Generalità sulle reti di computer e sui livelli dello stack TCP/IP

A cura del prof. Giuseppe Mastrandrea
Sistemi e Reti

A.S. 2019/2020 I.I.S. A. Righi - Cerignola (FG)

Introduzione

Internet è un sistema di computer, reti e programmi interconnessi fra di loro piuttosto complicato. Sistemi periferici, programmi, router, protocolli, mezzi trasmissivi, sono tutti parte della grande infrastruttura che ci permette di utilizzare i nostri dispositivi preferiti (smartphone, tablet, computer, smart TV, persino elettrodomestici) in rete.

Livello Data Link

Il ruolo del Data Link è di consentire la comunicazione tra due dispositivi contigui. Guardando le cose dal basso, il livello fisico (Physical Layer) non ha intelligenza integrata, serve solo a trasformare bit in segnali e viceversa, senza sapere cosa significano in realtà questi bit. Stando subito sopra quest'ultimo, il livello Data Link controlla il livello fisico ed è quello che conosce il significato di questi bit. Cambiando il punto di vista e guardando invece le cose dall'alto, il livello Data Link è anche quello che consente al livello di rete (Network Layer) di comunicare senza problemi su una rete eterogenea (cioè su una rete che ha diversi mezzi trasmissivi e protocolli di accesso al mezzo diversi). In altre parole, il livello di rete non si preoccupa di come i dati sono immessi inseriti nel canale trasmissivo, né se il mezzo di trasmissione cambia da cablato a wireless; a livello di rete tutti gli host sono uguali. È il livello Data Link che gestisce tutto ciò autonomamente.

Ma chi si occupa di gestire il livello data-link? Esso deve essere pur implementato da qualche parte. Esso è presente in qualsiasi dispositivo di rete (altrimenti il dispositivo non sarebbe in grado di comunicare affatto), in particolare nella scheda di interfaccia di rete (**NIC - Network Interface Card**), o **scheda di rete**.

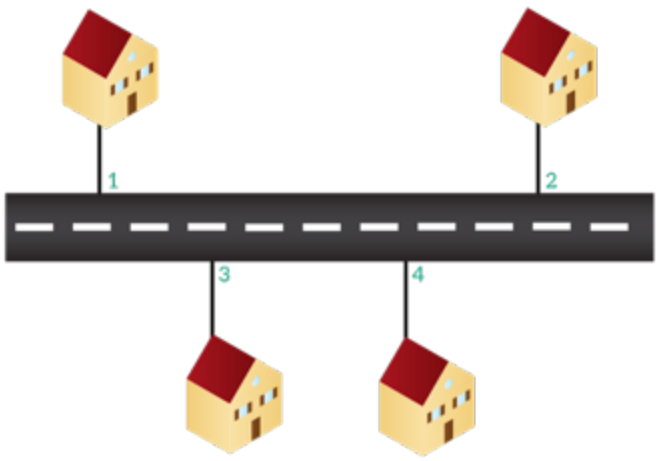
La scheda NIC è un modulo interno che si trova già incorporato in quasi tutti i computer moderni che consentono al dispositivo di connettersi alla Rete. Una NIC gestisce solo un tipo di connessione, quindi si avrà una NIC per la connessione ethernet, una NIC per le connessioni wireless, una per le connessioni Bluetooth, ecc ecc. I dispositivi di rete intermedi hanno diverse dozzine di NIC perché il loro ruolo è quello di connettere altri dispositivi insieme. Dentro il computer, la NIC comunica con l'unità di elaborazione centrale (CPU) del PC grazie al **bus PCI Express**, un canale di comunicazione all'interno di ogni computer che consente a componenti interni di comunicare tra loro. La CPU è il cervello del dispositivo, ed è il mezzo hardware tramite cui il sistema operativo gestisce i dati dell'applicazione: le applicazioni vengono eseguite all'interno della CPU. Ogni applicazione inizierà la comunicazione con un altro host con i propri dati (come la pagina Web in caso di un browser Web) e suddividerà tali dati in un datagramma a livello di Rete all'interno della CPU; la CPU quindi passerà questi dati del livello di rete alla NIC che gestirà il livello Data Link e quindi la trasformazione di quello in segnali (Physical Layer).

È qui che il concetto di **incapsulamento** diventa estremamente importante. Come sappiamo, l'informazione è originata dal livello applicativo (livello 7) del modello OSI, quindi viene trasmesso fino a raggiungere il livello fisico. Ogni volta che i dati attraversano un livello dall'alto verso il basso, ogni strato aggiunge alcune informazioni. Questi elementi extra sono quindi inviati sul supporto fisico e letti dall'altro dispositivo, allo stesso livello: informazioni aggiunte dal livello di presentazione del mittente, verrà letto dallo strato di presentazione sul

ricevitore, le informazioni aggiunte al livello di sessione verranno lette nel livello di sessione e così via. Sul dispositivo ricevente, questi pezzi extra delle informazioni vengono lette dal livello previsto e quindi rimosse prima di passare quei dati al livello superiore, in modo che alla fine il livello di applicazione ricevente leggerà esattamente ciò che l'applicazione sorgente ha inviato. L'incapsulamento avviene aggiungendo alcune informazioni all'inizio e, a volte, anche alla fine. Tuttavia, pezzi di informazioni non vengono mai aggiunte nel mezzo dei collegamenti. I dati dell'applicazione, oltre a tutte le informazioni aggiuntive aggiunte dai livelli, sono chiamate **Protocol Data Unit (PDU)**, la PDU del livello di presentazione sarà inferiore alla PDU del Data Link, poiché più livelli del Data Link hanno aggiunto le loro informazioni. Al fine di simulare che due applicazioni parlino "direttamente" (come se fossero sullo stesso computer), sfruttiamo pesantemente l'incapsulamento. Nella CPU si preparano i dati da inviare al livello di rete, mentre è la NIC che riceve i dati e aggiunge alla PDU di livello 3 le informazioni specifiche per il livello di collegamento dati. La NIC del dispositivo ricevente utilizzerà tali informazioni sul collegamento dati per garantire che i dati non siano interrotti durante il trasferimento e che i dati siano effettivamente destinati a tale dispositivo. In tal caso, rimuoverà le informazioni del livello Data Link e passerà il resto alla CPU, in modo che la CPU ricevente penserà che abbia appena ricevuto i dati dal livello di rete. La PDU del livello di rete si chiama Datagram, mentre la PDU del livello di collegamento dati è chiamata Frame, quindi la NIC dovrà inserire i pacchetti in frame durante la fase di invio ed estrarre pacchetti dai frame durante la fase di ricezione. La NIC del dispositivo ricevente utilizzerà tali informazioni sul collegamento dati per garantire che i dati non siano interrotti durante il trasferimento e che i dati siano effettivamente destinati a tale dispositivo. In tal caso, rimuoverà le informazioni del livello Data Link e passerà il resto alla CPU, in modo che la CPU ricevente penserà che abbia appena ricevuto i dati dal livello di rete.

La PDU del livello di rete si chiama Datagram, mentre la PDU del livello di collegamento dati è chiamata Frame, quindi la NIC dovrà inserire i pacchetti in frame durante la fase di invio ed estrarre pacchetti dai frame durante la fase di ricezione.

La funzione più importante del livello 2 del modello OSI è la consegna di informazioni al dispositivo di destinazione su un mezzo trasmissivo condiviso. Senza questa funzione la comunicazione non sarebbe possibile. Al fine di fornire dati al dispositivo di destinazione corretto, è necessario sapere dove si trova questo dispositivo, ovvero: dobbiamo conoscere il suo indirizzo. Questo procedimento è molto simile alla posta tradizionale, le informazioni del livello Data Link sono in una busta contenente i dati degli strati superiori. Come nella consegna

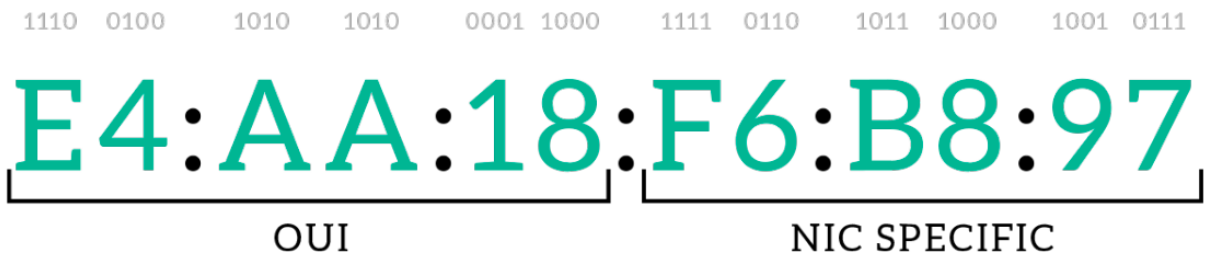


della posta ordinaria, leggiamo l'indirizzo sulla busta per sapere dove deve andare la lettera. Il networking è estremamente simile a quanto appena detto. Cosi come qualsiasi casa ha il suo indirizzo, ogni dispositivo ha il proprio indirizzo nel networking. Un dispositivo avrà un indirizzo di livello 2 e un indirizzo di livello 3, e quindi ogni applicazione su quel dispositivo avrà un indirizzo di livello 4. Per ora, ci concentreremo sull'indirizzo di livello 2.

Un indirizzo MAC è lungo sei byte ed è rappresentato in notazione esadecimale. La notazione esadecimale è solo una convenzione per rappresentare gli indirizzi MAC in un modo più leggibile; essi sono memorizzati in binario come qualsiasi altra informazione. Esploriamo l'indirizzo MAC nel dettaglio. Un indirizzo MAC è un indirizzo di livello 2 assegnato in maniera univoca a tutte le NIC, quindi un dispositivo con due NIC avrà due indirizzi MAC differenti. Può essere rappresentato in modi diversi: con una notazione trattino **E4-AA-18-F6-B8-97** o con una notazione tratteggiata dividendo per parole **e4aa.18f6.b897**: sono accettate sia maiuscole che minuscole, purché nello stesso indirizzo se ne usi solo una(tutto in minuscolo o tutto maiuscolo).

Di nuovo, la notazione esadecimale è solo una convenzione per rappresentare i MAC address in una maniera più human readable, ma in realtà, come qualsiasi altro dato, essi sono conservati in memoria in formato binario. Guardiamo ora più nel dettaglio la struttura degli indirizzi MAC.

Un MAC address è un indirizzo di livello 2 (o indirizzo fisico) che viene assegnato a tutte le NIC. Due NIC qualsiasi avranno di sicuro due indirizzi MAC diversi. Può essere rappresentato in vari modi: utilizzando un trattino (es. E4-AA-18-F6-B8-97) o con una notazione puntata (e4aa.18f6.b897): sono accettate maiuscole e minuscole, purché la notazione sia uniforme (tutte maiuscole o tutte minuscole).



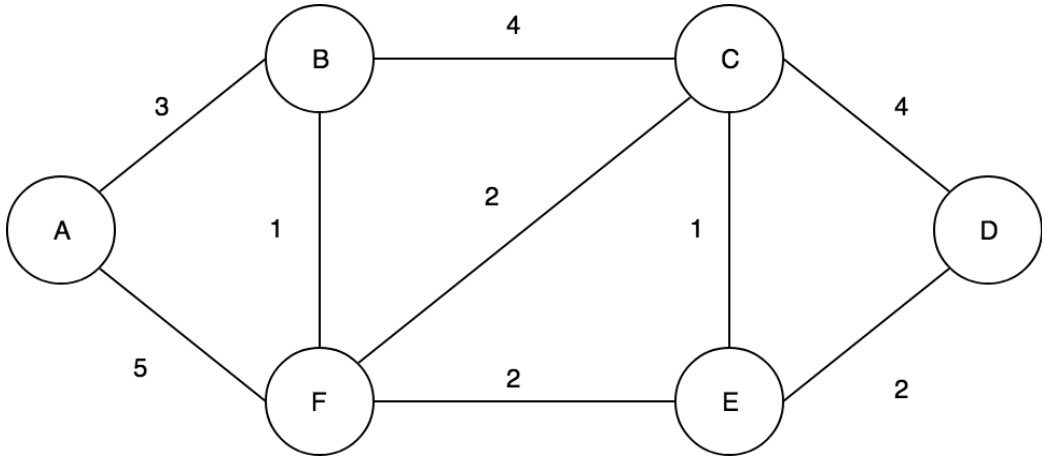
Un indirizzo MAC identifica una NIC in maniera univoca: non esiste nessun indirizzo MAC uguale ad un altro nel mondo. I primi tre byte (tre coppie di cifre) sono chiamati Identificatori univoci (OUI) e loro sono assegnati alla casa produttrice di quella NIC. Gli ultimi 3 byte invece sono dedicati al modello della NIC. Grandi produttori come Cisco hanno acquistato diversi OUI perché producono così tante NIC che avere solo tre byte per definire il loro identificatore univoco non era abbastanza. Poiché l'OUI è assegnato alle aziende, possiamo capire dall'indirizzo MAC della NIC l'azienda che l'ha prodotta; esistono diversi strumenti per questo scopo e sono disponibili online gratuitamente. Un esempio di questi programmi è Wireshark. Su Windows, per conoscere l'indirizzo fisico bisogna premere Windows + R, digitare cmd e premere invio. Quindi, digitare ipconfig / all e cercare la voce di elenco "Indirizzo fisico".

L'indirizzo MAC viene già installato nella NIC. Può essere modificato tramite software, ma il MAC originale della NIC non può essere rimosso completamente perché è codificato nell'hardware, questo è chiamato **Burnt-in Address** (BIA – indirizzo marchiato a fuoco) perché non può essere rimosso. L'uso di un indirizzo MAC diverso dalla BIA non è una pratica comune e generalmente comporta attività dannose come spoofing (provare leggere pacchetti che non erano destinati a noi). Gli indirizzi MAC sono quelli usati per fornire le informazioni al dispositivo di destinazione corretto. Il frame di livello Data Link contenente le informazioni di livello 3 avrà su di esso l'indirizzo MAC di destinazione e l'indirizzo MAC di origine. La consegna non è basata solo sulla destinazione dell'Indirizzo MAC, ma anche all'indirizzo MAC di origine dal dispositivo di destinazione per sapere dove inviare la risposta (questo non è vero per tutti dispositivi).

Livello di rete

Una volta compreso come funziona la comunicazione tra host adiacenti, cioè host che condividono uno stesso mezzo fisico, è arrivato il turno di salire di livello e occuparci del livello di rete. Il livello di rete offre il macro-servizio di comunicazione **end-to-end**, ovvero: offre il servizio di comunicazione fra i nodi terminali di una comunicazione. Per fare questo il livello di rete sfrutta il livello sottostante (data-link), basato sulle tipologie di collegamento, per offrire un concetto di rete di computer molto più generico. A livello di rete infatti **non ci interessa più come siamo connessi ad una rete**. Un host che faccia parte di una rete a livello 3 (o rete IP, dal nome del principale protocollo di livello di rete: **Internet Protocol**) è un generico membro della rete, e prende il nome di **host**.

Prima di parlare più nel dettaglio del livello di rete e sue funzionalità è fondamentale capire *se* esiste un modo rappresentare una rete in maniera generica come un insieme di host connessi fra loro. Effettivamente questo modo generico di rappresentare una rete esiste, ed è preso direttamente dalla matematica. È infatti possibile rappresentare una rete con una struttura matematica che prende il nome di **grafo**. Un grafo è la rappresentazione matematica equivalente di una rete nel networking. Dal punto di vista formale, un grafo `G(V, E)` è una coppia di due insiemi V e E che indicano rispettivamente un insieme di vertici o nodi (Vertices) e lati o archi (Edges). Meno formalmente, è possibile immaginarsi un grafo come un insieme di nodi e di archi che connettono coppie di nodi. Gli archi di un grafo possono avere un numero ad essi associato che indica il peso (in questo caso si parla di *grafo pesato*), e che si indica con `c(x, y)`, dove x e y sono due nodi arbitrariamente scelti. Il costo per andare da un nodo ad un altro è semplicemente la somma dei pesi degli archi attraversati lungo il percorso fra i 2 nodi.



Gli insiemi che formano il grafo in figura sono:

```
V = {A, B, C, D, E, F}
E = { (A,B), (A,F), (B,C), (B,F), (C,D), (C,E), (C,F), (D,E),
(E,F) }
```

Ipotizzando il seguente percorso fra il nodo F e il nodo D:

```
F, C, E, D
```

il costo di questo percorso è semplicemente la somma dei pesi lungo gli archi attraversati:

```
c(F, C) + c(C, E) + c(E, D) = 2 + 1 + 2 = 5
```

Ritornando alle reti, ogni nodo del grafo rappresenta un host della rete e ogni arco un collegamento verso un altro host. Come avviene dunque la comunicazione fra due host di una rete IP? I messaggi che escono da un host non arrivano direttamente da un host all'altro, ma passano -appunto- attraverso gli archi della rete fino a raggiungere l'host destinazione. I costi presenti sugli archi sono dei numeri che potrebbero rappresentare caratteristiche del collegamento fra i nodi come bitrate, affidabilità, etc. Ad esempio: tramite il nostro smartphone vogliamo visitare un sito Internet. Il nostro smartphone rappresenta un nodo della rete Internet, che è una rete IP. Questo significa che il nostro smartphone può essere visto come un nodo del grafo che rappresenta la rete Internet: diciamo che rappresentiamo la rete Internet con il grafo in figura (nella realtà la rete Internet è formata da milioni di nodi interconnessi fra loro) e che il nostro smartphone sia il nodo A. Anche il sito che vogliamo visitare risiede su un computer che fa parte della rete Internet (altrimenti non potremmo visitarlo), quindi ovviamente anch'esso può essere rappresentato come un nodo di un grafo: diciamo che sia il nodo D. Quali e quanti percorsi esistono fra il nodo A e il nodo D in figura?

1. `A, B, C, D`
2. `A, F, E, D`
3. `A, F, C, D`
4. `A, B, F, C, D`
5. `A, B, F, E, D`
6. `A, B, C, E, D`
7. `A, F, B, C, D`
8. `A, F, C, E, D`
9. `A, F, E, C, D`
10. `A, B, C, F, E, D`
11. `A, B, F, C, E, D`

Notiamo che un percorso è valido solo se un nodo viene visitato al più una volta sola lungo la comunicazione (non possono esserci *anelli* o *circoli*). Abbiamo detto all'inizio della sezione che il livello di rete si occupa di gestire la comunicazione end to end. In questo caso quindi si occupa di gestire la comunicazione fra il nodo A (il nostro smartphone) e il nodo D (il sito che vogliamo visitare). Per garantire questo servizio, il livello di rete deve gestire quindi una cosa fondamentale: la scelta del percorso fra un nodo sorgente e un nodo destinazione.

Ma la scelta di un percorso casuale non sarebbe efficiente, poichè porterebbe (ad esempio) a ritardi nella comunicazione, perdite di messaggi, e in generale problemi che renderebbero la comunicazione non ottimale. Il livello di rete non si accontenta di scegliere un percorso qualsiasi: il suo compito è quello di scegliere il percorso più adatto fra quelli disponibili. Il percorso migliore è quello a **costo minore**. Quindi nel nostro caso, il livello di rete calcola per tutti i percorsi possibili il rispettivo costo, e sceglie il percorso a costo minore:

1. `A, B, C, D` => 11
2. `A, F, E, D` => 9
3. `A, F, C, D` => 11
4. `A, B, F, C, D` => 10
5. `A, B, F, E, D` => 8
6. `A, B, C, E, D` => 10
7. `A, F, B, C, D` => 14
8. `A, F, C, E, D` => 10
9. `A, F, E, C, D` => 12
10. `A, B, C, F, E, D` => 14
11. `A, B, F, C, E, D` => 9

Il livello di rete sceglierà il percorso a costo minore (cioè il n. 5) per consegnare un messaggio dal nodo A al nodo D. Per trovare il percorso a costo minore il livello di rete si serve di algoritmi che siano in grado di trovare il percorso a costo minimo e di protocolli di routing.

Questa importantissima funzionalità del livello di rete prende il nome di **routing** o **instradamento.** Esso è un servizio **esterno** o **globale**, ovvero fornito dal livello di rete nella sua globalità. L'altra funzionalità tramite la quale il livello di rete offre il servizio di comunicazione end to end è il **forwarding** o **inoltro**: strettamente connesso al routing, esso consiste nella capacità degli host, in base alle informazioni ricavate dai protocolli di routing, di inoltrare un determinato messaggio sull'arco appropriato. Ad esempio, nel percorso 5, il nodo F utilizza il forwarding per fare viaggiare il messaggio sull'arco appropriato: in questo caso il nodo F fa viaggiare il messaggio sull'arco `(F, E)`, perchè sa che il percorso migliore (quello selezionato dal livello di rete grazie al routing) è il percorso `A, B, F, E, D`. Il forwarding è un servizio **interno** ai nodi che compongono una rete IP.

Ma al di là delle strutture matematiche, quali sono le regole di Internet? Ovviamente il traffico Internet è regolato da regole molto rigide. A livello di rete, il protocollo più importante e senza il quale Internet stessa non esisterebbe è il **protocollo IP** (Internet Protocol, RFC 791). IP è un insieme molto complesso di regole e formati di scambio di dati. In questa fase ci limiteremo ad esaminare le sue caratteristiche peculiari senza scendere troppo nel dettaglio.

Abbiamo già introdotto il concetto di "indirizzo" analizzando il livello data-link. Il concetto di indirizzo a livello di rete (d'ora in poi **indirizzo di rete** o **indirizzo logico** o **indirizzo IP**) presenta alcuni aspetti comuni con quello di indirizzo a livello data-link. Un indirizzo di livello data-link è noto solo ai dispositivi fra loro contigui, mentre un indirizzo di rete, potenzialmente, è universalmente conosciuto. L'indirizzo IP è associato al singolo dispositivo connesso ad una rete IP. Per cercare un parallelo, prendiamo ad esempio la posta tradizionale. Per far arrivare

una lettera a destinazione c'è bisogno di scrivere sulla busta: nazione, città, CAP, via e numero civico. Tutte queste singole informazioni formano l'indirizzo. Esso è strettamente dipendente dal luogo fisico in cui si trova un destinatario: due nazioni diverse hanno due indirizzi diversi. Se ci trasferiamo da una nazione all'altra avremo indirizzi diversi: allo stesso modo, se un dispositivo si muove da una porzione di Internet ad un'altra, esso cambierà indirizzo IP. Che cosa intendiamo per "porzione"? Come vengono assegnati gli indirizzi IP? Per rispondere a queste domande dobbiamo capire come è strutturata la rete Internet. Essa è fatta da una serie di sistemi autonomi (cioè da gruppi di computer, cioè da grafi) gestiti da una serie di aziende chiamate **ISP** (**Internet Service Provider**). I provider del servizio Internet in sostanza gestiscono quindi delle porzioni di Internet rappresentate da grafi. Cosa intendiamo per "gestire"? Fra le altre cose, essi (i provider) forniscono i loro clienti di indirizzi IP, quindi sostanzialmente fanno in modo che un loro cliente faccia parte del grafo, cioè della rete Internet, e quindi possano raggiungere ed essere raggiunti da un qualsiasi altro nodo presente sulla rete Internet. Un aspetto molto importante legato all'assegnazione degli indirizzi IP è il seguente: **in una rete IP, non possono esserci due host che abbiano lo stesso indirizzo IP**. Sarebbe come dire che 2 persone completamente diverse avessero lo stesso indirizzo: il servizio postale non saprebbe a chi consegnare la posta destinata a quell'indirizzo! Detto questo, cerchiamo di capire come è fatto un indirizzo IP.

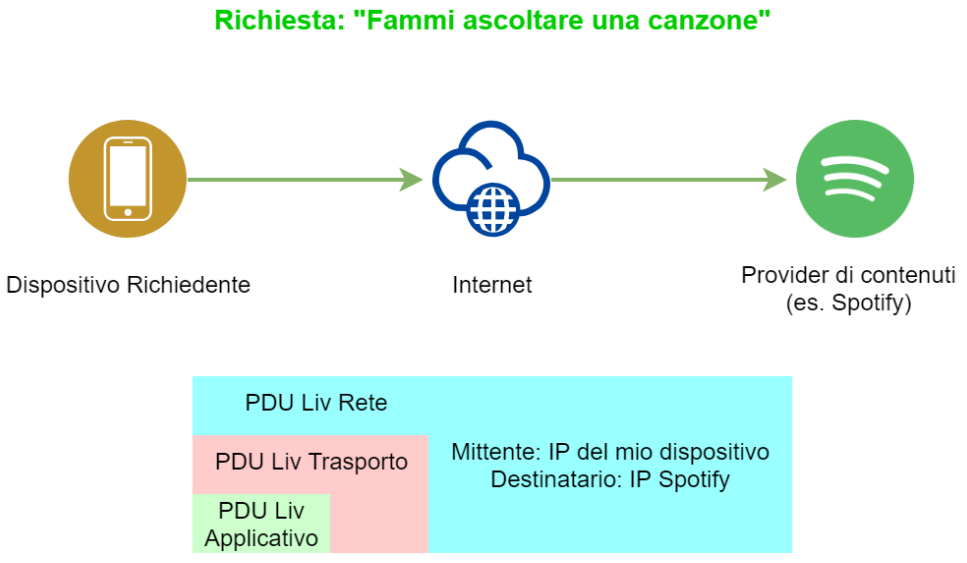
Un indirizzo IP è lungo 4 byte (32 bit). Per rappresentarlo in una forma più human readable, ogni byte viene convertito in un numero decimale e viene separato dagli altri byte da un punto:

Un indirizzo IP rappresenta semplicemente un modo per raggiungere un host all'interno di una rete IP. Essendo Internet una rete IP (che connette centinaia di milioni di nodi sparsi in tutto il mondo fra loro), possiamo dire che un indirizzo IP rappresenta l'indirizzo di un nodo della rete Internet**,** ovvero **rappresenta l'indirizzo di un qualsiasi host connesso ad Internet**. Pensiamo ad un sito internet (Facebook, Instagram, i siti dei quotidiani), ad un servizio di streaming (Youtube, Spotify, Netflix), ad un servizio di Instant Messaging (Skype, Whatsapp, Telegram). Tutte queste piattaforme "fanno parte" di Internet: cioè esse sono raggiungibili via Internet. Ma internet è una rete IP. Quindi, esse si trovano su un computer che ha un indirizzo IP nella rete Internet, vale a dire su un **host**. Quando un utente generico si

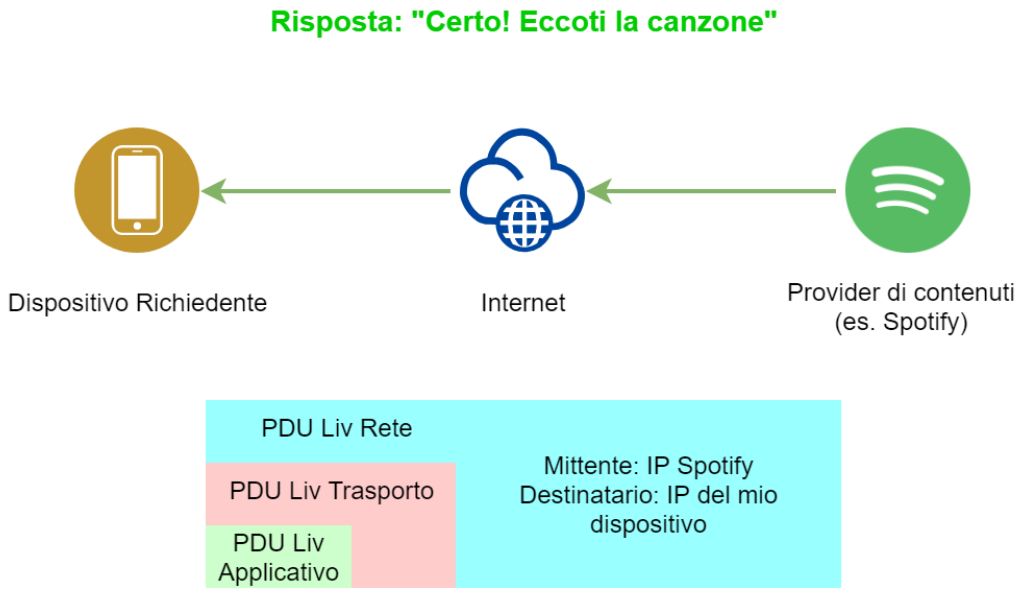
connette ad un sito Internet, o usa uno di questi servizi di streaming o di IM, in realtà egli si sta connettendo ad un computer con un determinato indirizzo IP.

Come avviene dunque la comunicazione via Internet? Dalle nostre nozioni sull'incapsulamento, sappiamo che a livello di rete viene costruita una PDU contenente un certo tipo di informazioni. Sappiamo inoltre che il livello di rete si occupa della comunicazione end to end (cioè fra gli host terminali). Alla luce di queste nozioni, è abbastanza immediato immaginare il contenuto principale di una PDU di livello di rete: essa contiene principalmente gli **indirizzi IP sorgente e destinazione** per una determinata comunicazione.

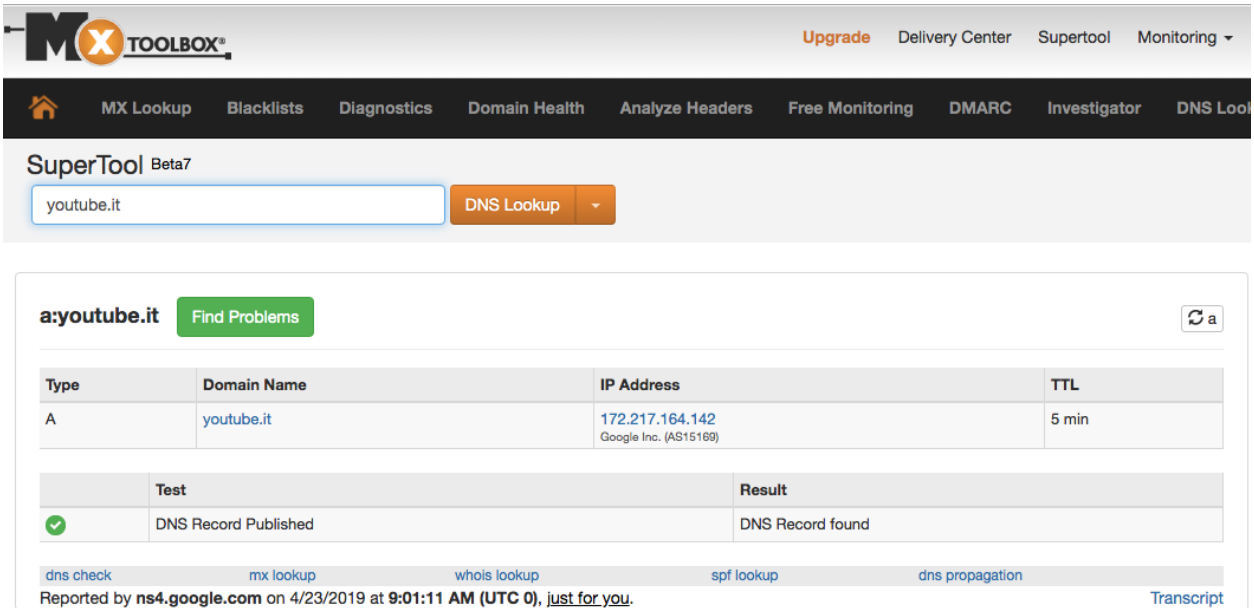
Pensiamo ad una busta contenente un messaggio di posta tradizionale: essa contiene l'indirizzo del destinatario della nostra lettera sul retro, mentre davanti contiene (o perlomeno è buona norma che contenga) l'indirizzo del mittente della lettera, in maniera che il destinatario della nostra lettera non solo sappia chi gli ha scritto, ma possa anche spedire ad un preciso indirizzo un'eventuale risposta alla lettera. Trasportando questo esempio alla rete Internet, quando proviamo a connetterci ad un sito Internet, o proviamo ad ascoltare una canzone o vedere un film utilizzando un servizio di streaming (es. Spotify, Netflix, Youtube), fondamentalmente stiamo applicando lo stesso principio della posta tradizionale: dal nostro dispositivo esce un messaggio indirizzato verso il nodo della rete Internet (cioè l'host) contenente le informazioni che cerchiamo; siccome fa parte della rete Internet, esso ha un indirizzo IP. Ma anche il nostro dispositivo fa parte della rete Internet, e anch'esso quindi è dotato di un indirizzo IP. Succede quindi che dal nostro dispositivo parte un messaggio che a livello di rete contiene *l'indirizzo IP del nostro dispositivo* come indirizzo sorgente (mittente) e *l'indirizzo IP della piattaforma da cui vogliamo informazioni* (destinatario) come indirizzo destinazione. La nostra richiesta viaggia attraverso il grafo che rappresenta la rete Internet e viene recapitata a destinazione (il nodo della rete IP corrispondente al sito Internet o alla piattaforma da cui vogliamo dei contenuti).



Come è possibile che quei dati poi ritornino indietro? Facile: nella richiesta partita dal nostro dispositivo viene specificato anche l'indirizzo IP sorgente, quindi il provider di contenuti non farà altro che spedire indietro sulla rete Internet un altro messaggio contenente i dati che erano stati richiesti, stavolta con gli indirizzi scambiati:



Sorge quindi spontanea una domanda: come è possibile conoscere l'indirizzo IP di un fornitore di servizi (nel nostro esempio Spotify)? Per un essere umano è più facile ricordarsi un nome di dominio (es. facebook.com, instagram.com, youtube.it, ilpost.it, etc) che una serie di numeri. Per questo è stato inventato il protocollo **DNS** (Domain Name System), che si occupa di effettuare la conversione da un nome di dominio ad un indirizzo IP. In questo modo un utente piuttosto che connettersi ad un sito utilizzando il suo indirizzo IP, può digitare il nome associato a quell'indirizzo e lasciare che sia il protocollo DNS a effettuare la conversione. Chiunque sia curioso di scoprire quale sia l'indirizzo IP corrispondente ad un determinato sito Internet può usare uno degli innumerevoli strumenti presenti online, per esempio il SuperTool di MxToolbox.



Al contrario, come è possibile conoscere l'indirizzo IP del *proprio* dispositivo? Anche per risolvere questo problema esiste una moltitudine di siti Internet specializzati. Uno fra i più conosciuti è https://www.whatsmyip.org/.

A decidere come effettuare la "distribuzione" degli indirizzi IP è l'ISP che ci fornisce la connessione (Tim, Vodafone, Wind, Tre, etc.). Se un indirizzo IP viene assegnato in maniera permanente ad un utente si parla di **indirizzamento fisso** (Fastweb è un ISP che fornisce indirizzamento fisso), mentre se gli indirizzi IP variano ogni volta che ci connettiamo/disconnettiamo dalla rete dati si parla di **indirizzamento dinamico**.

Utility: ping e traceroute

Ping: round-trip time

Un'alternativa per scoprire un indirizzo IP di un host, e anche per avere altre informazioni, è un software chiamato "**ping**", e presente in tutti i sistemi operativi. Ping è in realtà un programma che serve a capire quanto tempo ci impiega un pacchetto di dati di una certa dimensione a raggiungere un certo host su Internet; ci fornisce quindi informazioni relative al tempo di connessione verso un certo host. Minore sarà il tempo, ovviamente, migliore sarà la qualità della connessione verso quell'host. Per aprirlo da Windows basta:

- Su windows, premere la combinazione di tasti "Win" + "R" (apre la finestra di dialogo "esegui") oppure su un sistema operativo Linux o Mac OS aprire "terminale"
- Solo Windows: digitare "cmd" e premere invio (apre il prompt dei comandi di windows, un'interfaccia a caratteri per usare windows)
- Nella finestra di dialogo che si aprirà digitare "ping *nomedominio*"

L'output di questa serie di passaggi sarà qualcosa di simile a questo (screenshot di un comando ping eseguito da un sistema operativo Mac OS):

```
Mac-mini-di-Mac:~ giumast$ ping youtube.it
PING youtube.it (172.217.23.110): 56 data bytes
64 bytes from 172.217.23.110: icmp_seq=0 ttl=54 time=94.798 ms
64 bytes from 172.217.23.110: icmp_seq=1 ttl=54 time=88.682 ms
64 bytes from 172.217.23.110: icmp_seq=2 ttl=54 time=70.623 ms
64 bytes from 172.217.23.110: icmp_seq=3 ttl=54 time=38.267 ms
^C
--- youtube.it ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 38.267/73.093/94.798/21.983 ms
```

Analizziamo l'output che ci è restituito dal comando "ping":

- Alla prima riga vediamo la corrispondenza nome di dominio/indirizzo IP. Il programma ci informa che sta effettuando un ping verso youtube.it, che ha indirizzo IP pari a 172.217.23.110, utilizzando dei pacchetti di dati grandi 56 byte.
- Le 4 righe successive ci dicono invece che abbiamo provato a mandare 4 pacchetti di dati e che l'host 172.217.23.110 ci ha risposto con pacchetti di dati grandi 64 byte, mandati nell'ordine indicato dal numero "icmp_seq" (0,1,2,3) in un tempo rispettivamente pari a: 94.798 millisecondi, 88.682 millisecondi, 70.623 millisecondi, 38.267millisecondi.
- Infine nelle ultime due righe troviamo le statistiche:
    - 4 pacchetti di dati inviati
    - 4 pacchetti di risposta ricevuti
    - Percentuale di pacchetti persi: 0.0% (tutti i pacchetti inviati sono ritornati indietro)
- Infine abbiamo informazioni sul tempo di *round-trip*, cioè il tempo che è trascorso da quando abbiamo inviato il messaggio di test fino al momento in cui è tornato indietro il messaggio:
    - Round Trip minimo
    - Round trip medio
    - Round trip massimo
    - Deviazione standard

Traceroute

Un'altra utility molto importante per comprendere a fondo cosa succede quando ci connettiamo ad un certo host su internet è **traceroute** (su Windows abbreviato in **tracert**). Nella parte introduttiva sul livello di rete abbiamo infatti detto che il livello di rete gestisce la comunicazione end to end, cioè fra nodi terminali. Quello che accade però è che i pacchetto di dati (richieste e risposte) viaggiano nel grafo che è Internet, ovvero vengono inoltrati di nodo in nodo fino a raggiungere il nodo di destinazione. L'utility traceroute ci indica esattamente quale percorso fa all'interno del grafo che rappresenta Internet un certo pacchetto di dati prima di raggiungere una destinazione che noi specifichiamo. Per usare traceroute basta effettuare i seguenti passaggi:

- Su windows, premere la combinazione di tasti “Win” + “R” (apre la finestra di dialogo “esegui”) oppure su un sistema operativo Linux o Mac OS aprire “terminale”;
- Solo Windows: digitare “cmd” e premere invio (apre il prompt dei comandi di windows, un’interfaccia a caratteri per usare windows);
- Nella finestra di dialogo che si aprirà digitare “tracert *nomedominio*” (su windows) o “traceroute nomedominio” su Mac OS o Linux

L’output di questa serie di passaggi sarà qualcosa di simile a questo (screenshot di un comando traceroute eseguito da un sistema operativo Mac OS):

```
Mac-mini-di-Mac:~ giumast$ traceroute youtube.it
traceroute to youtube.it (216.58.205.78), 64 hops max, 52 byte packets
 1  modemtim (192.168.1.1)  36.246 ms  2.133 ms  11.455 ms
 2  * * *
 3  172.17.144.214 (172.17.144.214)  33.690 ms
    172.17.144.212 (172.17.144.212)  28.541 ms
    172.17.144.250 (172.17.144.250)  58.707 ms
 4  172.17.145.108 (172.17.145.108)  62.256 ms
    172.17.145.126 (172.17.145.126)  93.316 ms
    172.17.145.116 (172.17.145.116)  7.209 ms
 5  172.19.245.81 (172.19.245.81)  76.652 ms
    172.19.245.77 (172.19.245.77)  54.703 ms  106.236 ms
 6  etrunk12.milano1.mil.seabone.net (93.186.128.205)  60.008 ms
    etrunk38.milano50.mil.seabone.net (195.22.196.92)  92.532 ms
    etrunk12.milano1.mil.seabone.net (93.186.128.205)  39.871 ms
 7  74.125.51.148 (74.125.51.148)  87.122 ms
    72.14.195.206 (72.14.195.206)  98.883 ms
    72.14.204.72 (72.14.204.72)  106.280 ms
 8  108.170.245.65 (108.170.245.65)  62.555 ms
    108.170.245.81 (108.170.245.81)  63.045 ms
    108.170.245.65 (108.170.245.65)  55.317 ms
 9  216.239.42.11 (216.239.42.11)  88.721 ms *  116.771 ms
10  mil04s25-in-f14.1e100.net (216.58.205.78)  95.008 ms  91.958 ms  93.159 ms
```

Semplificando di molto, potremmo dire che il comando traceroute manda 3 pacchetti di dati verso l’host destinazione (in questo caso 216.58.205.78) e “registra” ogni passaggio, vale a dire l’indirizzo IP di ogni nodo intermedio del grafo prima di raggiungere il nodo host di destinazione finale (il server di youtube.it), specificando anche il round trip time (analogamente a quanto fa ping) di ogni nodo intermedio. Notiamo che ognuno dei 3 pacchetti può fare percorsi diversi (motivo per cui per i passaggi 3, 4, 7, 8) abbiamo 3 indirizzi IP diversi, mentre per il passaggio 5 ne abbiamo 2. Notiamo anche che nel passaggio 2 abbiamo degli asterischi, che indicano una perdita di dati. In questo caso traceroute cercherà un altro percorso per arrivare a destinazione.

Geolocalizzazione degli indirizzi IP

Finora abbiamo parlato solo di indirizzi IP e di host. Ma se è vero che questi host sono dei nodi della rete, è anche vero che essi si troveranno da qualche parte nel mondo. I servizi di geolocalizzazione servono proprio a risalire ad una locazione fisica (intesa come coordinate

geografiche di latitudine e longitudine) di un nodo della rete Internet con un certo indirizzo IP. Ce ne sono di gratuiti (meno accurati) e a pagamento, di solito molto accurati. In realtà gli unici a poter fornire informazioni molto dettagliate sulle coordinate geografiche di un certo indirizzo IP sono i fornitori di servizi Internet (ISP). Ad ogni modo esistono molti servizi in grado di fornire un'indicazione di massima delle coordinate GPS di un certo indirizzo IP. La conversione da indirizzo IP a coordinate geografiche si chiama **IP lookup**. Una piattaforma molto popolare che offre il servizio di IP lookup è "Whatismyipaddress", raggiungibile all'URL https://whatismyipaddress.com/ip-lookup.

Lo step successivo è quello di visualizzare su una mappa tutti i passaggi dei pacchetti che transitano in una sessione traceroute. È esattamente quello che fa la piattaforma "ip2location", disponibile a questo URL: https://www.ip2location.com/free/traceroute

Ecco un esempio di output:

Traceroute to 216.58.205.78 from Italy

Hop	IP Address	Hostname	Location	Time
1	149.154.157.1	gw.it.milano.edis.at	Italy, Lombardia, Milan	0.907 ms
2	178.249.188.2	178.249.188.2	Italy, Lombardia, Milan	0.476 ms
3	10.130.1.41	10.130.1.41	LOCAL PRIVATE LAN	2136.737 ms
4	217.171.32.170	217.171.32.170	Italy, Lombardia, Milan	0.578 ms
5	217.29.66.96	google.mix-it.net	Italy, Lombardia, Milan	0.688 ms
6	108.170.245.81	108.170.245.81	United States, California, Mountain View	2.062 ms
7	216.239.42.11	216.239.42.11	Netherlands, Noord-Holland, Amsterdam	0.595 ms
8	216.58.205.78	mil04s25-in-f78.1e100.net	Netherlands, Noord-Holland, Amsterdam	0.535 ms

Reti pubbliche e reti private

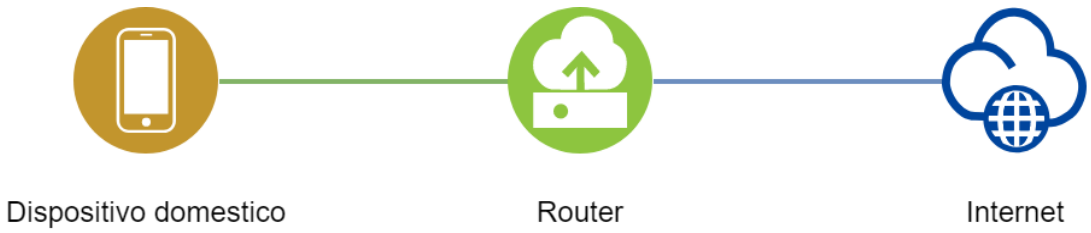
Nella figura precedente, oltre ai normali passaggi attraverso nodi del grafo che rappresenta Internet, c'è una riga che ancora non siamo in grado di interpretare. La riga 3

infatti è contrassegnata dalla Location “Local Private LAN”. Che significa? Perchè il servizio di ip lookup non è stato in grado di dirci da che nazione proviene quell’indirizzo IP?

Per rispondere a queste domande dobbiamo fare un piccolo passo indietro e tornare alla struttura generale degli indirizzi IP. Abbiamo detto che un indirizzo IP è formato da 4 byte, vale a dire 32 bit. Il che ci porta a dire che in totale ci sono 2^{32} indirizzi IP. Ma dalle nostre nozioni sulla rete Internet, dovrebbe ormai apparire chiaro che in Internet non possono esserci contemporaneamente 2 host che hanno lo stesso indirizzo IP (sarebbe come dire che persone completamente diverse fra loro hanno lo stesso indirizzo di posta, si creerebbe un’ambiguità per cui il postino non saprebbe a chi consegnare una lettera o una cartolina). Unendo queste 2 considerazioni, possiamo dedurre che in Internet non possono esserci più di 2^{32} persone connesse contemporaneamente; se così fosse tutti gli indirizzi IP sarebbero occupati e non sarebbe possibile più per nessuno ottenere un indirizzo IP univoco!

Per fronteggiare questo problema, l’IEEE ha escogitato 2 sistemi. Il primo consiste semplicemente nell’aumentare la dimensione degli indirizzi IP a 16 byte. La versione del protocollo IP che usa 16 byte si chiama **IPv6** mentre quella “tradizionale” a 4 byte si chiama **IPv4.** Tuttavia, il cambiamento di versione del protocollo non è di facile attuazione, visto che tutti gli apparati del mondo (milioni e milioni di apparati) sono stati progettati e costruiti per funzionare con IPv4. Siccome le due versioni del protocollo non sono *interoperabili* (cioè non possono funzionare insieme sullo stesso apparato) bisognerebbe sostituire o aggiornare tutti i dispositivi che fanno uso di IPv4, un’impresa titanica! Nell’attesa che questa transizione avvenga l’IEEE ha inventato un sistema più ingegnoso e compatibile al 100% con tutti gli apparati attualmente utilizzati in Internet. Questo sistema prevede la suddivisione delle reti che fanno parte di Internet in 2 grandi categorie: le **reti private** e le **reti pubbliche**.

Per capire meglio la differenza fra queste 2 reti, pensiamo alle nostre reti domestiche. Quando ci troviamo a casa e ci connettiamo con un dispositivo (un portatile, un pc fisso, una smart TV, uno smartphone) al WiFi di casa o tramite cavo Ethernet, quello che stiamo facendo è entrare a far parte della rete di casa nostra. Tutti i dispositivi che si connettono ad Internet tramite la nostra rete fanno parte della **rete privata** di casa nostra. Qual è il punto di accesso ad Internet, se utilizziamo il WiFi? Il **router**, a sua volta collegato alla rete Internet grazie ad un certo ISP (Tim, Wind, 3, Vodafone, Linkem). Il collegamento ad Internet può avvenire tramite cavo telefonico, fibra ottica, WiMAX (es. Linkem, Vodafone Station). Il collegamento al router, invece, avviene tramite WiFi o cavo Ethernet. Distinguiamo quindi 2 tipi di “connessione”: la connessione fra il nostro dispositivo e il router e la connessione fra il router e la rete Internet, come illustrato nella figura successiva:



Ovviamente quando ci allontaniamo dalla portata del nostro WiFi o stacchiamo il cavo Ethernet non sfruttiamo più il router per andare su Internet, vale a dire per connetterci ad un qualsiasi sito Internet o utilizzare un qualsiasi servizio di rete (messaggistica, posta elettronica, streaming, etc). Possiamo quindi dire che il router si pone come “cancello di accesso” fra i dispositivi connessi alla nostra rete domestica. Proprio per questo motivo un router che funziona in questo modo viene chiamato anche **gateway**.

In questo scenario, quando dobbiamo connetterci ad un provider di contenuti su Internet non ci connettiamo più “direttamente” ad esso (come abbiamo visto poche pagine fa), ma usiamo il nostro router come intermediario. La comunicazione avviene in questi passaggi:

1. Richiesta dal dispositivo al router: “Chiedi a Spotify di farmi ascoltare una canzone”
2. Richiesta dal router a Spotify: “Fammi ascoltare una canzone”
3. Risposta di Spotify destinata al router: “Ecco la tua canzone”
4. Risposta del router al dispositivo: “Ecco la risposta che mi ha mandato Spotify”

Come ben sappiamo anche un altro dispositivo potrebbe connettersi al router e fare la stessa richiesta. Il router è in grado di capire da chi proviene la richiesta e, una volta ricevuta la risposta da parte del provider di contenuti, inoltrare la risposta al dispositivo corretto. Il provider di contenuti, dal canto suo, non sarà mai in grado di vedere da chi proviene realmente la richiesta. Esso semplicemente dialogherà con il router, completamente ignaro del fatto che “alle spalle” di quel router ci sono i dispositivi che hanno realmente fatto le richieste.

Possiamo identificare quindi due grandi sezioni interconnesse fra loro tramite il router. La rete domestica si dice **rete privata**, mentre la rete Internet si dice **rete pubblica**. Entrambe le reti funzionano secondo il protocollo IP: questo significa che *in entrambe le reti tutti gli host hanno un indirizzo IP*. Ma che differenza c’è fra le 2 reti?

In una rete privata, gli host sono liberi di comunicare fra loro esattamente come avviene su Internet. Essa è tuttavia una rete locale (**LAN - Local Area Network**), nel senso che è formata da pochi host (se confrontata ai miliardi di host che compongono la rete Internet). Le reti private hanno a disposizione dei gruppi particolari di indirizzi IP. In particolare gli indirizzi IP appartenenti agli host di ciascuna rete possono appartenere ad una di queste 3 classi, a seconda della dimensione della rete privata:

A. Da 10.0.0.0 a 10.255.255.255 (consente di avere reti private con più di 16 milioni di host)
B. Da 172.16.0.0 a 172.31.255.255 (consente di avere reti private con poco più di 1 milione di host)
C. Da 192.168.0.0 a 192.168.255.255 (consente di avere reti private con al massimo 65536 host)

Per scoprire qual è l’indirizzo IP privato del nostro computer possiamo effettuare i seguenti passaggi:

- Su windows, premere la combinazione di tasti “Win” + “R” (apre la finestra di dialogo “esegui”) oppure su un sistema operativo Linux o Mac OS aprire “terminale”;

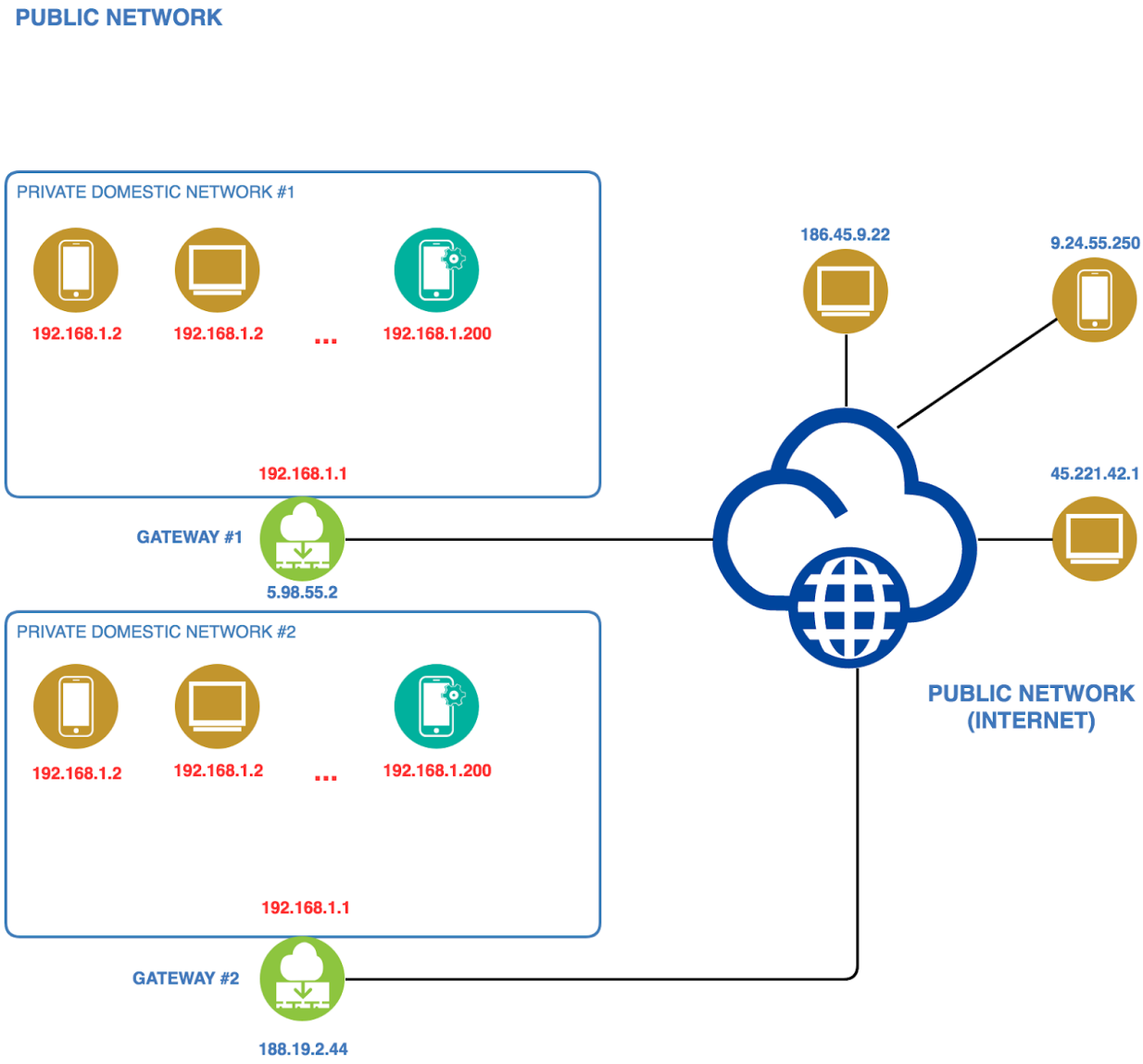
- Solo Windows: digitare “cmd” e premere invio (apre il prompt dei comandi di windows, un’interfaccia a caratteri per usare windows);
- Nella finestra di dialogo che si aprirà digitare “`ipconfig`” (su windows) o “`ifconfig`” su Mac OS o Linux

Fra le informazioni che ci appariranno a schermo, dovremo cercare il tipo di connessione che stiamo utilizzando per comunicare con il router (WiFi, Ethernet, Bluetooth, etc) e cercare la dicitura “Indirizzo IPv4”:

```
C:\Users\Chalie>ipconfig

Configurazione IP di Windows


Scheda LAN wireless Connessione alla rete locale (LAN)* 2:

   Stato supporto. . . . . . . . . . . . : Supporto disconnesso
   Suffisso DNS specifico per connessione:

Scheda LAN wireless Connessione alla rete locale (LAN)* 3:

   Stato supporto. . . . . . . . . . . . : Supporto disconnesso
   Suffisso DNS specifico per connessione:

Scheda LAN wireless Wi-Fi:

   Suffisso DNS specifico per connessione: homenet.telecomitalia.it
   Indirizzo IPv6 locale rispetto al collegamento . : fe80::a194:35d1:676b:5565%13
   Indirizzo IPv4. . . . . . . . . . . . : 192.168.1.227
   Subnet mask . . . . . . . . . . . . . : 255.255.255.0
   Gateway predefinito . . . . . . . . . : 192.168.1.1

Scheda Ethernet Connessione di rete Bluetooth:

   Stato supporto. . . . . . . . . . . . : Supporto disconnesso
   Suffisso DNS specifico per connessione:
```

In questo caso, l’indirizzo IP del computer è 192.168.1.227. Come possiamo notare, con questo comando è possibile anche sapere qual è l’indirizzo IP del gateway (vale a dire del router). La regola dell’univocità dell’indirizzo IP cambia lievemente nel caso di reti private: l’indirizzo IP 192.168.1.227 non può essere utilizzato da un altro host presente nella rete domestica di cui fa parte il computer da cui è stato lanciato il comando “ipconfig”. Tuttavia, in un’altra rete domestica, un altro host potrebbe tranquillamente utilizzare quell’indirizzo IP. Tutti gli host devono avere indirizzi IP diversi solo all’interno della stessa rete privata.

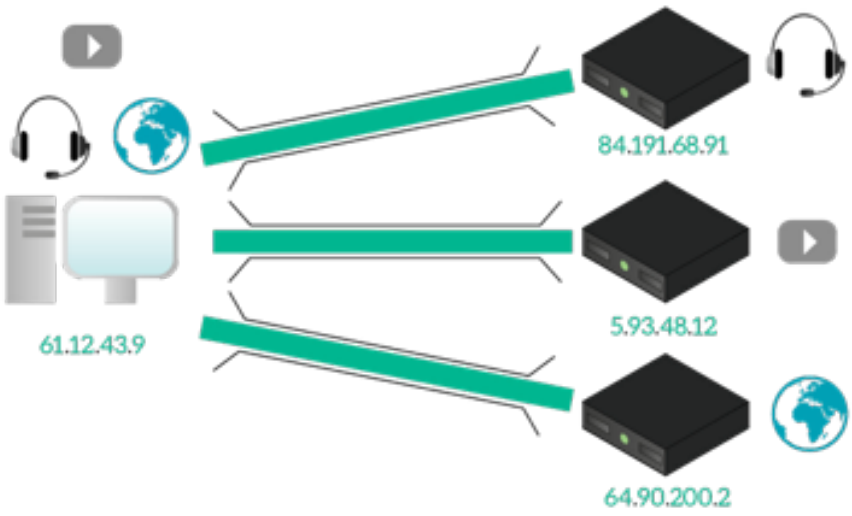
Tutti gli indirizzi IP che non rientrano negli intervalli A, B, C visti in precedenza, tranne alcune eccezioni, sono **indirizzi IP pubblici**. La rete pubblica coincide con la rete Internet. Il fatto che gli indirizzi IP siano pubblici implica il fatto che tutti possono conoscere un indirizzo IP pubblico, e che quindi tutti possono raggiungere in qualsiasi momento un qualsiasi host con indirizzo IP pubblico. I router domestici, fra le loro peculiarità, hanno anche quella di avere 2 indirizzi IP: uno privato, che si interfaccia con la LAN privata domestica, e uno pubblico, che si “affaccia” su Internet e che quindi è in grado di comunicare con tutti gli host connessi ad Internet, facendo da gateway agli host connessi alla rete privata locale. In figura vediamo un esempio di come è strutturata la comunicazione tra reti private e pubbliche in Internet.



Livello di trasporto

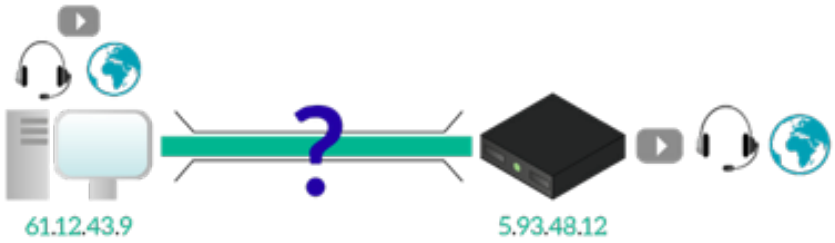
Dalle nostre conoscenze del livello 3 dello stack ISO/OSI dovremmo già sapere che due computer remoti (cioè presenti in punti molto distanti della rete) sono in grado di comunicare sulla rete. Tuttavia, se riflettiamo un attimo sulla comunicazione fra due computer, si scopre che in realtà a comunicare fra loro sono le applicazioni installate su di essi. *Tutto il traffico che proviene da un'applicazione è destinato a un'altra applicazione su un altro computer*. Pensiamo a ciò che quotidianamente facciamo con un computer o con uno smartphone: è molto probabile che navighiamo sul web, guardiamo video online (usando un browser) o facciamo chiamate VoIP, per esempio con Skype. Queste applicazioni richiedono l'accesso ad internet per

comunicare coi rispettivi server, cioè con dei computer remoti che forniscono i contenuti di cui si ha bisogno. Per navigare in Internet, abbiamo bisogno di un server, cioè del computer che contiene la pagina web che stiamo cercando: è questo computer che ci manderà la pagina web ogni volta che gliela chiederemo; per quanto riguarda Youtube, è il computer server che ci manda i video che vogliamo guardare, frame dopo frame, mentre per una chiamata VoIP è il computer del nostro interlocutore che ci invia l'audio.

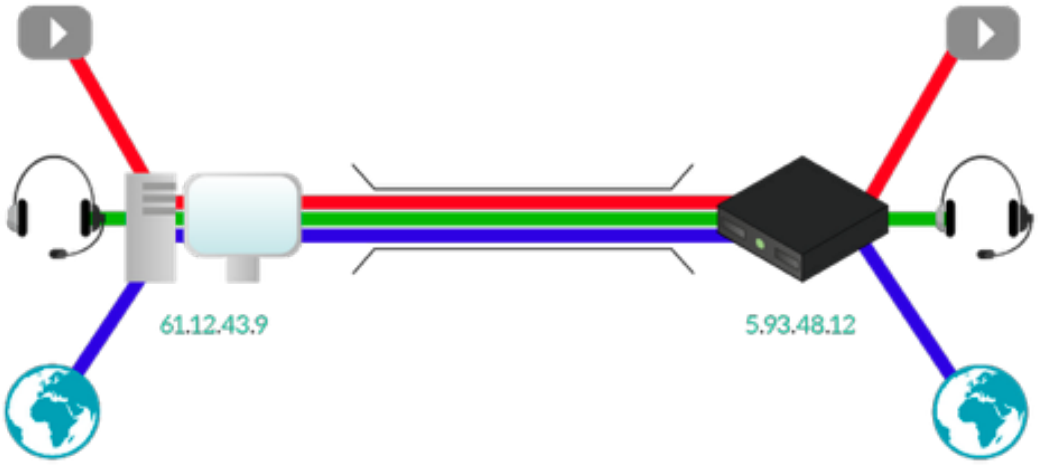


Tutto ciò avviene *su Internet*, e per i computer è piuttosto semplice mandare le richieste verso le **giuste destinazioni**. Quando poi il mittente visualizza la pagina web o il video richiesti, vuol dire che il computer ha spedito il traffico *alla giusta applicazione*, e solo a quella, ragione per cui ogni applicazione può vedere esclusivamente il proprio traffico. Farlo può sembrare facile quando parliamo di server separati; in questo caso infatti, possiamo sapere se un determinato server si sta interfacciando con un'applicazione specifica, mentre un'altra applicazione sta comunicando con un altro server, proprio come nella figura sottostante.

Grandioso, ma non possiamo presupporre che ogni applicazione comunicherà con un server separato dedicato solo a quello. Per esempio, YouTube è un sito che contiene video: come si identificano e si differenziano il traffico della pagina web dal traffico dei video? La faccenda potrebbe diventare ancora più difficile quando si comunica con un singolo server per molteplici applicazioni, quindi non possiamo fare sempre affidamento sui canali separati IP (ovvero su diversi indirizzi IP dei server).



Il livello di trasporto è stato progettato proprio per risolvere questo problema, e lo fa in una maniera terribilmente semplice. A livello di trasporto, possiamo dare un “indirizzo” ad ogni applicazione che vuole comunicare sulla rete Internet. Esso identifica in maniera univoca **una applicazione installata su un computer**, ed è un intero di 16 bit (quindi da 0 a 65535) chiamato **numero di porta**. Un’applicazione può chiedere di ricevere uno specifico numero di porta, ma è il sistema operativo del dispositivo che deve assicurarsi che esso non sia già usato da un’altra applicazione. Grazie a questi numeri siamo in grado di mantenere separati i flussi di traffico che entrano ed escono da un computer.



Coi numeri di porta, il traffico di un’applicazione rimane separato da quello delle altre applicazioni a prescindere dall’indirizzo IP.

È importante notare come il numero di porta sia univoco solo *all’interno di uno stesso computer*: quando un computer riceve un pacchetto IP da un qualsiasi altro dispositivo, inoltrerà il traffico all’applicazione giusta basandosi su questo numero di porta. La consegna del pacchetto al computer giusto è sempre compito del routing del livello di rete (e di IP in particolare). Conseguentemente, un numero di porta può essere usato per *identificare in maniera univoca un’applicazione su un computer*, mentre un IP pubblico è usato per *identificare*

in maniera univoca un computer su Internet. Questo significa che la combinazione di *indirizzo IP e numero di porta* **identifica in maniera univoca un'applicazione su Internet**.

Questa combinazione di indirizzo IP e numero di porta prende il nome di **socket**: insieme, due socket identificano una comunicazione fra due applicazioni su Internet.

Dato che il numero di porta è un numero intero, viene rappresentato come un numero decimale positivo e quando dobbiamo annotare un socket dobbiamo solo scrivere l'indirizzo IP e numero di porta, separati da due punti, cosa abbastanza semplice in IPv4, dal momento che un indirizzo IP è scritto con 4 interi separati da un punto (e.g. `10.10.10.91:80`), ma un po'più complessa con IPv6, visto che gli indirizzi IPv6 contengono già come separatore i due punti. Per questo motivo usiamo in IPv6 usiamo le parentesi quadre per separare IP e numero di porta: `[2001:db8:acad::12]:80`.

Due socket identificano in maniera univoca una comunicazione fra 2 applicazioni su Internet.

Il concetto è abbastanza semplice: bisogna dare un'identità all'applicazione per mantenere il traffico separato. Ad ogni modo, non sappiamo ancora *come concretamente venga implementata* questa separazione. Proviamo a indovinare: l'applicazione (ad esempio un browser, un gioco online, Skype etc.) passa i suoi dati al livello di trasporto, che aggiunge dei dati extra. I dati delle applicazioni più gli elementi aggiunti a questo livello, formano una PDU di livello di trasporto, conosciuta come Segmento o Datagram. L'intestazione del segmento è diversa a seconda del protocollo usato (TCP o UDP), ma a prescindere dal protocollo utilizzato due elementi saranno sempre presenti: il numero di porta *sorgente* e il numero di porta *destinazione*. Proprio come indica il nome, il numero di porta sorgente identifica l'applicazione sul dispositivo che ha creato quel segmento, mentre il numero di porta destinazione identifica l'applicazione sul dispositivo a cui quel segmento è destinato.

I protocolli principali di livello di trasporto sono 2: TCP (Transmission Control Protocol) e UDP (User Datagram Protocol). Essi hanno in comune il fatto di garantire il servizio di consegna dei pacchetti di dati alle corrette applicazioni utilizzando la tecnica del **multiplexing** e **demultiplexing**. La prima serve a convogliare all'interno di una singola PDU di livello di trasporto (chiamata anche **segmento**) dati provenienti da più applicazioni. La seconda invece serve a effettuare l'operazione contraria: lo smistamento dei pacchetti di dati all'applicazione giusta partendo da una PDU di livello di trasporto. La differenza fra i due protocolli risiede nel

fatto che TCP viene usato per fornire un *servizio affidabile*, cioè un servizio che gestisce i casi di errori o di pacchetti di dati smarriti o scartati durante l'inoltro, garantendo una consegna sicura al destinatario dei messaggi; il tutto al prezzo di una maggiore lentezza nella comunicazione. Al contrario UDP fornisce un *servizio non affidabile*: i pacchetti di dati viaggiano sulla rete senza nessuna garanzia sulla loro effettiva consegna, ma in generale una comunicazione basata su UDP è più veloce di una basata su TCP.

In TCP un socket è formato dai seguenti campi:

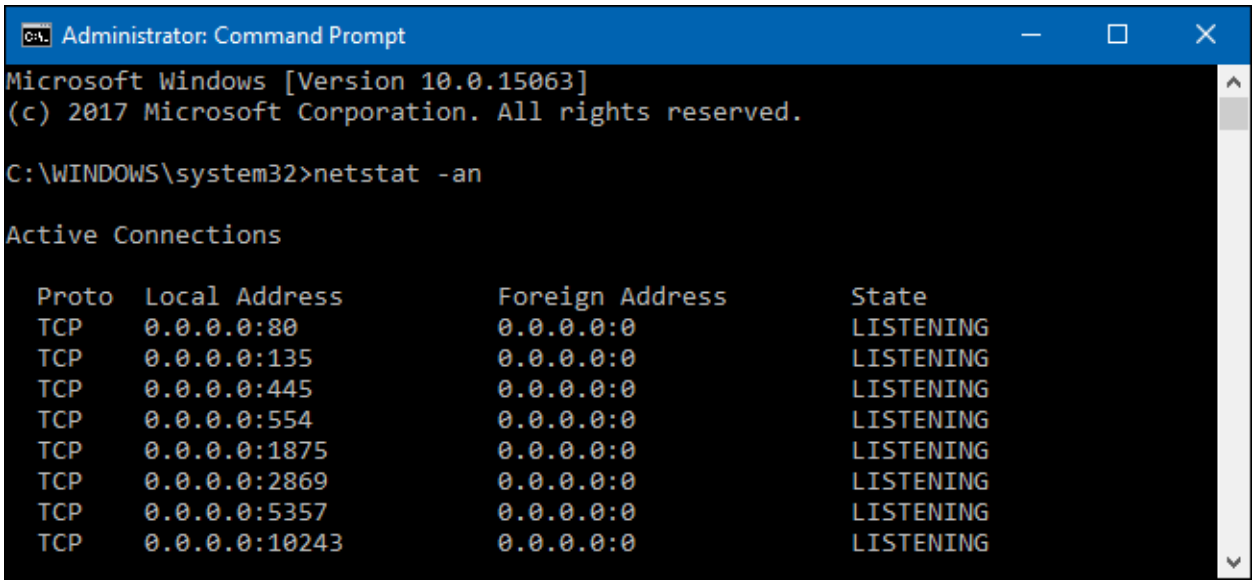
- Indirizzo IP mittente;
- N. porta mittente;
- Indirizzo IP destinazione;
- N. porta destinazione;

In UDP invece un socket è formato semplicemente da:

- Indirizzo IP
- N. porta

È possibile, su un computer, verificare lo stato dei socket aperti usando da prompt dei comandi o da shell il comando `netstat`

Questo è un esempio dell'output di questo comando:



Le varie colonne indicano rispettivamente:

- Protocollo a livello di trasporto di quel socket
- Indirizzo IP locale + n. Porta
- Indirizzo IP del computer remoto + n. Porta del programma remoto
- Stato del socket

In questo caso lo stato dei socket è "listening". Capiremo studiando il livello applicativo che cosa significa.

Proprio come con gli indirizzi IP abbiamo alcuni indirizzi specifici (privati e pubblici), anche per i numeri di porta abbiamo alcuni intervalli specifici: non tutte le applicazioni possono usare gli stessi numeri di porta. In particolare, possiamo dividere i numeri di porta in due intervalli: **Porte Well-Known** e **Porte Dinamiche o Private**. La finalità e i dettagli dei seguenti intervalli sono contenuti nella tabella seguente:

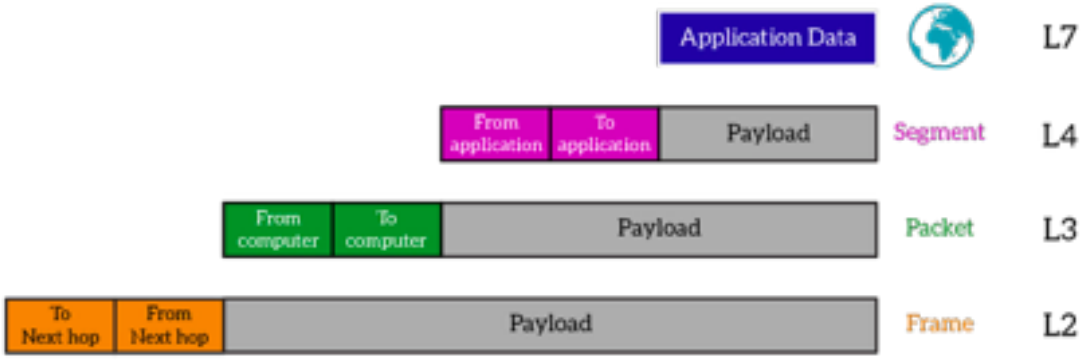
Nome intervallo	Intervallo	Descrizione
Well-known	0-1023	Le porte in questo intervallo sono usate per servizi “noti” (come HTTP, FTP, DNS etc.) e sono generalmente usate sul dispositivo che offre il servizio (il server).
Dynamic	1024-65535	Porte usate da applicazioni che non hanno uno specifico protocollo applicativo associato

Alla luce di quello che abbiamo detto finora sui livelli data-link, network e trasporto, possiamo dire una cosa molto importante. Le PDU di tutti e 3 questi livelli individuano degli indirizzi diversi a seconda della funzione offerta da livello.

Il livello data-link offre il servizio di comunicazione punto-punto. Gli indirizzi che vengono specificati nella sua PDU sono gli indirizzi fisici (MAC) delle schede di rete degli host successivi lungo il percorso che porta da una sorgente ad una destinazione.

Il livello di rete offre il servizio di comunicazione end-to-end. Gli indirizzi che vengono specificati nella sua PDU sono gli indirizzi IP dei nodi terminali della comunicazione (sorgente e destinazione).

Ili livello di trasporto offre il servizio di multiplexing/demultiplexing. Gli indirizzi che vengono specificati nella sua PDU sono i numeri di porta delle applicazioni che vogliono comunicare in rete sfruttando la rete Internet.

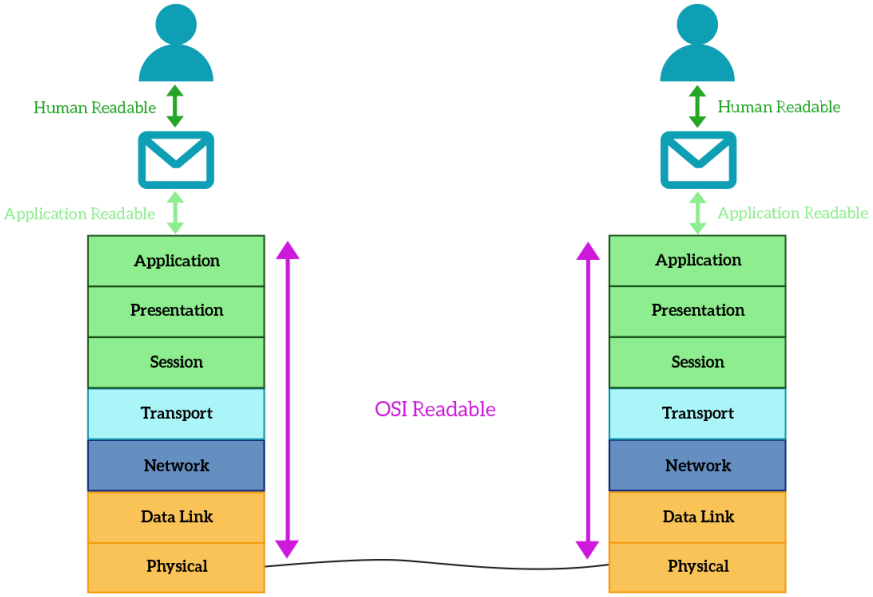


Livello applicativo

Finora abbiamo parlato in generale di “comunicazione” fra host. Solo nel livello di trasporto ci siamo posti il problema su *chi* realmente comunica su Internet. I socket identificano una particolare applicazione sulla rete Internet, e il livello di trasporto si occupa di consegnare i pacchetti di dati ai corretti destinatari: le applicazioni. Il tutto avviene utilizzando i numeri di porta, che sono semplicemente gli “indirizzi” delle applicazioni.

Faremo ora una panoramica sul livello applicativo, il più utilizzato dagli utenti finali. A questo livello si trovano le applicazioni, i software che vengono usati dagli utenti finali per usufruire dei servizi di rete: posta elettronica, World Wide Web, streaming video, streaming audio, VoIP (Voice over IP), IM (Instant Messaging), live streaming, gaming, ecc. ecc.

Innanzitutto chiariamo che con il termine “applicazioni” non intendiamo solo le app installate sui nostri smartphone, ma un qualsiasi software in grado di comunicare su Internet installato su una qualsiasi piattaforma (desktop, laptop, smartphone, etc): a questo proposito si dice che le applicazioni che utilizzano Internet per fornire i propri servizi si chiamano **applicazioni di rete**. A prescindere da quello a cui servono (navigare in Internet, mandare/leggere messaggi di posta elettronica), tutte le applicazioni hanno uno scopo comune: quello di *fornire agli utenti finali un’interfaccia per utilizzare lo stack ISO/OSI o TCP/IP*. Le applicazioni si **interfacciano con gli utenti umani**, prendono informazioni da loro e le trasformano in informazioni capaci di attraversare lo stack TCP/IP, viaggiare in rete ed essere comprese dall’applicazione del destinatario della comunicazione:



Tutte le applicazioni offrono un particolare servizio di rete fra quelli già elencati in precedenza. Ogni servizio, a sua volta, è rappresentato da uno o più protocolli. Quindi si può dire che le applicazioni di rete non sono altro che delle implementazioni di un particolare

protocollo di livello applicativo. Esamineremo ora alcuni dei servizi più utilizzati della rete Internet.

World Wide Web

Il World Wide Web (abbreviato in WWW) è il servizio più utilizzato di Internet. Quando qualcuno dice “cerco una particolare informazione su Internet”, in realtà sta dicendo (oltre a causare un attacco di cuore ad un ingegnere delle telecomunicazioni) “cerco una particolare informazione sul Web”. C’è una grossa differenza fra il Web e Internet. Internet è l’insieme di tutti i dispositivi dotati di indirizzo IP pubblico connessi fra loro in tutto il mondo. Il web, al contrario, è un sottoinsieme di Internet formato dai dispositivi che contengono i contenuti web. Tipicamente questi contenuti sono aggregati in *siti web*, e sono formati da pagine di testo, immagini, video, documenti di varia natura, che possono essere raggiunti e ottenuti dagli utenti su loro richiesta.

Per fornire questo servizio di rete viene utilizzato fondamentalmente un protocollo chiamato **HTTP**: **Hyper Text Transfer Protocol**. Il nome del protocollo è abbastanza autoesplicante: esso serve a trasferire ipertesti fra 2 applicazioni (un client e un server). Ma cos’è un ipertesto? Esso è un file di testo semplice (cioè un file di testo contenente solo caratteri ASCII o Unicode direttamente comprensibili dall’utente finale) usualmente codificato in un linguaggio che si chiama **HTML** (**Hyper Text Markup Language**). Esso è uno standard che definisce un linguaggio di formattazione dei documenti. Nella quasi totalità dei casi, quando navighiamo su un sito web non stiamo facendo altro che consultare le pagine HTML che formano quel sito. HTTP conta due versioni: la versione 1.0 e la versione 1.1, quella tuttora in uso; attualmente è ancora in fase di sviluppo e test la versione 2.0.

HTTP è un protocollo di tipo client/server. Significa che le entità coinvolte nella comunicazione sono solo 2: un client che fa delle domande e un server che fornisce delle risposte. Questo implica che anche i tipi di messaggio che viaggiano su HTTP sono solo 2: le richieste HTTP (generate dal client) e le risposte HTTP (generate dal server). Un’applicazione di rete che faccia da client HTTP viene chiamata **browser**. In generale invece un’applicazione che faccia da server HTTP viene chiamata **Server HTTP** o **demone HTTP**: un server HTTP è un programma che sta in ascolto sulla **porta 80**. Questo significa che i client che vogliono una certa risorsa disponibile su HTTP devono effettuare delle connessioni che abbiano nel campo “porta destinazione” la porta 80.

Pensiamo a quando facciamo vogliamo consultare un sito Internet, per esempio il sito di un giornale online. Il browser effettua le richieste HTTP ad un determinato server, cioè a un demone HTTP in ascolto su porta 80 che risiede su un host che ha un certo indirizzo IP. Quando l’utente digita l’indirizzo del sito web sulla barra degli indirizzi del browser e preme “invio”, il browser crea la richiesta HTTP, che non è altro che una stringa di testo plaintext. In essa sono specificati:

- il tipo di operazione da effettuare (lettura/scrittura/modifica/cancellazione)
- l’indirizzo HTTP della risorsa richiesta (chiamato URL - Uniform Resource Locator)
- altre informazioni come la lingua in cui si preferisce ricevere la richiesta, informazioni sul client (S.O., versione del browser, etc)

Ecco un esempio di richiesta HTTP ad un file che si chiama “index.html” presente sul sito internet “www.repubblica.it”:

```
    GET /index.html HTTP/1.1
    Host: www.repubblica.it
    User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:50.0)
Gecko/20100101 Firefox/50.0
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
    Accept-Encoding: gzip, deflate
    Connection: keep-alive
    Upgrade-Insecure-Requests: 1
    Pragma: no-cache
    Cache-Control: no-cache
```

Facendo l’analogia per cui ogni protocollo è assimilabile ad una lingua diversa, notiamo la prima riga, in cui c’è il valore “`GET index.html http 1.1`”: esso, fondamentalmente significa: “Ciao, sono un client che parla la lingua HTTP versione 1.1 Mi dai il file index.html?”. Il “GET” iniziale, infatti, indica che il tipo di operazione che vogliamo fare sulla risorsa specificata (il file “index.html”) è un’operazione di *lettura*. “GET” in questo caso prende il nome di **metodo HTTP**. Un metodo indica quindi l’operazione (lettura, scrittura, modifica, cancellazione) che vogliamo effettuare su una risorsa. I metodi HTTP principali sono:

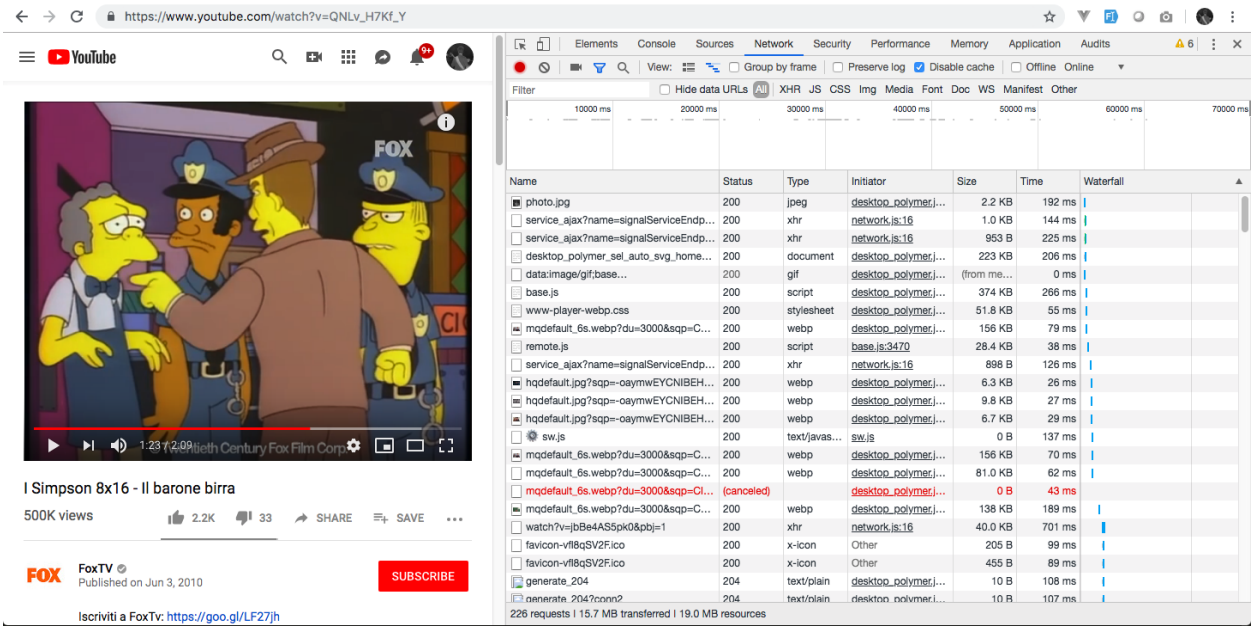
- GET: operazione di lettura
- POST: operazione di scrittura (pensate a quando ci iscriviamo ad un sito web, o scriviamo un post sul nostro social preferito)
- PUT: operazione di modifica (es. un cambio password di un account)
- DELETE: operazione di cancellazione (es. rimozione di una foto dal nostro feed Instagram)

Come viene identificata una risorsa sul Web? Tramite il suo URL. **L’URL (Uniform Resource Locator) è semplicemente l’indirizzo web della risorsa che vogliamo consultare** su un server web, vale a dire ciò che vediamo sulla barra degli indirizzi del nostro browser.

Fondamentalmente un URL è formato da:

- Protocollo di accesso a quella risorsa (tipicamente `http://`)
- Indirizzo dell’host su cui si trova una risorsa (può essere un indirizzo IP o un nome di dominio che verrà convertito in indirizzo IP dal protocollo DNS)
- Numero di porta (se non specificata, si intende la porta 80)
- Percorso della risorsa specificata all’interno del server specificato

Tornando al nostro esempio, quando visualizziamo una pagina web di solito essa è piena di immagini, caratteri particolari, video, audio. Anche se fanno parte della stessa pagina web, tutte questi elementi che compongono la pagina sono a loro volta delle risorse web. Pertanto il browser una volta analizzata la pagina, farà altre richieste per ogni risorsa esterna che trova sulla pagina stessa. È possibile vedere quali sono le richeiste effettuate dal browser per visualizzare una pagina utilizzando i cosiddetti “strumenti per gli sviluppatori” messi a disposizione da ormai quasi tutti i browser. Per attivarli su Google Chrome si può utilizzare la combinazione di tasti CTRL+SHIFT+C (CMD + OPTION + I su Mac), oppure mentre si naviga cliccare con il pulsante destro su una parte qualsiasi della pagina web e cliccare la voce “Ispeziona elemento”. Si aprirà una finestra con svariate schede: per vedere quali risorse sono state caricate dall’esterno basterà andare alla scheda “Network”. Per catturare le richieste è necessario che il pallino rosso in alto a sinistra nella scheda “network” sia acceso.



È interessante notare come per la visualizzazione di una semplice pagina di youtube vengano effettuate così tante richieste. Google Chrome ci dà la possibilità di filtrare i contenuti caricati per tipologia (es. In alto nella scheda network ci sono i filtri “XHR”, “JS”, “CSS”, “IMG”, “MEDIA”) e anche la possibilità di scaricare anche in questo modo le risorse visualizzate (video, audio, immagini, font). Basterà individuare la risorsa desiderata all’interno di questa scheda, cliccarci su con il pulsante destro e cliccare su “Apri in una nuova scheda”. Dalla nuova scheda, successivamente sarà possibile salvare sul proprio computer la risorsa richiesta.

Abbiamo detto che ad ogni richiesta corrisponde una risposta da parte del demone HTTP. Esso risponde prima di tutto con dei numeri, chiamati **status code**, che indicano se la richiesta del client è andata a buon fine o se c’è stato un errore, specificando anche la tipologia di errore (in figura possiamo notare gli status code nella seconda colonna della scheda “network”, indicata appunto con “status”).

Ecco un elenco di status code notevoli:

- 200: OK, richiesta esaudita correttamente
- 404: FILE NOT FOUND; solitamente questo avviene quando c'è un errore nell'URL (es nome del file richiesto sbagliato)
- 500: INTERNAL SERVER ERROR, errore interno del server, riprovare più tardi

Posta elettronica

La posta elettronica è uno dei servizi di rete più utilizzati assieme al WWW. Essa è composta da 2 tipi di protocolli, entrambi di tipo client/server, uno per la lettura delle mail e uno per l'invio delle mail.

Essi non saranno trattati nel dettaglio, ma verranno solo elencati i nomi di questi protocolli con le loro funzionalità e i numeri di porta utilizzati:

- SMTP (Simple Mail Transfer Protocol): è utilizzato per l'invio di messaggi su una mailbox. Un server SMTP utilizza le porte 25, 465 o 587 (una delle 3)
- POP3 (Post Office Protocol v. 3): è utilizzato per l'accesso alle mailbox (quindi per leggere le mail). Un server POP3 utilizza le porte 110 o 995
- IMAP (Internet Message Access Protocol): è utilizzato per l'accesso alle mailbox, in alternativa a POP3. Un server IMAP utilizza le porte 143 o 993

Shell remota

È possibile controllare un computer da remoto (o meglio: avviare una shell su un computer remoto) utilizzando il protocollo **SSH** (Secure SHell). Anch'esso è un protocollo di tipo client/server in cui il server sta in ascolto sulla porta 22. Per avviare una sessione SSH su un computer su cui sia installato un server SSH in ascolto sulla porta 22 è necessario specificare l'indirizzo IP e il nome utente con cui ci si vuole loggare al comptuer remoto.

```
Mac-mini-di-Mac:~ giumast$ ssh -p 50000 giuseppe@192.168.1.8
giuseppe@192.168.1.8's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.18.0-15-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
   directly, see https://bit.ly/ubuntu-containerd or try it now with

     snap install microk8s --channel=1.14/beta --classic

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

156 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
*** System restart required ***
Last login: Mon Apr 29 00:23:59 2019 from 192.168.1.139
giuseppe@mediaserver:~$
```

L’istruzione per connettersi è la prima in figura. Notiamo come in questo caso il server fosse in ascolto sulla porta 50000 piuttosto che su quella classica 22. Facciamo caso anche alla sintassi del comando:

```
ssh -p PORT_NUM utente@ipaddress
```

In questo caso:

- PORT_NUM = 50000,
- utente = giuseppe
- ipaddress: 192.168.1.8

Notiamo come, alla riga successiva, venga richiesta dal computer remoto la password dell’utente su quel computer (in questo caso “giuseppe”), e una volta fornita, il demone SSH dia accesso al computer remoto; possiamo renderci conto di come l’accesso sia avvenuto correttamente analizzando l’ultima riga in figura: da quel momento in poi ogni comando inviato su quella riga di comando sarà eseguito sul computer remoto.

Tabella protocolli/descrizione/numeri porta

Ecco una tabella contenente alcuni fra i più utilizzati protocolli di rete client/server, una breve descrizione di essi e i numeri di porta utilizzati dai **server** di questi protcolli. Notiamo come, essendo tutti protocolli ufficiali, i numeri di porta utilizzati siano tutti **minori di 1024**.

Protocollo	Descrizione	Porte
HTTP	Protocollo per il trasferimento di risorse sul WWW	80, 443
SMTP	Invio di messaggi di posta elettronica	25, 465, 587
POP3	Accesso a mailbox (lettura messaggi di posta elettronica)	110, 995
IMAP	Accesso a mailbox (lettura messaggi di posta elettronica)	143, 993
FTP	Trasferimento file (solitamente usata per mettere online dei siti web)	20, 21
SSH	Shell remota sicura (controllo computer remoti)	22

Bibliografia

K. W. Ross, J. F. Kurose - Reti di calcolatori e Internet. Un approccio top-down (2017, Pearson)
Alessandro Maggio - Free CCNA Course - www.ictshore.com

Askleo.com - What is “ping” and what does its output tell me?