_seq=4 ttl=64 time=4.42 ms
64 bytes from 192.168.1.252: icmp_seq=5 ttl=64 time=4.48 ms
^C
--- 192.168.1.252 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 4.423/9.395/28.479/9.543 ms
root@Kalisys:~# _
```

Traceroute all'URL www.google.it. Il risultato è il percorso del pacchetto ICMP, con tutti i router attraversati.

```
root@Kalisys:~# traceroute www.google.it
traceroute to www.google.it (216.58.205.163), 30 hops max, 60 byte packets
 1  antenna.fidoka.it (192.168.1.251)  5.547 ms  5.465 ms  5.408 ms
 2  10.0.0.2 (10.0.0.2)  38.387 ms  38.373 ms  38.326 ms
 3  10.127.179.74 (10.127.179.74)  38.278 ms  39.960 ms  39.883 ms
 4  ip203-44.fastnet.it (195.96.203.44)  39.929 ms  39.882 ms  39.815 ms
 5  ip203-43.fastnet.it (195.96.203.43)  39.773 ms  39.725 ms  39.679 ms
 6  google.mix-it.net (217.29.66.96)  43.252 ms  31.131 ms  31.054 ms
 7  108.170.245.81 (108.170.245.81)  31.003 ms 108.170.245.65 (108.170.245.65)  26.149 ms 108.170.24
5.81 (108.170.245.81)  24.805 ms
 8  216.239.42.23 (216.239.42.23)  24.429 ms 216.239.42.21 (216.239.42.21)  22.035 ms 216.239.42.23
(216.239.42.23)  25.758 ms
 9  mil04s28-in-f163.1e100.net (216.58.205.163)  24.955 ms  25.058 ms  24.810 ms
root@Kalisys:~# _
```

PORTE

Con questo termine non intendiamo le porte di comunicazione fisiche di un PC, ma del canali software per l'accesso alle varie applicazioni.

Supponiamo di avere un host con un determinato indirizzo ad esempio il classico 192.168.1.10.

Questo host, che può essere un PC o un server, riceve da remoto richieste da diverse applicazioni, magari riceve in maniera consecutiva datagrammi relativi al contenuto di una pagina web, o di un flusso video, o di una email.

Il livello rete ignora a chi sono destinati i datagrammi e per questo non può scegliere a quale applicazione indirizzarli, anche perché non ne avrebbe la possibilità. Il livello rete però identifica quale protocollo utilizzare ad esempio TCP o UDP, mediante l'apposito campo.

A questo punto il problema si sposta al livello trasporto il quale riuscirà ad indirizzare i dati all'applicazione giusta, facendoli passare per la porta corretta utilizzata da quell'applicazione. Il valore della porta è infatti contenuto nel segmento TCP come in quello UDP.

In un host, ci sono 65536 porte indicate con un numero che va da 0 a 65535.

Le porte sono assegnate a specifici servizi dallo **IANA** (**I**nternet **A**ssigned **N**umbers **A**uthority) lo stesso organismo che si occupa di assegnare gli indirizzi IP.

In rete possiamo trovare l'assegnazione delle porte alle varie applicazioni, è inutile ripeterle qui, anche perché sono veramente tante. Possiamo solo menzionarne qualcuna:

Port 21 utilizzata dalle applicazioni per il trasferimento dei file con l'**FTP** (**F**ile **T**ransfer **P**rotocol)
ad esempio il software Filezilla utilizza questa porta.

Port 25 per il trasferimento delle mail secondo il protocollo **SMTP** (**S**imple **M**ail **T**rasnfer **P**rotocol) utilizzato per inviare messaggi verso i server di posta.

Port 110 utilizzata sempre per il trasferimento delle mail ma utilizzando il protocollo **POP3**
(**P**ost **O**ffice **P**rotocol) che i client utilizzano per ricevere i messaggi dai server di posta.

Port 143 utilizzata per il trasferimento delle mail utilizzando il protocollo **IMAP**
(**I**nternet **M**essage **A**ccess **P**rotocol) sempre utilizzato dai client per ricevere la posta.

Port 80 utilizzata nei server per i servizi World Wide Web, che utilizzano l'**HTTP**
(**H**yper **T**ext **T**ransfer **P**rotocol)

Port 443 utilizzata nei server che che utilizzano l'**HTTPS** (**H**yper **T**ext **T**ransfer **P**rotocol **S**ecure)
il protocollo HTTPS garantisce una navigazione sicura.

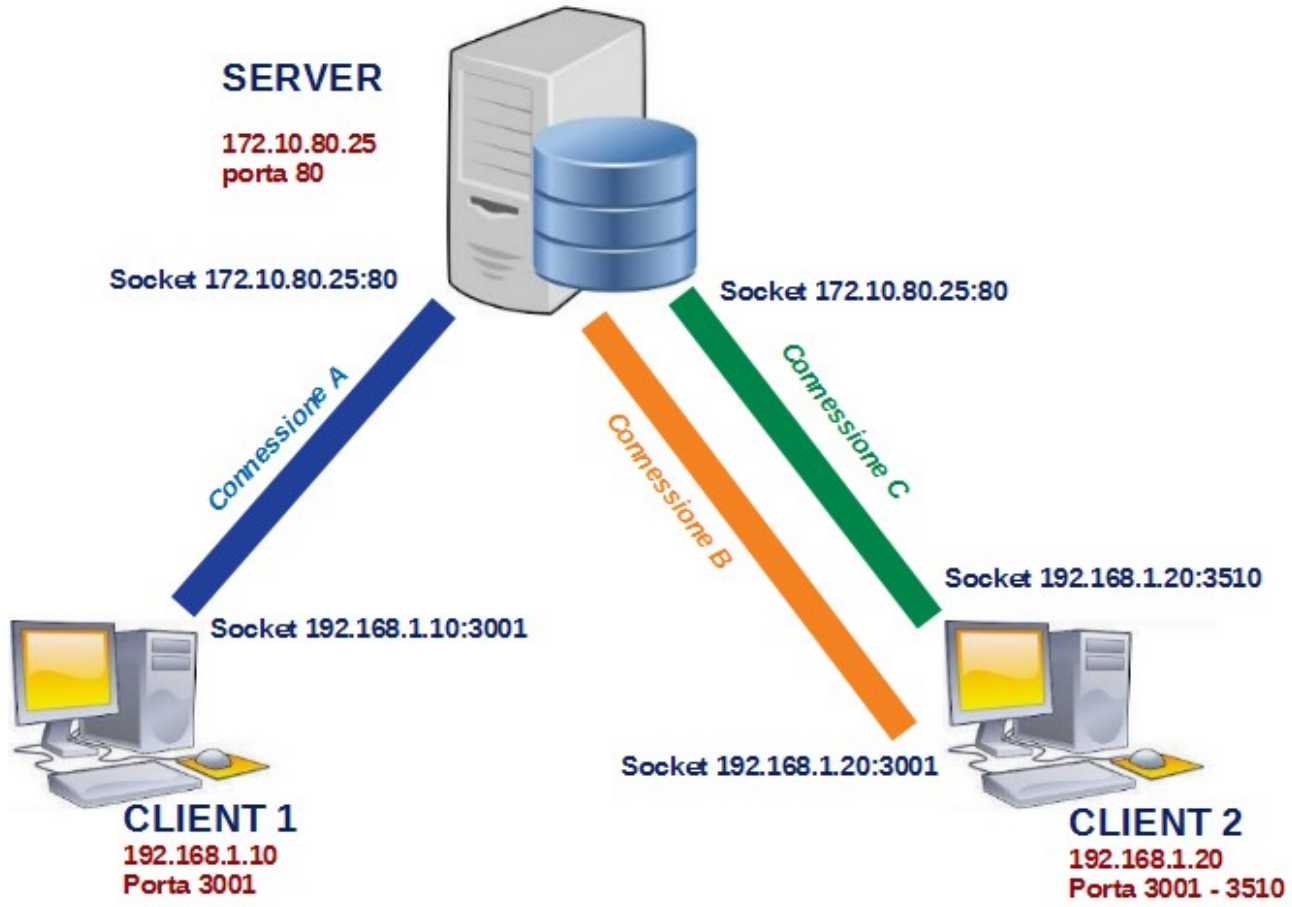
Su wikipedia troviamo un esaustivo elenco https://it.wikipedia.org/wiki/Lista_di_porte_standard

---

SOCKET

Per accedere ad un determinato servizio, occorre conoscere l'indirizzo IP e la porta utilizzata.

L'insieme di queste due informazioni viene definito Socket. Tramite il socket si può instaurare un canale di comunicazione tra due applicazioni presenti su due host come ad esempio un client ed un server.



Nell'immagine sopra, i Client instaurano con il server, tramite i socket, 3 canali di comunicazione;

connessione A, B e C.

I pacchetti che arrivano al server sulla porta 80 vengono distinti, perché anche se hanno la stessa porta sorgente (3001) hanno un indirizzo IP diverso. Il server riesce perciò sempre a discriminare la sorgente che sarà poi la destinazione delle sue risposte.

HTTP (**H**yper **T**ext **T**ransfer **P**rotocol)

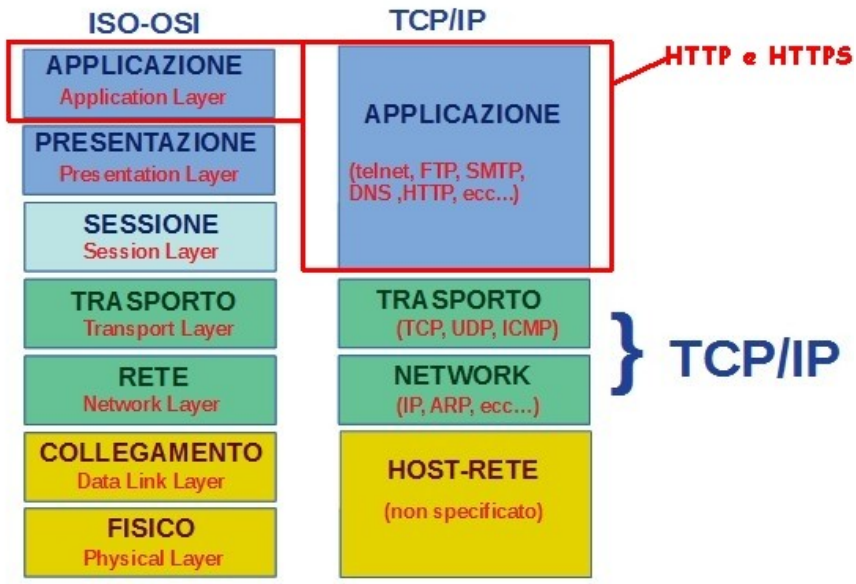
E' un protocollo tipico di un'architettura client-server, ed è utilizzato per trasmettere **ipertesti** (pagine web).

Il client invia mediante questo protocollo un messaggio di richiesta contenente, oltre all'indirizzo (**URL U**niform **R**esource **L**ocator) anche uno dei seguenti metodi:

- **GET**
- **POST**
- **HEAD**
- **PUT**
- **DELETE**
- **TRACE**
- **OPTION**
- **CONNECT**

Per fare in modo che la richiesta del client inviata sulla porta 80 del Server venga soddisfatta, è indispensabile che sul Server sia attivo un "**demone**" in ascolto. Un demone è un software che si comporta come fosse il custode della porta a lui assegnata. La gran parte del tempo, è dormiente con l'orecchio appoggiato alla sua porta, appena arriva la richiesta (in questo caso la richiesta di una pagina del sito) lui si sveglia e risponde alla richiesta. Il demone HTTP è attivo su ogni server Web. Uno dei server più comuni per rendere un qualsiasi PC un server Web è "Apache HTTP Server" liberamente scaricabile ed installabile.

Il protocollo HTTP lavora a livello applicazione



La classica struttura CLIENT – SERVER è schematizzata sotto.

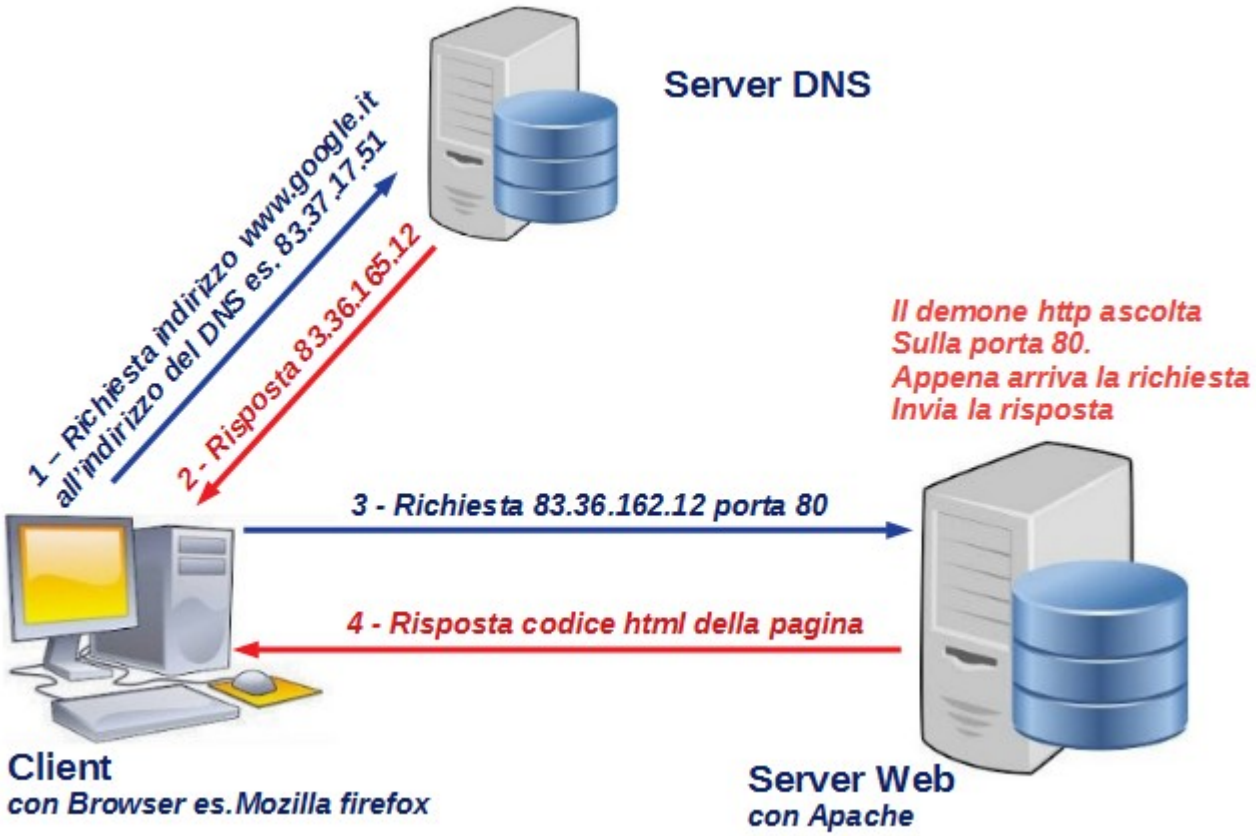
Il meccanismo è semplicemente composto da due fasi;

1) Richiesta del Client
2) Risposta del Server

Apriamo ora una piccola parentesi.

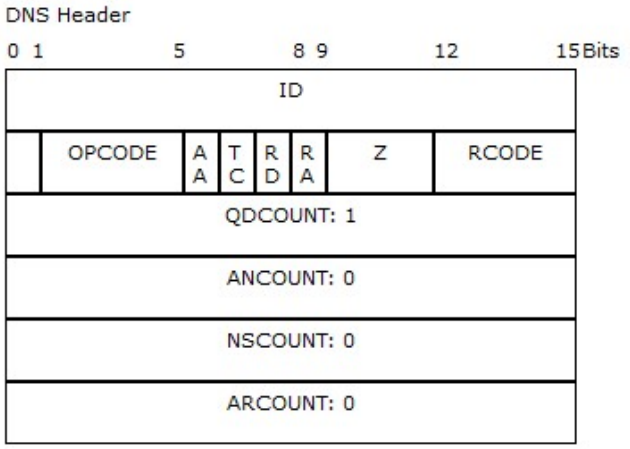
Nella realtà il meccanismo è un po' più complesso, perché la richiesta ad un sito necessitò dell'indirizzo IP del server Web di quel sito, pertanto prima bisogna accedere ad un server **DNS** (**D**omain **N**ame **S**ystem) di cui conosciamo l'indirizzo (ad esempio **8.8.8.8** il servizio **DNS** offerto da google, oppure **83.37.17.51** offerto da TIM). Perciò le fasi sono le seguenti:

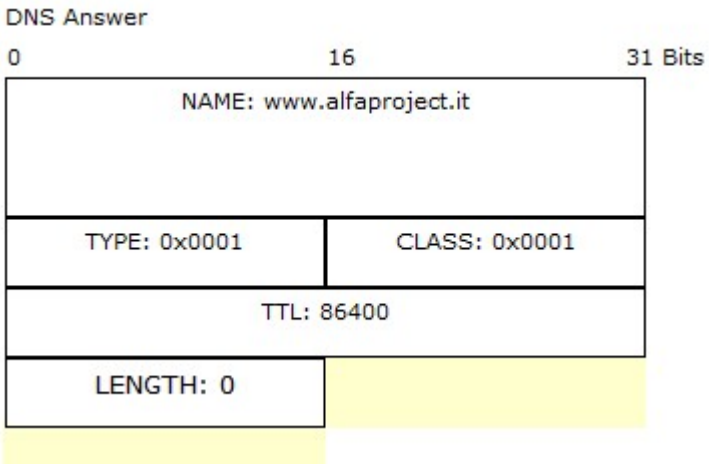
1) Apro nel mio PC Mozilla Firefox e digito l'indirizzo del sito che voglio raggiungere (URL).
2) Il mio browser invia una richiesta DNS all'indirizzo **83.37.17.51**, dove c'è un Server che conosce l'associazione **URL – Indirizzo IP** di tutti i siti.
3) Il server DNS risponde con l'indirizzo IP del sito richiesto ad esempio **83.36.165.12** che è l'indirizzo fisico del Server Web che ospita iil sito.
4) Il browser invia ora una richiesta all'indirizzo del Server Web alla porta 80, con protocollo http.
   Il socket utilizzato sarà in questo caso **83.36.165.12 : 80.**
5) Il demone in ascolto sulla porta 80 del server web riceve questa richiesta e la soddisfa rispondendo con la pagina HTML del sito richiesto.



Di seguito la struttura di un header DNS estratta da una simulazione fatta con Cisco Packet tracer, dove si chiedeva l'indirizzo l'IP di un sito (ovviamente simulato) www.alfaproject.it.
Si possono approfondire i significati dei vari campi cercando in rete "dns header" oppure al link dello **IANA**:
https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

La struttura a sinistra è la richiesta del CLIENT, a destra invece la risposta del server DNS.

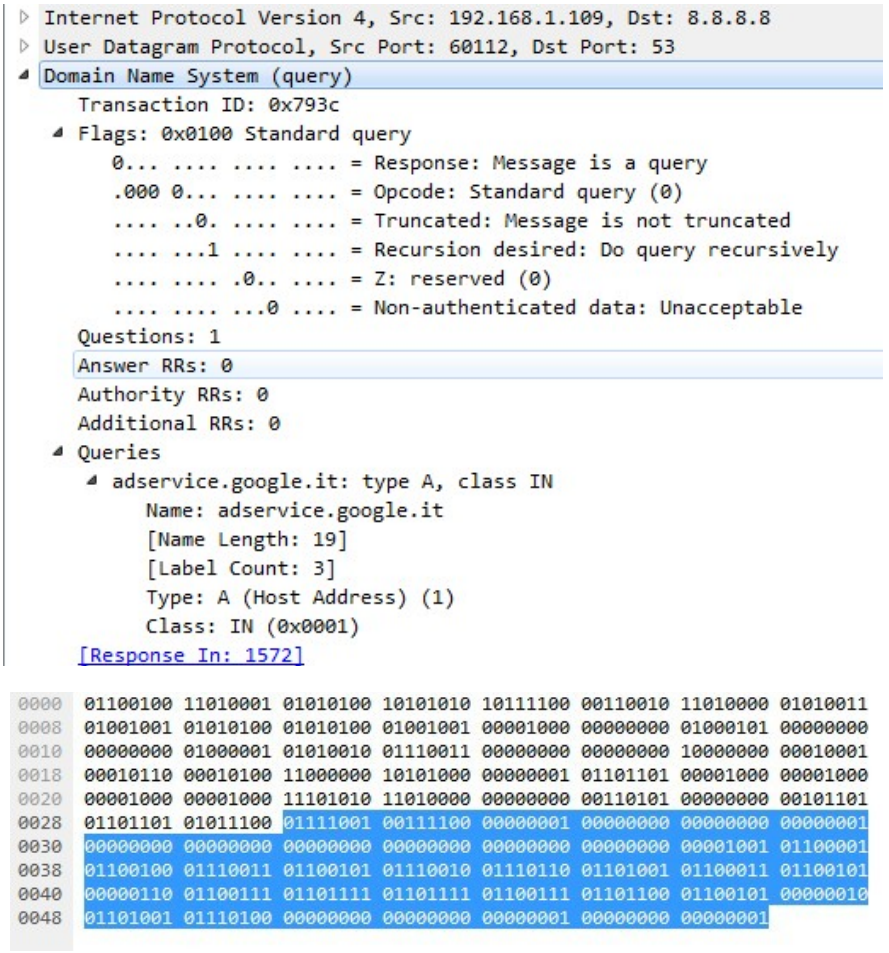




Con Wireshark invece possiamo vedere la struttura di una richiesta (a destra) o della risposta del DNS in maniera molto semplice.

Evidenziati in basso i bytes del DNS imbustati nel frame dei dati.

Quanto detto finora è è una descrizione molto sintetica, che ci ha permesso di conoscere il servizio DNS, per i nostri scopi è comunque sufficiente conoscere quanto detto.
Per chi volesse approfondire il discorso, esistono diversi siti che descrivono ogni dettaglio di questo protocollo.



Possiamo chiudere la parentesi sul DNS.

Torniamo ora al protocollo HTTP, come detto inizialmente, nel messaggio di richiesta al Server viene indicato un metodo, i metodi possibili sono quelli elencati prima e cioè:

- **GET** *Richiesta di lettura di un pagina o di una risorsa WEB tramite una "**query string**" cioè una stringa codificata in caratteri **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange un codice che rappresenta ogni carattere o simbolo, con un numero binario ad 8 bit).*

  La query string si aggiunge all'URL e la possiamo vedere nel seguente esempio.

```
Frame 14: 568 bytes on wire (4544 bits), 568 bytes captured (4544 bits) on interface 0
Ethernet II, Src: LiteonTe_54:54:49 (d0:53:49:54:54:49), Dst: Raspberr_f5:5d:3a (b8:27:eb:f5:5d:3a)
Internet Protocol Version 4, Src: 192.168.1.109, Dst: 192.168.1.94
Transmission Control Protocol, Src Port: 51390, Dst Port: 80, Seq: 1, Ack: 1, Len: 514
Hypertext Transfer Protocol
  GET /form.html HTTP/1.1\r\n
  Host: 192.168.1.94\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7\r\n
  If-None-Match: "1461125039"\r\n
  If-Modified-Since: Mon, 11 Feb 2019 14:19:58 GMT\r\n
  \r\n
  [Full request URI: http://192.168.1.94/form.html]
  [HTTP request 1/3]
  [Next request in frame: 17]
```

```
0030  00000001 00000000 10100001 01101011 00000000 00000000 01000111 01000101   ···k··GE
0038  01010100 00100000 00101111 01100110 01101111 01110010 01101101 00101110   T /form.
0040  01101000 01110100 01101101 01101100 00100000 01001000 01010100 01010100   html HTT
0048  01010000 00101111 00110001 00101110 00110001 00001101 00001010 01001000   P/1.1··H
0050  01101111 01110011 01110100 00111010 00100000 00110001 00111001 00110010   ost: 192
0058  00101110 00110001 00110110 00111000 00101110 00110001 00101110 00111001   .168.1.9
0060  00110100 00001101 00001010 01000011 01101111 01101110 01101110 01100101   4··Conne
```

La query string inizia con la riga evidenziata dove viene innazitutto specificato il metodo utilizzato.

*In questo caso è stato richiesto al server di aprire un file che si chiama **form.html**, e che è memorizzato nel server che ha indirizzo **192.168.1.94**, l'esempio è stato fatto navigando in locale verso un Server Web.*

*I simboli **\r \n** significano rispettivamente "**carriage return**" (ritorno ad inizio linea) e **"line feed"** (avanzamento alla riga successiva).*

Messi uno di seguito all'altro, indicano di andare a capo all'inizio della linea, come avviene col tasto invio.

Nella query string ci sono anche altre informazioni e dei campi opzionali non sempre utilizzati.

Bisogna inoltre ricordare che nella codifica ASCII,ogni carattere ha un suo codice binario, anche lo spazio.

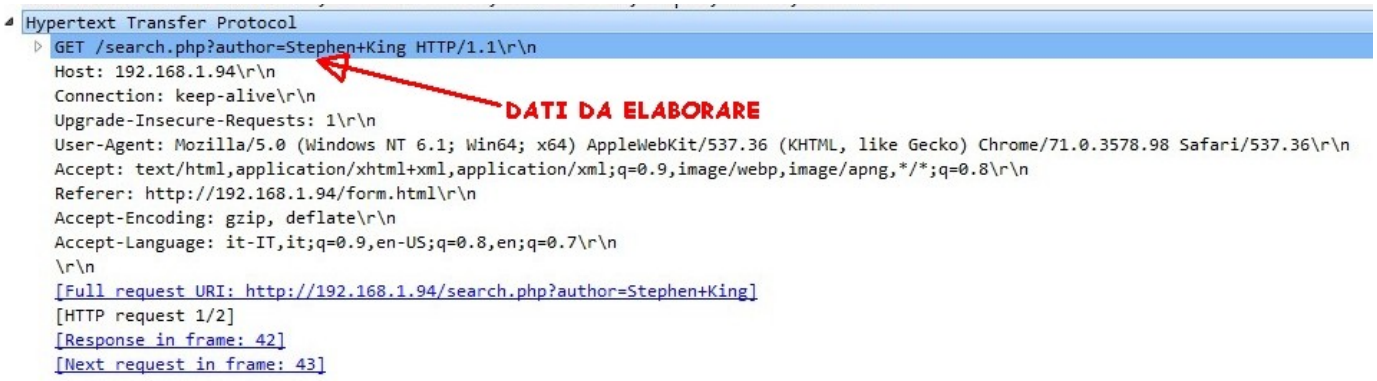
*Pertanto ad ogni **spazio** verrà associato il corrispondente codice che è **00100000 (20h)**.*

La risposta del server a questa richiesta è la seguente e conterrà oltre ai campi previsti dal protocollo, anche il contenuto della pagina web richiesta, che verrà interpretato dal nostro Browser visualizzando la pagina Web scritta in HTML.

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Content-type: text/html; charset=UTF-8\r\n
  Content-Length: 405\r\n
  Date: Mon, 11 Feb 2019 15:04:47 GMT\r\n
  Server: lighttpd/1.4.45\r\n
  \r\n
  [HTTP response 3/3]
  [Prev request in frame: 8]
  [Prev response in frame: 14]
  File Data: 405 bytes
Line-based text data: text/html (6 lines)
  \n
      <html><head>\n
          <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1"/>\n
          <link rel='stylesheet' href='/pihole/blockingpage.css' type='text/css'/>\n
      </head><body id='splashpage'><img src='/admin/img/logo.svg'/><br/>Pi-<b>hole</b>: Your black hole for
```

Contenuto del file HTML richiesto

Nel metodo GET visto precedentemente, nella query string troviamo tutte le informazioni, e se avessimo inviato alla pagina anche dei dati da elaborare, sarebbero stati visibili nella query string.



Nella figura sopra utilizzando il metodo GET, chiediamo al Server, che ha indirizzo 192.168.1.94, di elaborare una stringa di caratteri che contiene il nome di un autore, tramite un file che si chiama ***search.php****.*

E' il classico esempio che avviene se devo cercare un nome in un Elenco di nomi.

Invio il nome da cercare e chiedo di avviare il programma che fa la ricerca, che in questo caso è scritto in linguaggio ***PHP*** *(un linguaggio di scripting interpretato, per realizzare pagine web dinamiche).*

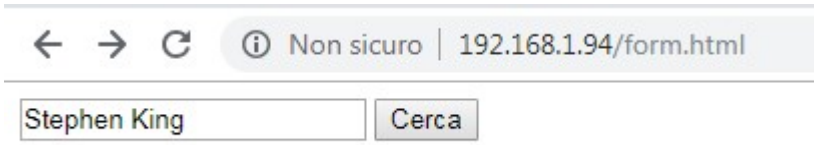
In questo caso il nome che ho inviato (cioè il dato da elaborare) appare nella query string.

La query string può essere anche digitata sulla barra degli indirizzi secondo il seguente formato:

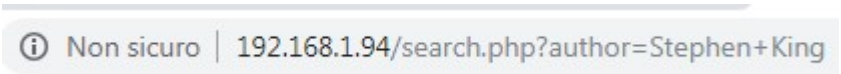
http://server/percorso/programma?campo1=valore&campo2=valore2&campo3=valore3

Nel nostro caso avendo un unico campo sarebbe: http://192.168.1.94/search.php?author=Stephen+King

In genere non avviene che si scriva la query string nella barra degli indirizzi, ma avviene in automatico lanciando un file html con uno o più ***form***



Cliccando sul tasto cerca, il form invierebbe la query string, che in ogni caso sarebbe visibile sulla barra degli indirizzi.

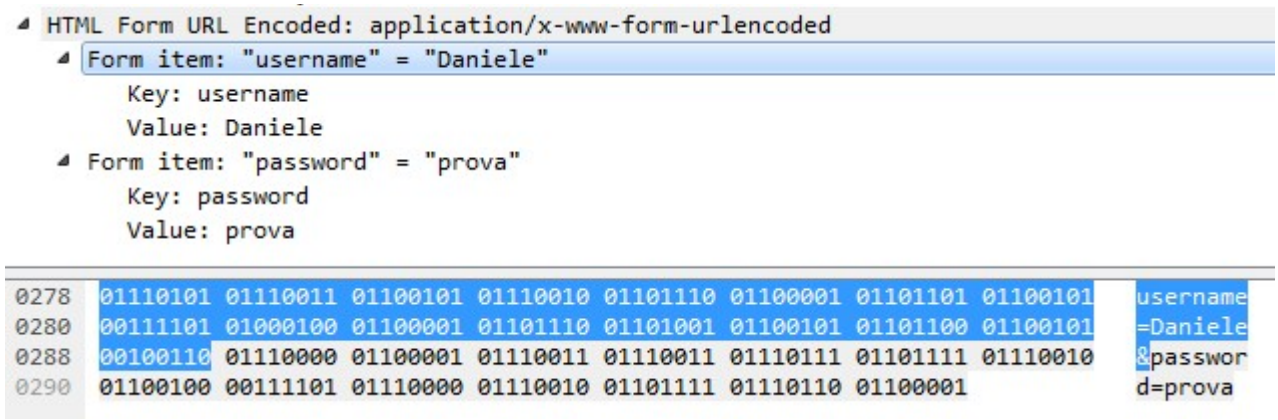


A meno che non vengano inviati dati non importanti, il metodo GET non si utilizza per il passaggio di parametri al Server, per questo scopo è più adatto il metodo POST.

- **POST** *Con il metodo **POST** questo non accade. **Esso viene utilizzato quando devo inviare informazioni al Server, queste informazioni non risultano visibili nella query string** e pertanto non possono essere tracciate e visualizzate. Classico esempio è quello di un'accesso con nome utente e password.*

*Nell'esempio sopra, viene chiesto al server Web che ha indirizzo 192.168.1.94, di elaborare un nome utente ed una password tramite il programma **register.php**. Come si può notare nella query string non appare il nome utente e nemmeno la password. **I parametri vengono comunque inviati al Server ma sono contenuti nel body dell'HTTP, per questo non sono visibili nella query string.***

Sempre con Wireshark, possiamo vedere che il nome utente e la password sono contenuti nei dati inviati con l'HTTP.



Come detto nel metodo GET, anche in questo caso, occorre avere una pagina HTML con dei form, per preparare ed inviare la richiesta con il metodo POST. Di seguito il codice di una pagina di esempio, e il risultato sul Browser.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    <form action="register.php" method="post">
        <input type="text" name="username" placeholder="Inserisci username" /><br>
        <input type="password" name="password" placeholder="Inserisci password" /><br>
        <input type="submit" value="Registrati" />
    </form>
</body>
</html>
```

Inserendo username e password e cliccando su registrati, I dati verranno inviati e sull'URL non comparirà nulla.

Con il metodo POST inoltre non c'è un limite predefinito sulla lunghezza dei dati che possono essere inviati anche in binario.

*Oltre ai dati, con il metodo POST posso anche inviare delle **variabili d'ambiente**, inviate nelle righe successive all'intestazione del messaggio. Le variabili d'ambiente sono delle variabili globali del sistema che si sta utilizzando. Vediamone qualcuna.*

- *HTTP_CONNECTION* — *contiene informazioni sulla connessione tra client e server*
- *HTTP_ACCEPT* — *tipi MIME ch il browser può accettare*
- *HTTP_ACCEPT_ENCODING* — *tipi di compressione dati accettata dal browser*
- *HTTP_ACCEPT_LANGUAGE* — *lingua accettata dal browser*
- *CONTENT_LENGHT* — *lunghezza in byte dei dati inviati*
- *CONTENT_TYPE* — *tipo di contenuti inviati nel body, se presenti.*
- *REMOTE_ADDR* — *indirizzo IP della macchina che ha inviati I dati*
- *REMOTE_PORT* — *numero della porta della macchina che ha inviato I dati*
- REQUEST_METHOD — tipo di metodo utilizzato
- SERVER_NAME — nome del server da cui viene eseguita l'applicazione
- SEVER_PORT — porta del server che esegue l'applicazione
- SERVER_SOFTWARE — Server Web utilizzato, ad esempio Apache

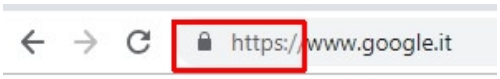
Sicuramente il metodo GET ed il metodo POST sono quelli più utilizzati nel protocollo HTTP.

Potremmo continuare con altre informazioni, ma lo scopo non è approfondire in maniera dettagliata questo protocollo, ma acquisire la conoscenza necessaria per gestire gli strumenti di analisi delle reti.

Vediamo velocemente gli altri metodi.

- **HEAD** *Chiede al server solo l'header del protocoll HTTP. Viene utilizzato in fase di diagnostica.*
- **PUT** *Esegue l'upload di un file sul server, creandolo come nuovo o sovrascrivendolo con il nome indicato ed I contenuti specificati nella parte che segue l'header.*
- **DELETE** *Cancella un file sul server, l'utente attivo nel server deve avere i permessi necessari.*
- **TRACE** *Chiede di ricevere copia della richiesta, incapsulata nel body della risposta HTTP.*
- **OPTION** *Richiede l'elenco dei metodi consentiti dal server.*
- **CONNECT** *Riservato all'uso di Server Proxy per creare un canale sicuro SSL.*

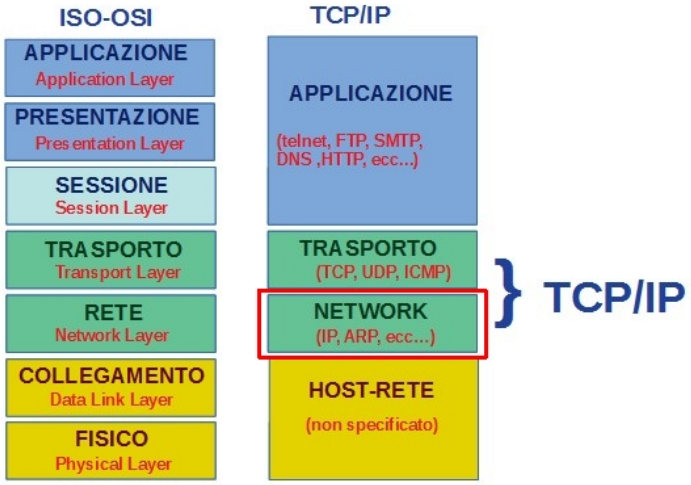
HTTPS (**H**yper **T**ext **T**ransfer **P**rotocol **S**ecure)



E' un estensione del protocollo HTTP, ed è usato per avere delle comunicazioni sicure.
Ormai gran parte dei siti oggi utilizzano questo protocollo.
La comunicazione viene criptata utilizzando **T**ransport **L**ayer **S**ecurity **(TLS**) o, in precedenza, il suo predecessore, **S**ecure **S**ockets **L**ayer (**SSL**) entrambi sono dei **protocolli crittografici**.
La motivazione principale per l'uso dell'HTTPS, è la protezione della privacy e dell'integrità dei dati scambiati. Protegge da attacchi e da tentativi di intercettazioni nella comunicazione. Inizialmente questo tipo di connessione si utilizzava principalmente per le transazioni monetarie fatte tramite internet, ma oggi è ormai utilizzata in tutti i principali siti.

Ci sono comunque diversi siti che non utilizzano il protocollo HTTPS, in questo caso può capitare che il broswer, chieda conferma. Comunque sia, sulla barra degli indirizzi verrà segnalata la connessione non sicura.

ARP (**A**ddress **R**esolution **P**rotocol)



Questo protocollo definito da RFC 826, opera a livello di rete. Il suo compito è fornire la mappatura tra indirizzo IP e MAC address di ogni dispositivo.
Ogni dispositivo tiene in memoria la mappatura in una apposita cache (memoria) chiamata **ARPcache.**

Il dispositivo host che vuol conoscere il MAC address di un altro dispositivo, invia in **broadcast** *(invia tutti i dispositivi della rete, utilizzando l'indirizzo di broadcast, che è l'ultimo possibile di una rete, ad esempio nella rete 192.168.1.0/24 l'indirizzo di broadcast è 192.168.1.255)* la richiesta (**ARP request)** contenente il proprio indirizzo MAC e l'indirizzo IP del destinatario di cui vuol conoscere il MAC. Ogni dispositivo riceverà il pacchetto e confronterà l'indirizzo IP con il proprio, se corrispondono, invierò una risposta al mittente (**ARP reply**) contenente il proprio MAC address.
Di seguito due pacchetti ARP a sinistra la richiesta del PC con indirizzo 192.168.1.2 (Source) di conoscere il MAC del dispositivo con indirizzo 192.168.1.1 (Target)

ARP Request

ARP

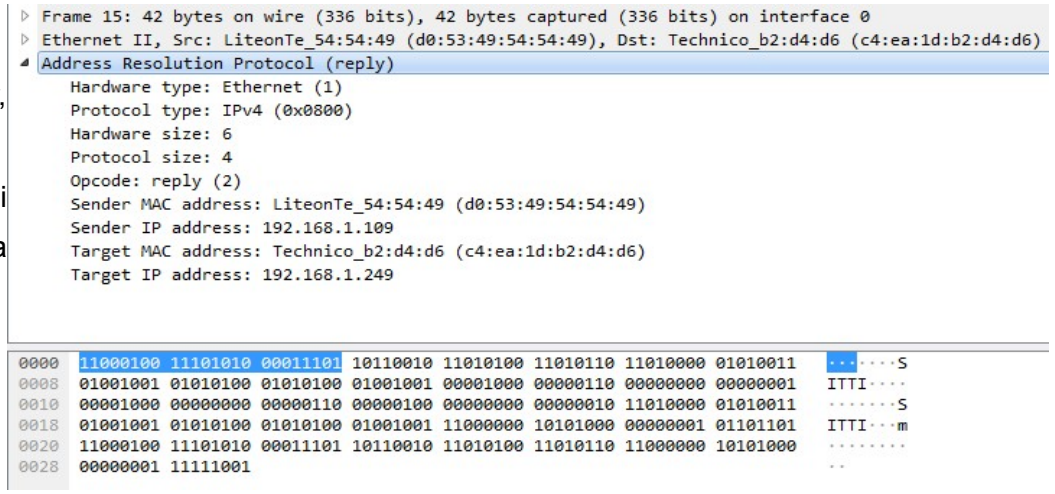
0	8	16	31 Bits
HARDWARE TYPE: 0x1		PROTOCOL TYPE: 0x800	
HLEN: 0x6	PLEN: 0x4	OPCODE: 0x1	
SOURCE MAC: 00D0.FF0B.34C9 (48 bits)			
		SOURCE IP (32 bits) ==>	
192.168.1.2			
TARGET MAC: 0000.0000.0000 (48 bits)			
TARGET IP: 192.168.1.1 (32 bits)			

ARP reply

ARP

0	8	16	31 Bits
HARDWARE TYPE: 0x1		PROTOCOL TYPE: 0x800	
HLEN: 0x6	PLEN: 0x4	OPCODE: 0x2	
SOURCE MAC: 0009.7C3A.D3CA (48 bits)			
		SOURCE IP (32 bits) ==>	
192.168.1.1			
TARGET MAC: 00D0.FF0B.34C9 (48 bits)			
TARGET IP: 192.168.1.2 (32 bits)			

A destra un pacchetto ARP reply, catturato con Wireshark.
Possiamo vedere gli stessi campi, presenti nella struttura vista sopra.



Esiste anche il Protocollo **RARP** (**R**everse **A**ddress **R**esolution **P**rotocol) che opera esattamente in senso opposto, cioè associa ad un indirizzo MAC l'indirizzo IP del dispositivo (mentre invece ARP associa ad un indirizzo IP l'indirizzo MAC del dispositivo).

DHCP (**D**ynamic **H**ost **C**onfiguration **P**rotocol)

Il protocollo **DHCP** è un protocollo per l'assegnazione dinamica degli indirizzi IP in una rete, esso opera a livello applicazione,

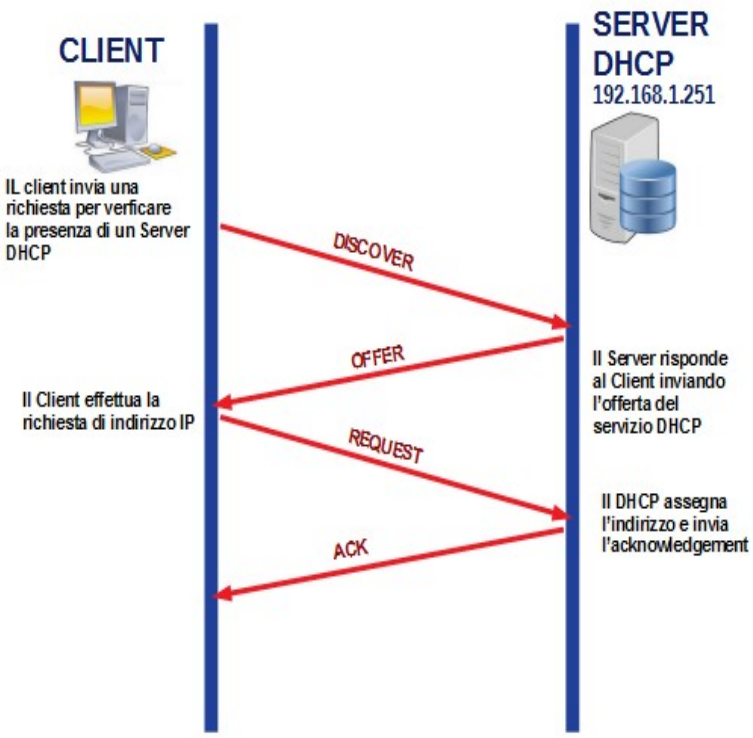
La comunicazione tramite protocollo DHCP avviene tra un **client** *(che vuole appartenere ad una rete e pertanto avere un proprio indirizzo IP)* ed un **server** (**server DHCP**) che si occupa di assegnare secondo delle regole predefinite l'indirizzo al dispositivo.

In una normale rete domestica wifi, questo compito viene svolto dal router, che offre questo servizio a tutti gli host della rete. Se in un Client è stata attivata l'assegnazione automatica degli indirizzi IP, all'accensione del dispositivo Client avverrà quanto descritto in figura che possiamo strutturare in 4 fasi:

1. **DISCOVER - scoperta**

   Non conoscendo l'indirizzo del server DHCP, il client invia a tutti gli apparati della rete mediante l'indirizzo generico di Broadcast 255.255.255.255 un messaggio di scoperta DHCP sulla porta 67 incapsulato in un datagramma IP con indirizzo sorgente 0.0.0.0.

   IP sorgente 0.0.0.0:68 IP destinazione 255.255.255.255:67



2. **OFFER – offerta**

   Il Server (o i server) DHCP presenti in rete, che hanno ricevuto la richiesta, rispondono con un offerta che contiene l'indirizzo IP proposto al Client, la subnet mask, ed il tempo di validità dell'indirizzo, scaduto il quale non sarà più valido per la rete.

   IP sorgente 192.168.1.251 IP destinazione 255.255.255.255:68
   IP offerto 192.168.1.109 DHCP Server 192.168.1.251
   Tempo di lease 3600 s

3. **REQUEST – richiesta**

   Il Client ricevuta l'offerta effettua seleziona l'IP tra quelli che gli sono stati offerti, ed invia un messaggio a tutti i server DHCP segnalando di aver accettato un'offerta con i parametri di configurazione.

   IP sorgente 0.0.0.0:68 IP destinazione 255.255.255.255.67 IP address 192.168.1.109 DHCPserver 192.168.1.251
   Tempo di lease 3600 s

4. **ACK – accettazione**

   I server prescelto invia, in broadcast, un messaggio di accettazione al client, con la conferma dei parametri scambiati.

   IP sorgente192.168.1.251 IP destinazione 255.255.255.255.67 IP address 192.168.1.109 DHCPserver 192.168.1.251
   Tempo di lease 3600 s

Di seguito vediamo lo scambio dei pacchetti dell'esempio precedente catturato tramite Wireshark.

```
   4 0.150650      0.0.0.0           255.255.255.255   DHCP      342 DHCP Discover - Transaction ID 0x3484293b
   5 0.402996      192.168.1.251     255.255.255.255   DHCP      342 DHCP Offer    - Transaction ID 0x3484293b
   6 0.403478      0.0.0.0           255.255.255.255   DHCP      358 DHCP Request  - Transaction ID 0x3484293b
   7 0.710463      192.168.1.251     255.255.255.255   DHCP      342 DHCP ACK      - Transaction ID 0x3484293b
```

DISCOVER

```
Source: 0.0.0.0
Destination: 255.255.255.255
▲ User Datagram Protocol, Src Port: 68, Dst Port: 67
  Source Port: 68
  Destination Port: 67
  Length: 308
  Checksum: 0x205c [unverified]
  [Checksum Status: Unverified]
  [Stream index: 0]
```

OFFER

```
Source: 192.168.1.251
Destination: 255.255.255.255
▲ User Datagram Protocol, Src Port: 67, Dst Port: 68
  Source Port: 67
  Destination Port: 68
  Length: 308
  Checksum: 0x6e5c [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2]
▲ Bootstrap Protocol (Offer)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x3484293b
    Seconds elapsed: 0
  ▷ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.1.109
    Next server IP address: 192.168.1.251
```

REQUEST

```
Source: 0.0.0.0
Destination: 255.255.255.255
▷ User Datagram Protocol, Src Port: 68, Dst Port: 67
▲ Bootstrap Protocol (Request)
    Message type: Boot Request (1)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x3484293b
    Seconds elapsed: 0
  ▷ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 0.0.0.0
    Next server IP address: 0.0.0.0
    Relay agent IP address: 0.0.0.0
    Client MAC address: LiteonTe_54:54:49 (d0:53:49:54:54:49)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▷ Option: (53) DHCP Message Type (Request)
  ▷ Option: (61) Client identifier
  ▲ Option: (50) Requested IP Address
      Length: 4
      Requested IP Address: 192.168.1.109
  ▲ Option: (54) DHCP Server Identifier
      Length: 4
      DHCP Server Identifier: 192.168.1.251
```

ACK

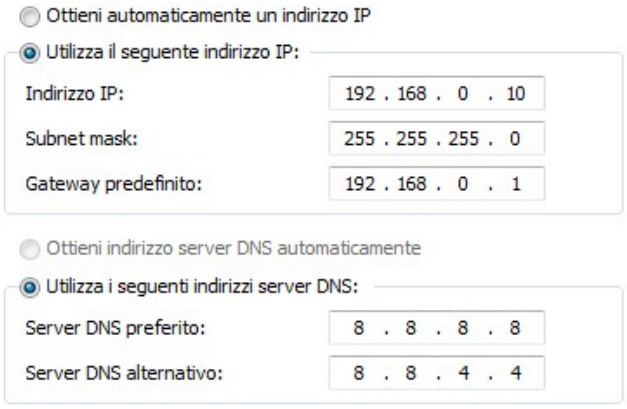
```
Source: 192.168.1.251
Destination: 255.255.255.255
▲ User Datagram Protocol, Src Port: 67, Dst Port: 68
  Source Port: 67
  Destination Port: 68
  Length: 308
  Checksum: 0x6b5c [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2]
▲ Bootstrap Protocol (ACK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x3484293b
    Seconds elapsed: 0
  ▷ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.1.109
    Next server IP address: 192.168.1.251
    Relay agent IP address: 0.0.0.0
    Client MAC address: LiteonTe_54:54:49 (d0:53:49:54:54:49)
    Client hardware address padding: 00000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
  ▲ Option: (53) DHCP Message Type (ACK)
      Length: 1
      DHCP: ACK (5)
  ▲ Option: (54) DHCP Server Identifier
      Length: 4
      DHCP Server Identifier: 192.168.1.251
```

Dopo aver analizzato i protocolli principali delle reti e di internet, andiamo ad analizzare altri termini molto utilizzati quando si parla di reti.

GATEWAY

Come dice la stessa parola, il Gateway è una porta, e più esattamente è una porta di accesso verso una rete esterna. Non si intende la porta software, ma proprio il dispositivo hardware che ha un suo indirizzo all'interno della rete, e che consentirà il collegamento dei vari dispositivi verso altre reti remote.

Nel classico esempio della rete domestica, il Gateway è lo stesso Router che ci consente di entrare in contatto con la rete Internet. Su ogni Client, se non è attivo il servizio DHCP (assegnazione automatica degli indirizzi) va indicato l'indirizzo del gateway che nel caso della rete domestica corrisponde all'indirizzo del Router.

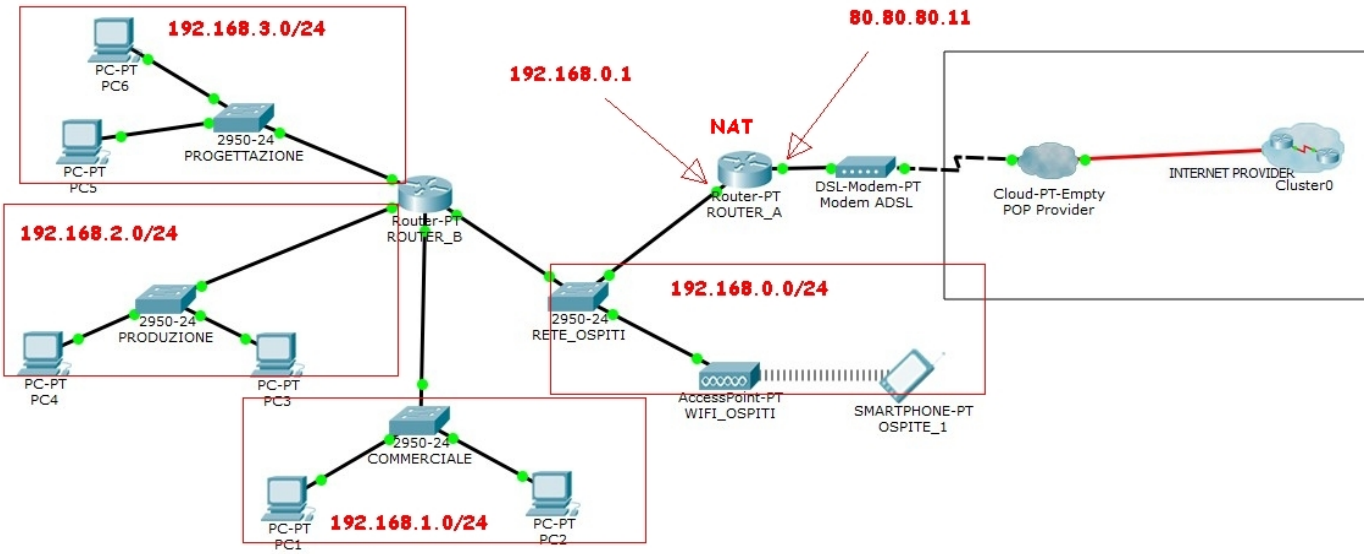


NAT (**N**etwork **A**ddress **T**raslator)

Il NAT è una tecnica utilizzata per modificare gli indirizzi IP contenuti negli header dei pacchetti, utilizza un Gateway, e nasconde dietro ad un unico indirizzo pubblico, gli indirizzi privati della rete.

Il suo compito può essere svolto da Router o dai Firewall.

In pratica da una LAN verso internet, **gli indirizzi dei Client della rete vengono mascherati in modo che esternamente appaia il solo indirizzo pubblico** assegnato dall'**ISP** (**I**nternet **S**ervice **P**rovider).

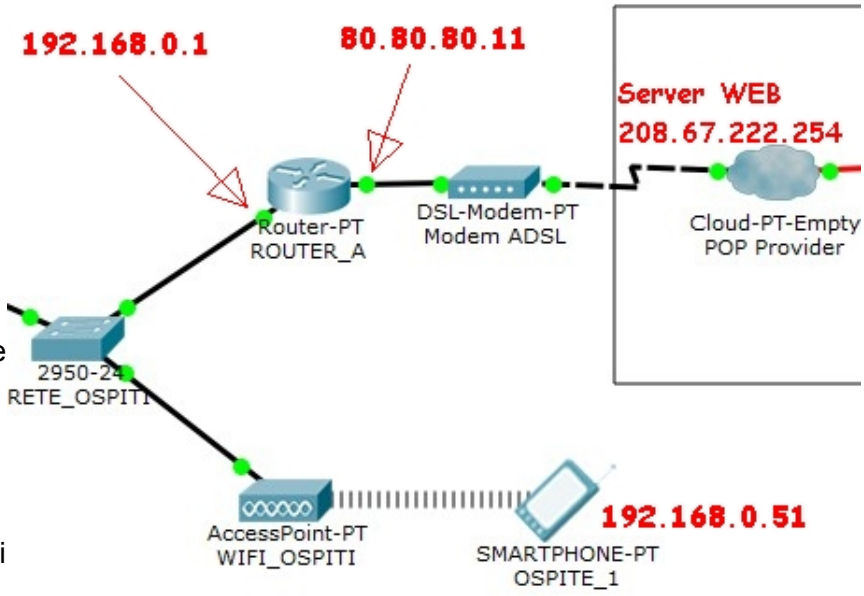


Nell'immagine sopra, ci sono 4 reti di cui 3 passanti per il ROUTER_B. L'accesso ad internet ed il NAT viene eseguito dal ROUTER_A, che a destra verso internet ha un indirizzo pubblico 80.80.80.11 mentre a sinistra verso la LAN ha un indirizzo 192.168.0.1.

Tutto i pacchetti in uscita dalla rete verranno mascherati presentando verso internet l'indirizzo pubblico.

In realtà quello che viene genericamente definito **NAT** è un **NAT** con **PAT** (**P**ort **A**ddress **T**raslator) o anche **NAPT** (**N**etwork **A**ddress and **P**ort **T**raslator) perché di fatto vengono anche mascherate le porte dei Client e non solo gli indirizzi. Prendiamo ad esempio il caso sopra ed analizziamo cosa accade quando il Client Smartphone si collega ad un server WEB con indirizzo 172.18.124.36.

Supponiamo che l'ISP abbia assegnato al ROUTER_A di figura un indirizzo IP pubblico pari a 80.80.80.11.
La sua scheda di rete verso la LAN ha indirizzo 192.168.0.1 ed è anche il Gateway della rete a cui appartiene lo Smartphone con indirizzo 192.168.0.51, rete con indirizzo 192.168.0.0/24.
Supponiamo inoltre che dallo Smartphone vogliamo accedere ad un Server WEB in internet con indirizzo 208.67.222.254 porta 80.
Nello Smartphone viene utilizzata come porta di uscita la 20100.



Vediamo ora cosa accade con la tecnica NAT. Il ROUTER_A riceve la richiesta di instaurare una connessione tra ***192.168.0.51 : 20100*** e ***208.67.222.254 : 80.***

Il Router sostituisce l'indirizzo **192.168.0.51** con il proprio indirizzo pubblico **80.80.80.11** e sostituisce la porta **20100** con una sua porta libera, supponiamo la **60100**. Questa operazione viene fatta nell'header IP del pacchetto dati che andrà al Server WEB.
Il router salverà in una tabella NAT interna alla sua memoria la sostituzione avvenuta e cioè:

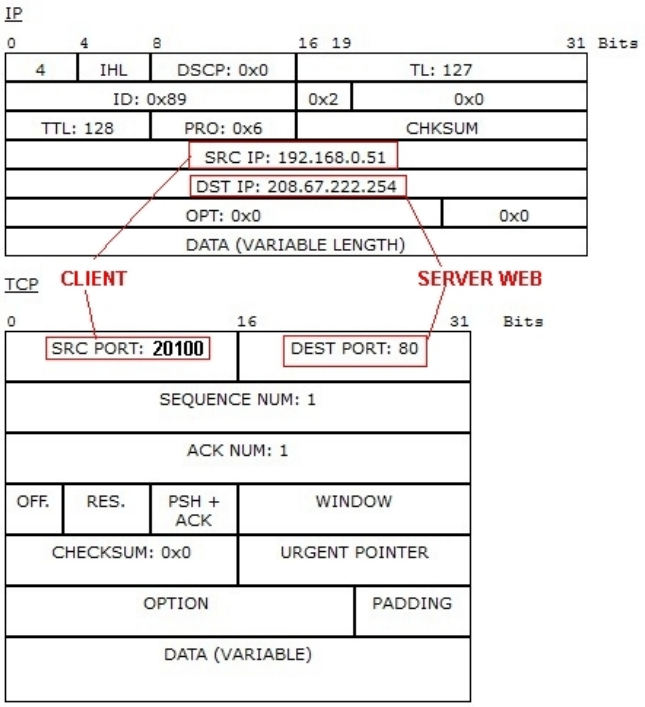
IP Privato del Clienti	Porta del Client	IP pubblico del Router	Porta pubblica del Router
192.168.0.51	20100	80.80.80.11	60100

Il Server WEB riceverà la richiesta, e dopo avera elaborata, invierà un messaggio, all'indirizzo pubblico ed alla porta pubblica del Router, **80.80.80.11 : 60100.**
Il Router ricevuto il pacchetto, controllerà la tabella di NAT e lo invierà al Client tramite il suo indirizzo privato e la sua porta privata, **192.168.0.51 : 20100.**

Nela pagina successiva i 4 pacchetti dati, relativi a quanto detto sopra.

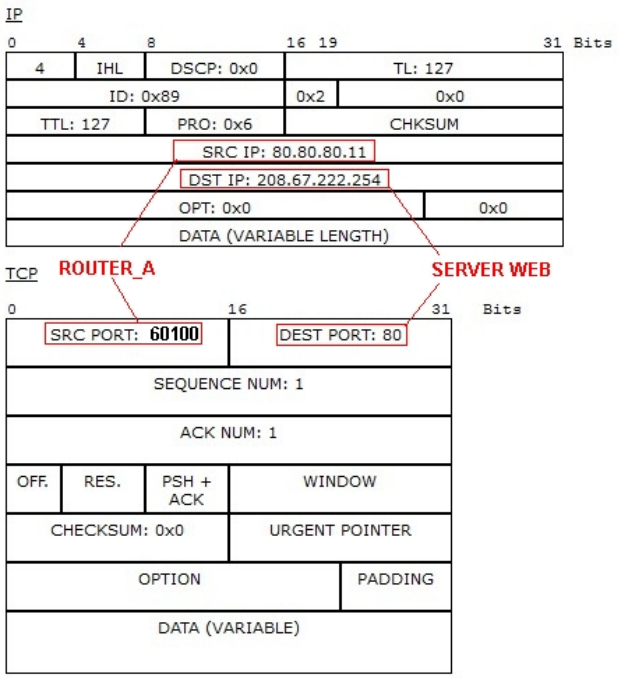
In uscita dal **CLIENT** verso il **ROUTER_A**

Contiene l'indirizzo sorgente e la porta del CLIENT e la l'indirizzo e la porta del Server WEB



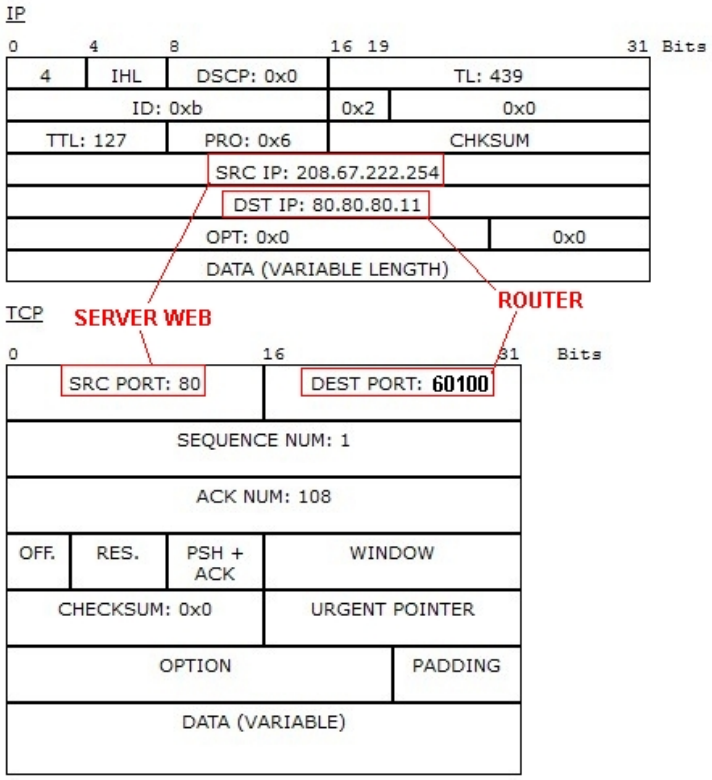
In uscita dal **ROUTER_A** verso il **SERVER WEB**

Contiene l'indirizzo pubblico e la porta del ROUTER e la l'indirizzo e la porta del Server WEB



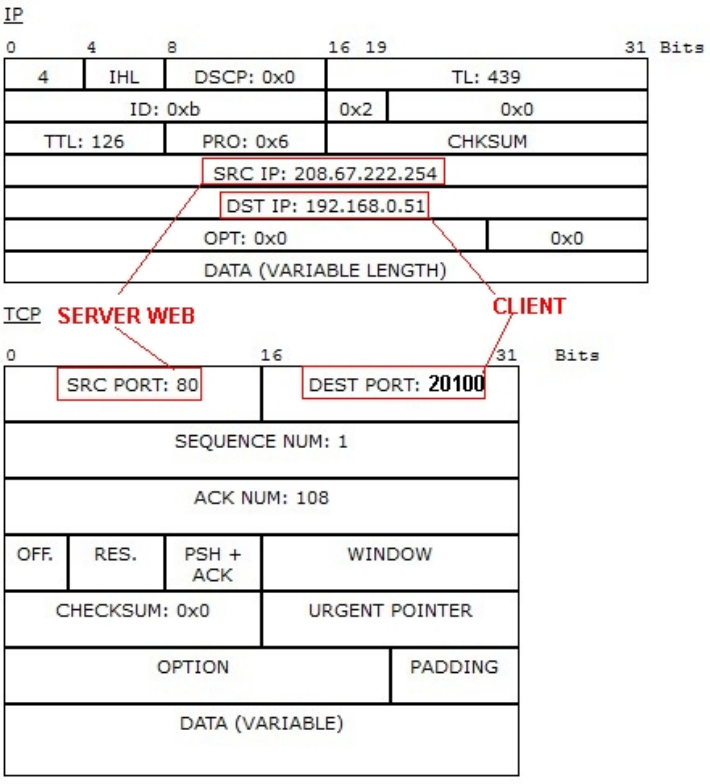
In arrivo al **ROUTER_A** dal **SERVER WEB**

Contiene l'indirizzo pubblico e la porta del ROUTER e la l'indirizzo e la porta del Server WEB



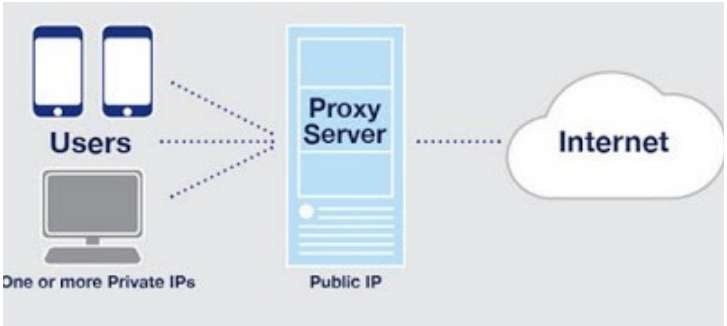
In arrivo al **CLIENT** dal **ROUTER**

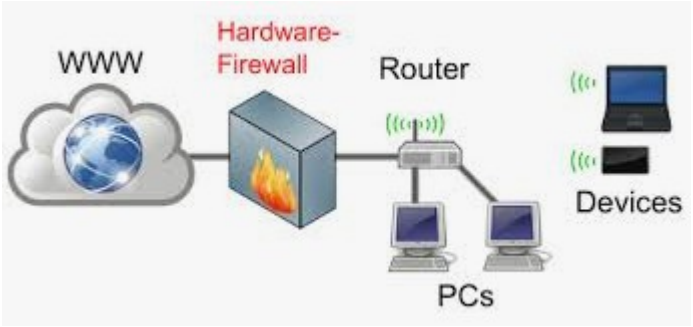
Contiene l'indirizzo privato e la porta del CLIENT e la l'indirizzo e la porta del Server WEB



SERVER PROXY

Il server Proxy, è una sorta di intermediario tra i Client di una rete LAN e altre reti come anche quella globale di internet. Questo servizio consente di “disaccoppiare” la rete con il mondo esterno, in modo da poter filtrare, regolare, e proteggere la navigazione.
Con un Server Proxy ad esempio si può scegliere quali risorse della rete possono essere raggiunte, e da quali utenti. Si possono creare delle copie di Cache delle risorse visitate in modo da renderle disponibili ai client senza effettuare altre richieste alla rete, si possono inoltre filtrare i contenuti verso i Client interni.
Nel Proxy si può realizzare anche un sistema FIREWALL (un componente software o anche hardware, per la sicurezza informatica) che consente l’accesso verso l’esterno della rete e dall’esterno di essa, secondo determinate regole. Un server Proxy che trasmette e riceve le richieste senza modificarle è il Gateway.





DISPOSITIVI DI RETE

Prima di terminare ricordiamo il funzionamento dei dispositivi di rete nominati fino ad ora:

SWITCH Lo Switch è un componente hardware che consente il collegamento di più Host ad una rete. Questo dispositivo lavora a livello 2, datalink. Esso contronlla il flusso di dati e lo indirizza solo al corretto destinatario secondo il proprio indirizzo.

ROUTER Il router è un dispositivo hardware che si occupa di instradare i dati suddivisi in pacchetti, e consente il collegamento tra sottoreti diverse. Lavora nel livello 3 network, ed utilizza degli algoritmi e protocolli di routing, per definire il percorso ottimale dei pacchetti di dati.

CONCLUSIONI

Lo scopo di questa seconda parte del tutorial sull’analisi delle Reti, era quello di conoscere con qualche dettaglio in più i principali protocolli, e di aprire qualche parentesi sui vari termini ed acronimi che spesso troviamo nel mondo delle reti.
Dopo aver descritto nella prima parte gli strumenti software necessari per analizzare e studiare il traffico dei dati su una rete, e dopo questa seconda parte, ora potremo affrontare con qualche conoscenza in più la parte relativa all’utilizzo degli strumenti software visti inizialmente (Wireshark e Kali Linux).
Nelle parti successive del tutorial, affronteremo questi aspetti anche con esempi pratici.