Riassuntone RETI

Cosa troviamo all'esame?

- Come funzionano le reti
- Domande pratiche che riguardano il progetto

Inizio

Internet: una ragnatela di collegamenti; possiamo immaginarlo come un grafo dove ***i nodi*** sono gli **endpoint/routers** e gli ***archi*** sono i **link.**

endpoint: sono i **nodi foglia**. Sono quelli che ricevono i dati dall'esterno e li trasmettono a loro volta. Un **server**, ad esempio, è un endpoint; non è mai in posizione intermedia. Sono anche chiamati **host**

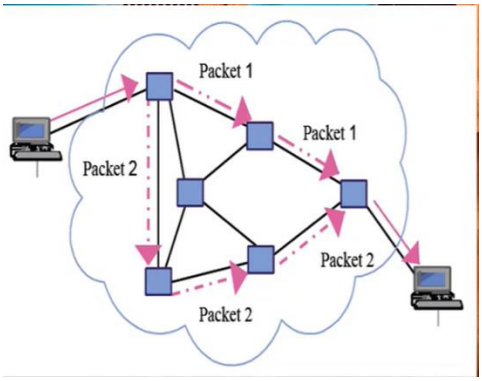
- **router**: sono coloro che si occupano di **instradare i pacchetti***;* i pacchetti sono *blocchi di dati*
- **Link**: il collegamento fisico che c'è tra un nodo ed un altro.

Altre informazioni utili:

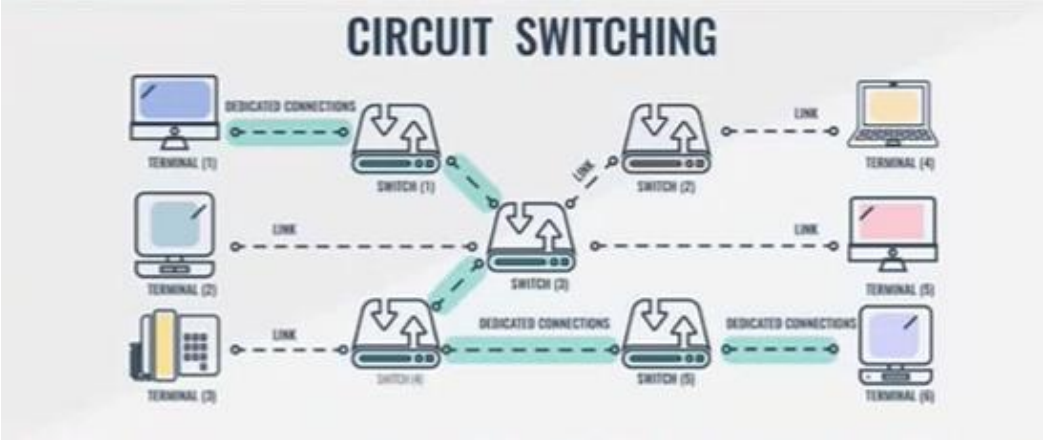
- **Banda:** velocità di trasmissione in *bit al secondo*
- **Protocolli:** sono delle **regole di trasmissione** dove i vari nodi sono d'accordo per comunicare tra loro. In particolare *specificano le modalità di* **invio e ricezione** dei pacchetti.
- **Pacchetti:** il pacchetto possiamo pensarlo come **una lettera** che dev'essere messa in busta e poi smistata.

Macro tipi di rete

- **Commutazione di pacchetto:** abbiamo due **host** che comunicano e dei pacchetti da mandarsi a vicenda. I **router** instradano i pacchetti e li fanno arrivare all'host destinatario.

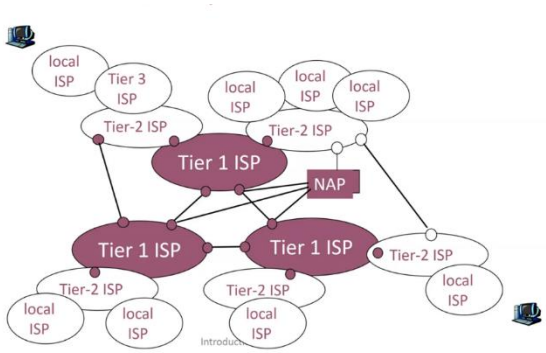


- **Commutazione di circuito:** ci sono end-point e centralini che per comunicare hanno bisogno di un **collegamento fisico riservato** per ogni singolo end-point. La differenza con la *commutazione di pacchetto* è che le linee sono linee sono finite e quindi vengono bloccate quando sono tutte in utilizzo.



Rete delle reti

Quando degli endpoint vogliono comunicare fanno affidamento ad una struttura simile:



È letteralmente un grafo.
I vari termini significano:

- **ISP:** internet service provider
- **AS:** autonomus system – gruppo di router e reti sotto il controllo di un'autorità amministrativa
- **NAP:** Network Access Point – un dispositivo dove arrivano fili da più ISP

Tipologie di link

- **Link punto-punto:** connette due interlocutori direttamente
- **Link broadcast:** non c'è un filo diretto, ma delle stazioni affacciate sullo stesso link
    - **Wifi casalingo:** ha un access point che raggiunge tutte le stazioni collegate all'access point

Internet e cosa offre

Metodi di comunicazione di internet (nomi metaforici)

- **A piccione viaggiatore**: mandare singoli pacchetti nella rete; corrisponde all'utilizzo del protocollo **UDP**.
    - *Poco affidabile – il pacchetto potrebbe perdersi*
- **A tubo**: sono delle conversazioni punto – punto. Corrisponde ad utilizzare il protocollo **TCP**.
    - *Massima affidabilità.*

Stack di protocolli

I protocolli di comunicazione nascono per definire come *degli interlocutori* devono conversare.

Ad ogni messaggio speciale ci sono delle azioni che vengono affettuate

Un **protocollo di comunicazione** è definito:

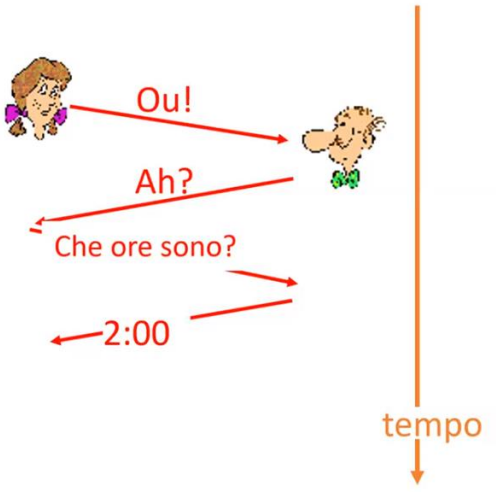
- Individuando una serie di messaggi speciali
- Assegnando delle azioni speciali da fare alla ricezione di messaggi speciali

Pensa come un **automa a stati**.

"I protocolli definiscono il formato, l'ordine e il significato dei messaggi, le azioni da compiere all'atto dell'invio e della ricezione, la dimensione degli spinotti, i materiali usati, ecc. ecc."

Protocollo tra umani

C'è un certo tempo tra **l'invio del messaggio** e la **ricezione del messaggio**. Il fatto che la freccia sia inclinata indica un **tempo che passa**.

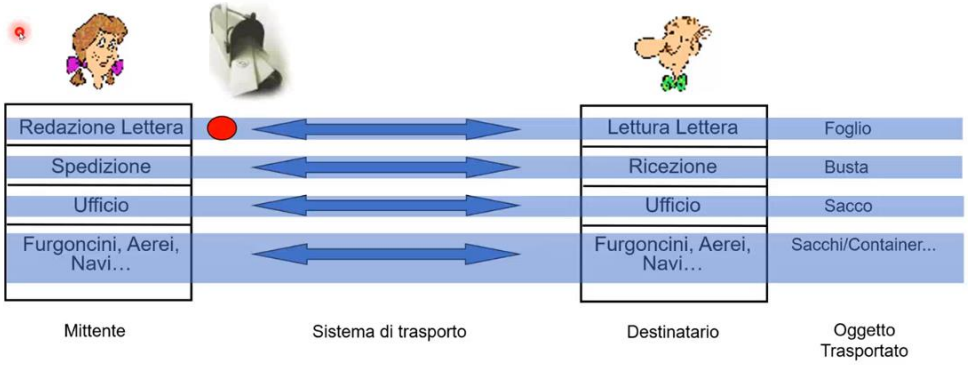


Protocollo tra hosts

Il discorso è simile. C'è una **richiesta** che determina una specifica **risposta** ed un lasso di tempo che c'è tra la richiesta e la risposta tra host.

Questa overview ci aiuta a capire il **protocollo a strati**

Gerarchia di protocolli / stack di protocolli



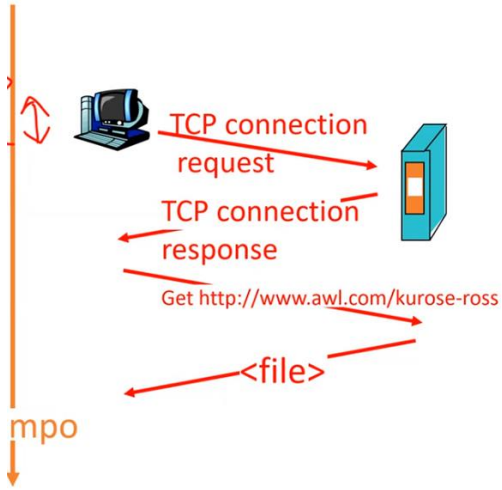
Alice deve mandare una lettera a Bob – va all'ufficio postale e la imbuca; quello che succede dietro per ora non ci interessa.
Bob riceve la lettera a casa.

Risultato: conversazione diretta sfruttando dei **canali di comunicazione.**

Ufficio postale = **router –** i vari router conversano tra di loro. Ergo, le lettere vengono mandate tra vari router.

Motivazione delle gerarchie di reti

- **Pro:**
  - Modificare uno strato non modifica gli altri – esempio: cambiare modello di un aereo non cambia niente al sistema di trasporti
- **Contro:**



- Potrebbe portare ad inefficienze e duplicazione di compiti

Lo stack di protocolli

- **Livello applicazione:** supporta le applicazioni di rete
  - FTP, SMTP, HTTP

- **Livello trasporto:** servizi di trasmissione punto – punto
  - TCP, UDP

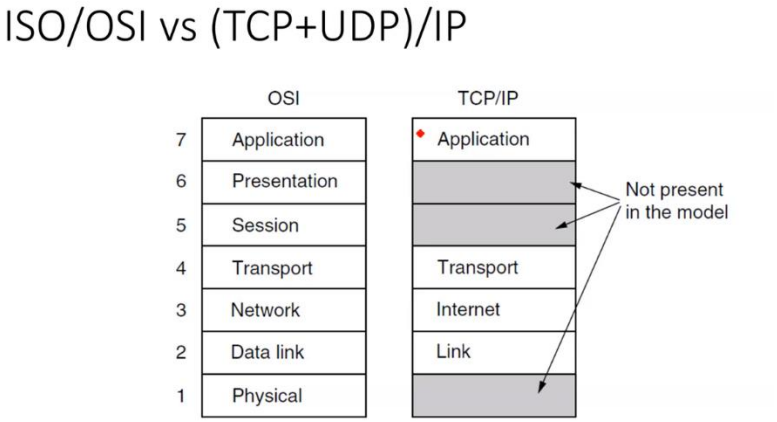
- **Livello network:** sistema di instradamento dei datagrammi
  - IP, Protocolli di routing

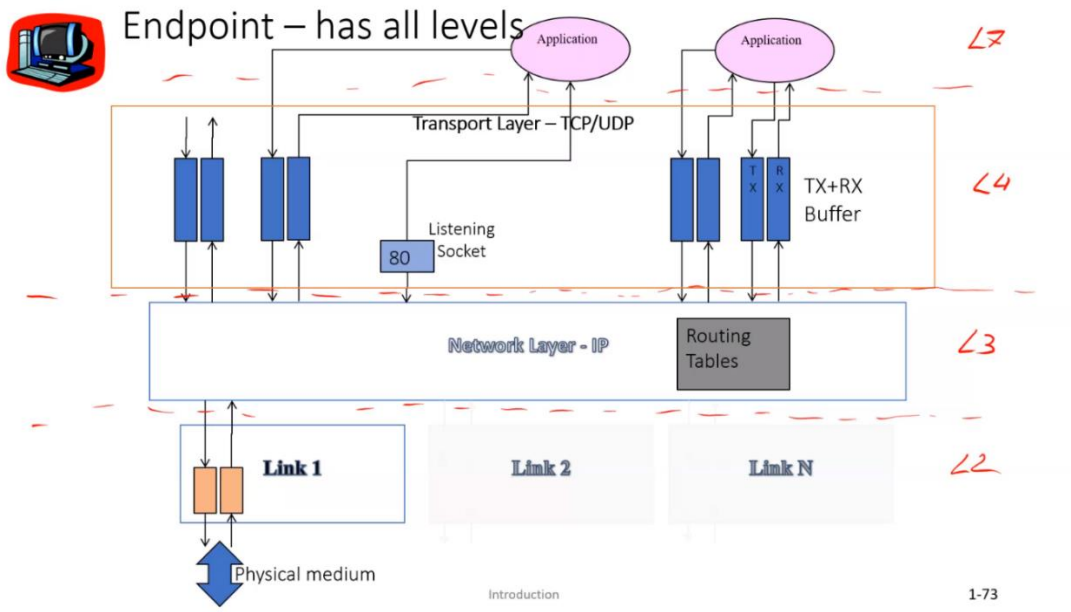
- **Link fisico:** servizi di trasmissione tra host fisicamente adiacenti (trasmissione diretta)
  - **PPP, Ethernet, Wi-Fi, LTE, 5G**

Pila ISO/OSI

L'architettura faceva riferimento a 7 strati; alcune cose non avevano implementazione nella vera **internet** e quindi si utilizza lo stack **TCP/IP**

Livelli nell'End-Point:
immagine molto importante per fissare bene i concetti





- **L2:**
  - implementato nelle schede di rete
  - implementato nei software nei driver della scheda di rete
- **L3**:
  - Offre ad L4 la possibilità di inviare i pacchetti e di riceverli
- **L4**:
  - Fornisce un'interfaccia agli applicativi allo strato di sopra
- **L7**:
  - Sono le applicazioni; i processi, i programmi, ecc..

Cosa succede nei vari livelli nel dettaglio

- **L7**: ci sono i **processi** e per *comunicare* sfrutta lo strato inferiore. Usa un **listening socket** che è associato ad un certo numero di porta.
  I rettangoli blu sono dei **buffer** che contengono array di byte; servono ad **inviare e ricevere** pacchetti / quindi a comunicare con l'interlocutore esterno.

- **L4:** raccoglie i dati che arrivano dalla rete. I pacchetti sono gestiti per essere inseriti nel *buffer di ricezione* (multiplexing = smistare).
  Vengono anche presi *i dati nel buffer di spedizione* e si interfaccia col **livello 3**

- **L3:** offre a 4 la possibilità di inviare e ricevere pacchetti. Possiamo immaginare all'interno un *mega buffer di invio e ricezione*; **incanala tutto quello che entra ed esce dall'end-point**. All'interno ci sono le **routing tables.**
  Viene preso un **datagramma** da L4 e va scelto in **quale scheda di rete usare;** questo viene fatto attraverso le **routing tables.**
  Una volta scelta la scheda, bisogna interfacciarsi con essa, quindi mettere nel **buffer di uscita o ricezione** dipendentemente se dobbiamo mandare o ricevere.
  N.B: In ricezione L3 riceve degli **interrupt** quando una scheda segnala che ci sono

dati in arrivo. Quando arriva questo interrupt il dato viene prelevato e vengono fatte le elaborazioni del caso.

- **L2:** è connesso ad una linea.
  invio:
  ricezione: arriveranno dati sulla linea fisica; bisogna convertire tutto e si poggia sul buffer di ricezione.

Cosa succede in un router puro:

riceve datagrammi; invia datagrammi; il datagramma viene preso, convertito e poggiato sul buffer di ricezione.
Il datagramma viene estratto dal buffer di ricezione e viene processato e poi spedito consultando le **routing tables**. Si sceglie su quale link instradarlo in particolare.

L2 Device: un device immaginato come una scatoletta che ha tante interfacce di rete; serve a **giuntare** i link che sono collegati. Riceve e manda i dati verso tutti, come un ripetitore.

Esempio di spedizione di un pacchetto analizzando i vari strati

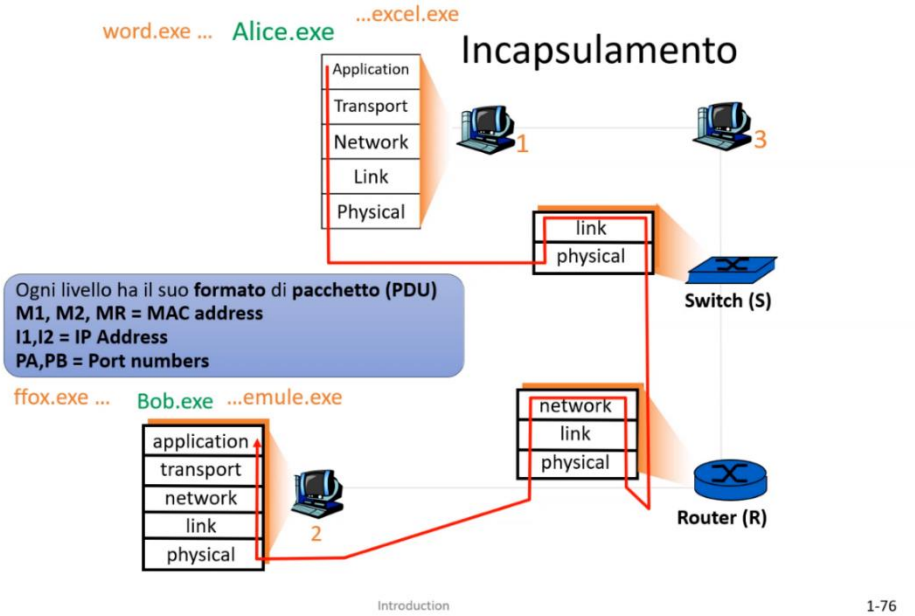


Figura 1: Immagine di riferimento

L7 – Livello applicazione: immaginiamo di scrivere qualcosa in un campo di testo su chrome. Si scrive il **messaggio** nel buffer collegato presente nel **layer di trasporto**.

L4 – Livello trasporto: il messaggio viene **incapsulato** e prende il nome di **semeento;** vanno inseriti in particolare gli estremi, cioè **porta sorgente e porta destinazione.** Serve a multiplexare in modo corretto

L3 – Network: avvio del **processo di routing;** il segmento viene inserito in contenitore ancora più grande, chiamato **datagramma**; qui vanno specificati **indirizzo ip sorgente** e **indirizzo ip destinazione.**

L2 – Link: il datagramma viene incapsulato in un contenitore chiamato **frame;** qui vanno inserite le specifiche della linea: **stazione mittente** e **stazione destinatario.** Visto che non c'è un collegamento punto – punto, noi abbiamo *ip destinatario quello di PC2,* ma come **MAC ADDRESS** serve quello del router che poi dovrà instradare il pacchetto verso PC2.

Il pacchetto viene mandato, entra nello switch (che ripete la stessa informazione che rientra sul link), arriva nel router. Il router consulta le routing table e sceglie il **next hop** a cui mandare il frame. Individua che il frame va mandato a **PC2**;
Importante: il **datagramma** viene tolto dal frame e viene inserito in un altro frame, che ha come stazione mittente la **scheda di rete del router R,** e come stazione destinataria quella di **PC2.**

Il frame entra a livello fisico in PC2; trasforma il segnale in **frame**; controlla il MAC e se è corretto viene sbustato il frame e il datagramma entra nello strato del **network.** Se anche l'indirizzo ip è corretto viene mandato allo **strato trasporto;** si controllano gli estremi delle porte e si indentifica il **socket**. Si identifica anche il buffer di ricezione e si ottiene il **messaggio.**

Livello Applicazione

Dobbiamo capire quali applicazioni utilizzano **quali servizi di comunicazione (**TCP / UDP**)**
In particolare, capire i benefici e difetti di entrambi. Vediamo nel dettaglio.

- **Trasmettere in TCP:** C'è una conversazione con l'interlocutore. Si apre un **doppio stream:** uno di invio e uno di ricezione. Lo stream è affidabile perché in caso di perdita di pacchetti c'è un meccanismo di rinvio; le perdite sono quindi escluse. È possibile che l'ordine dei datagrammi sia sbagliato e c'è tutta una logica dietro non mostrata direttamente al programmatore.

- **Trasmettere in UDP:** Molto più leggero e snello. **Non ci sono garanzie di arrivo del pacchetto e della sua integrità.** I pacchetti vanno mandato 1 alla volta.

Per quale motivo si utilizza UDP se ci sono tanti problemi?

In particolare*, quando non abbiamo paura di perdere pacchetti o quando ci sono pochi messaggi da scambiare.* Ad esempio, nelle conversazioni su teams anche se si perdono

dei pacchetti non importa perché la conversazione è in tempo reale e potremmo ripeterci. E' molto comodo anche perché si ha **pochissima latenza**.

TCP	***UDP***
email	*Internet telephone*
Web	*Real-time video conference*
Instant messaging	*Massive Parallel Computing*
P2P file sharing	*Massive multiplayer games*
Multi-user network games	*Cloud Computing*
Streaming stored video clips	*Cloud Storage*

Concetti di Quality of Service (QoS)

Le esigenze di un protocollo applicazione sono:

- **Affidabilità:** tranne quelle multimediali, non ci devono essere perdite di dati nelle applicazioni.
  espresso in **percentuale media di pacchetti persi**

- **Tempo di risposta / latenza:** il ritardo potrebbe essere un problema. espresso in **ms**

- **Banga / throughput:** ci sono applicazioni che sono esigenti e altre che hanno bisogno di molta banda. Espresso in **bit/sec**

- **Jitter:** ha impatto sulle altre tre; si esprime in **ms**

Quando misurare e Dove misurare

- **Dove**: bisogna scegliere gli **estremi della misurazione**. Da dove a dove. Ci possono essere dei fattori che aumentano la latenza in un link e in un altro invece no
- **Quando**: in base ai momento della giornata in cui si misura, ci possono essere risultati diversi.

Parametri importanti per speed test

- **RTT:** Round Trip Time – tempo di andata e ritorno di un pacchetto da un punto A ad un punto B.
- **Tempo di risposta / latenza ~** RTT / 2 → non è preciso fare diviso 2 ma è un arbitrio.
- **Latenza:** il tempo di andata da un punto A ad un punto B di un datagramma.
- **Banda:** quantità di dati che possono essere trasferiti in un dato periodo di tempo.
- **Throughput**: flusso massimo di **bit/sec** che si possono trasferire
- **Ritardo**: tempo di arrivo di un pacchetto dal punto A al punto B; uguale alla latenza

- **Congestione**: in base al throughput della rete e quante operazioni sono in atto, cambierà il throughput della rete; più pacchetti ci sono nel buffer, più si riempie.

Ritardo (latenza)

1) Elaborazione
2) Accodamento
3) Ritardo trasmissivo: dobbiamo tenere conto di:
    a. R: banda(bit/sec)
    b. L: lunghezza del pacchetto (bit)
    c. T: L/R (Tempo di trasmissione)
4) Ritardo di prorogazione
    a. D: banda(bit/sec)
    b. S:lungheza del pacchetto (bit)
    c. Latenza di prorogazione: d/s

Vediamo nel dettaglio

- **1)** ogni hop ha un tempo di elaborazione. Diciamo che qui possiamo prendere in considerazioni le fasi **di partenza e arrivo, di consultazione delle routing tables, dell'elaborazione del datagramma.**
- **2)** più datagrammi ci sono da elaborare, più aumenta la *coda*. Il **ritardo di accodamento** dipende da **quanto è lunga la coda,** dal ritardo di **elaborazione**, dalla **congestione** (altri pacchetti nel buffer che sono prima del nostro)
- **3) il ritardo trasmissivo** è il tempo di elaborazione a bordo di Alice + Tempo di accodamento di Alice + tempo di trasmissione.
- **4)** il ritardo di **prorogazione** dipende dal livello fisico. In base alla lontananza dei due **estremi della conversazione** e dal **mezzo trasmissivo usato.**

Dettagli aggiuntivi sul <u>Quality of Service</u>

Affidabilità: la probabilità di consegnare un pacchetto con successo. Complementare al *packet loss.*
Che impatti può avere il **packet loss:**

- **TCP**: In TCP quando un pacchetto viene perso viene mandato più volte fino ad una soglia. Questa cosa può aumentare notevolmente la **latenza**. Giustamente, se mandiamo lo stesso pacchetto più volte e non arriva, quello lo aspetta sempre. Influenza anche il **throughput** perché se calcoliamo un throughput di 1GBit e ne perdiamo il 50%, sommariamente abbiamo 500Mbit di throughput.

- **UDP**: il contrario. Non essendoci un meccanismo di ritrasmissione la **latenza è minima**. Il fatto è che il **throughput** viene in qualche modo influenzato visto che la perdita di pacchetti non è mai buona. Infatti viene visto come un **link con throughput inferiore ugualmente.**

Applicazioni	Affidabilità	Banda	Latenza
File transfer	No loss	Elastic	No
Email	No loss	Elastic	No
Web *documents*	No loss	Elastic	Yes
Real-time audio/video	Loss-tolerant	Audio: 5kbps -1Mbps Video: 10kbps – 5Mbps	Yes, 100msec
Stored audio/video	Loss-tolerant	Same as above	Yes,few secs
interactive games	Loss-tolerant	Few kbps up	Yes,100msec
Instant messaging	No loss	Elastic	Yes and no

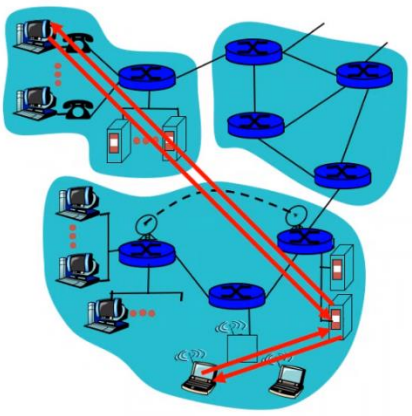
Jitter

Quando si comunica in una rete si mandami più pacchetti. Per calcolare **la latenza** di una famiglia di pacchetti devo fare il ritardo della famiglia di pacchetti. La latenza del gruppo viene calcolata come quella dell'ultimo elemento del gruppo arrivato.

- TCP**:** essendoci un ordine, se il primo pacchetto ci mette 300ms per arrivare, il secondo 50ms e il terzo 30ms, gli altri due sono bloccati dal primo e dovranno comunque aspettare che lui sia mandato (head of the line blocking)
  Jitter Alto: latenza alta
  Jitter Basso: tutti puntuali

- UDP: se c'è ordinamento, è uguale a TCP. Se non c'è ordinamento è diverso. Se arriva un pacco 2 prima di pacco 1, allora pacco 1 perso **perché non voglio trasmettere un informazione quando non mi serve più.**

Architettura Client – Server

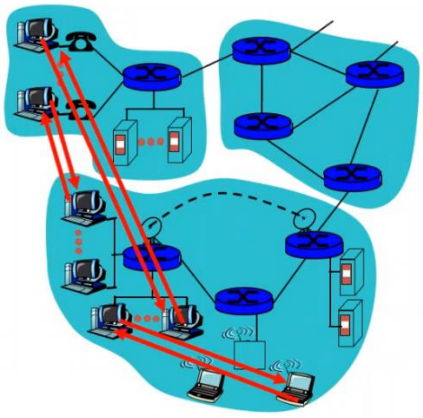


server:
- Sempre acceso
- Indirizzo IP (e porta) fissati
- Possibilità di avere "batterie di server"

clients:
- comunicano con il server
- si possono connettere a intermittenza
- Possono avere indirizzi IP dinamici
- Non comunicano tra loro direttamente

Architettura P2P Pura

- non serve necessariamente un server
- i sistemi comunicano direttamente tra loro
- i peer possono addirittura cambiare indirizzo IP
- Esempio: la rete Kad



Molto scalabile

Necessita di server al "bootstrap"

Tantissimi problemi tecnici da risolvere (non tutti ancora risolti al meglio)

I sistemi possono essere sia **client che server**. Ad esempio Skype è PEER 2 PEER. L'architettura p2p pura non esiste perché serve un server centrale.

Architetture Ibride

Napster / Kad + eDonkey

- Il trasferimento dei file era P2P
- La ricerca dei file era centralizzata, invece:
  - I peer registravano i nomi dei file su un server centrale
  - I peer interrogavano il server centrale

Messaggeria istantanea

- La chat avviene in modalità P2P o client-server
- Il file transfer avviene *preferibilmente* in P2P
- Il rilevamento della presenza on-line è centralizzato:
  - Quando un client va on-line viene registrato l'IP/porta attuale su un server centrale
  - Gli utenti contattano il server centrale per sapere chi è on-line e su che indirizzo/porta è reperibile.

Alcuni protocolli

Applicazione	Protocollo Applicazione	Protocollo di Trasporto usato
e-mail	SMTP [RfC 2821]	TCP
remote terminal access	SSH [RfC 4253]	TCP
Web	HTTP [RfC 2616]	TCP
file transfer	FTP [RfC 959]	TCP
streaming multimedia	RTP [RfC 1889] proprietary	TCP or UDP
Internet telephony	(es., Skype, Zoom…)	tipicamente UDP

Specifiche del livello applicazione

La comunicazione a livello applicazione avviene tra **processi**. Se i processi sono su diversi host devono usare un sistema a **messaggi.**

- **Processo client**: il processo che inizia la comunicazione
- **Processo server**: il processo che aspetta di essere contattato

Le applicazioni P2P hanno sia thread client che thread server

I protocolli applicazione definiscono:

- Il tipo di messaggi scambiati
- La sintassi: quali informazioni e in che formato
- La semantica dei messaggi
- Le regole su quando dei messaggi vanno scambiati

Protocolli di pubblico dominio: http, smtp

Protocolli proprietari: Kademlia (rete KAD)

Le regole di TCP (E IP)

- Ogni host possiede più interfacce di rete
- Ogni interfaccia è associata ad un diverso indirizzo ip (4 byte separati da puniti)
- Ogni interfaccia ha 65535 porte disponibili
- Non necessariamente un indirizzo ha un nome associato (www.mat.unical,it)

I socket

Ciò che permette al layer 7 e layer 4 di comunicare.

Il socket è simile ad un ingresso/uscita verso il mondo esterno

Ogni volta che una connessione TCP viene creata, il socket alloca una **coppia di buffer.**

- **Invio**: mettiamo i dati che vogliamo mandare
- **Ricezione**: ci sono i dati che riceviamo

Tipi di socket:

- **Listening socket**: identificato dall'estremità che ascolta – IP:PORTA
- **Socket legato ad una connessione attiva**: identificato da due coppie di coordinate:; IP1:PORTA1 <-> IP2:PORTA2

Come si programmano i socket

- **Il client deve contattare il server:** il programma server deve essere attivo e avere il socket collegato ad una porta e il client deve conoscere questo numero.
  Quando **il server viene contattato** crea una connessione per rispondere:
  - un server può parlare con più server sulla stessa porta
  - i client sono distinti tramite la coppia ip/porta

- ogni connessione ha un suo stato (connected,closed,…)

- **Come fa il client a connettersi al server:** crea un socket TCP e indica a che IP:PORTA connettersi. Il livello trasporto si occupa di stabilire la connessione

La posta elettronica

Attori principali nella posta elettronica:

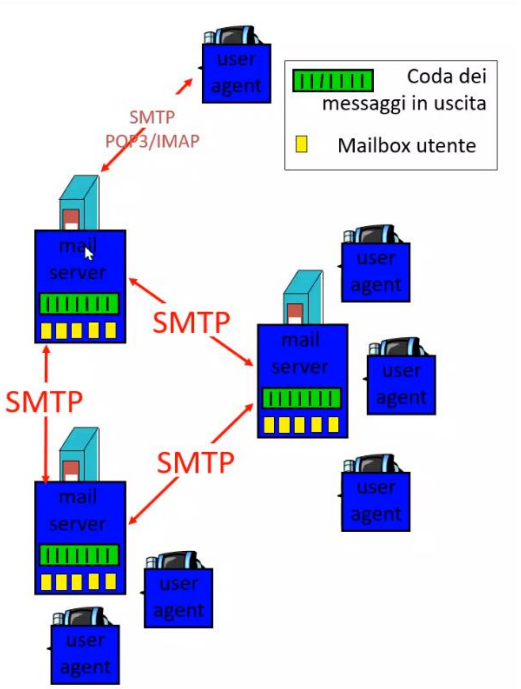
- Client di posta
- Server di posta
- SMTP: simple mail transfer protocol

Il protocollo STMP + un protocollo di **livello 7**

Protocollo di lettura: IMAP, POP3
La barra verde è la **coda dei messaggi in uscita.**
I quadrati gialli sono le inbox utente.
Gli user agent siamo noi (pc o smartphone)

Ciclo di vita di una mail:

1) **A** vuole mandare una mail a B.
2) **A** attraverso il suo user agent manda la mail a B.
3) **La mail** arriva al mail server di **A**
4) Il mail server di A apre una connessione con il mail server di B
5) Per trovare il mail server si utilizza un **server DNS per ricostruire l'indirizzo IP**
6) **Si apre** una connessone **TCP** tra i due mail server
7) **La mail** giace nel mail server di B;
8) **B** può utilizzare uno user agent per controllare la mail nella sua inbox



Formato mail:

- Intestazioni: **to, from, subject**
- Body: il messaggio in caratteri ASCII

Consultare la posta

Non si utilizza SMTP perché serve solamente per mandare la mail quel protocollo.

- POP: **POST OFFICE PROTOCOL**
- IMAP: **INTERNET MAIL ACCESS PROTOCOL**

Protocollo POP3

- **Fase di autorizzazione:**
  - User: nome utente
  - Pass: password
    Possiamo ricevere come risposte +**OK, -ERR** dal server
- **Fase di lettura**
  - List**:** elenca i numeri dei messaggi
  - Retr: recupera il messaggio in base al numero che gli diamo come parametro
  - Dele: cancella il messaggio in base al numero che gli diamo come parametro
  - Quit: indovina un po'

Queste operazioni potrebbero essere intercettate. Conviene usare uno strato crittografico

CHAP Authentication (challenge – response)

Si utilizza un meccanismo di challenge che va risolto attraverso un problema matematico.

Si può inserire la password in base64, ma attenzione che **non è uno formato crittografico.**

Web e HTTP

Una pagina web consiste in un file HTML che fa riferimento a diversi oggetti al suo interno. Ogni oggetto è riferito tramite un **URL**

L'URL è formato da:

Protocollo: **http**

- Host name: **www.google.com –** stringa valida che dev'essere capita dal DNS
- Path name: **/src/img.png –** nome del percorso

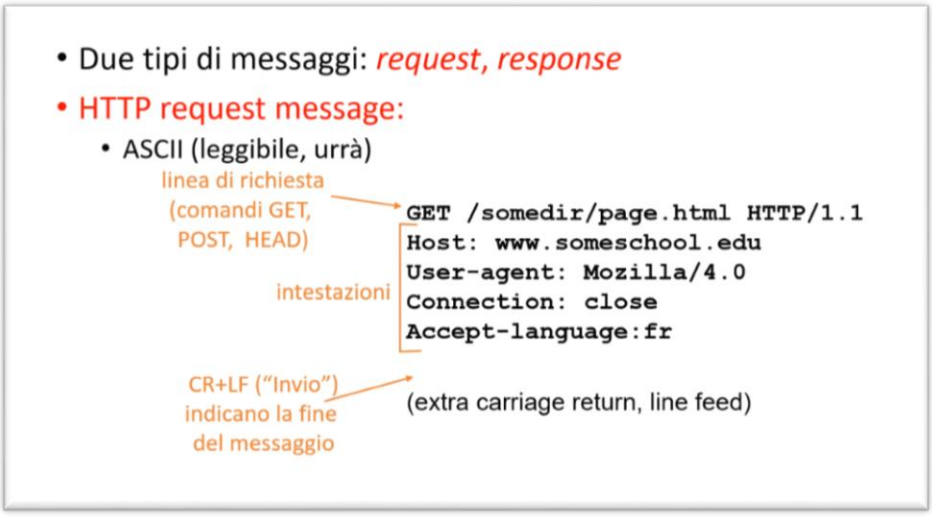
Come funziona http

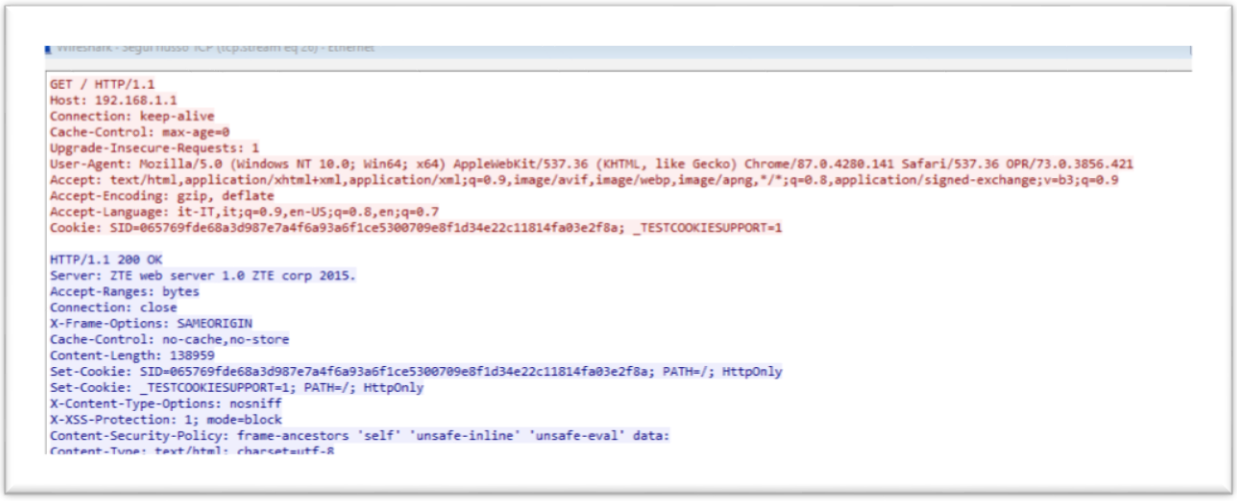
- Client:
  - un programma richiede risorse
- Server:
  - Il server accetta connessioni
  - un web server fornisce le risorse richieste
- Si usa **tcp**
- i due interlocutori scambiano messaggi in http

Connessione non persistente:

- client si connette
- chiede la risorsa
- riceve la risposta del server
- il client si disconnette

Molto lenta come cosa: in particolare dobbiamo pensare al **RTT** che c'è. Perché c'è il primo **RTT** che bisogna attendere per **aprire la connessione col server** e il secondo **RTT** che dobbiamo attendere quando richiediamo la risorsa e aspettiamo che ci arrivi; a pensiamo al tempo totale di trasmissione.





```
GET / HTTP/1.1
Host: 192.168.1.1
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36 OPR/73.0.3856.421
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: SID=065769fde68a3d987e7a4f6a93a6f1ce5300709e8f1d34e22c11814fa03e2f8a; _TESTCOOKIESUPPORT=1

HTTP/1.1 200 OK
Server: ZTE web server 1.0 ZTE corp 2015.
Accept-Ranges: bytes
Connection: close
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache,no-store
Content-Length: 138959
Set-Cookie: SID=065769fde68a3d987e7a4f6a93a6f1ce5300709e8f1d34e22c11814fa03e2f8a; PATH=/; HttpOnly
Set-Cookie: _TESTCOOKIESUPPORT=1; PATH=/; HttpOnly
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Security-Policy: frame-ancestors 'self' 'unsafe-inline' 'unsafe-eval' data:
Content-Type: text/html; charset=utf-8
```

- rosso: client - richiesta
- blu: server- risposta

Che cos'è una sessione? – un gruppo di connessioni TCP che sono legate dallo stesso accesso. Per distinguere le sessioni ognuna ha un identificativo diverso.
Viene **implementata attraverso i Cookie**

Differenze connessioni **TCP** e **IP:**

- **TCP**: ciò che viene scambiato al livello 4
- **IP**: ciò che viene scambiato a livello 3

Intestazioni rilevanti:

- **User agent:**
  - Dichiarato dal brower per idnicare il broware
  - Interpretato dai server per decidere quali contenuti mostrare

- **Referrer:**
  - Usato dai client per indicare l'url utilizzato per arrivare all'URL corrente

HTTPS e sicureza

Cosa significa la sigla HTTPS? – **http** over **TSL**

TLS: transport layer security

- Garantisce l'identità dei due interlocutori
- La confidenzialità
- L'integrità dei messaggi

Cookie

Quattro componenti:

- Campo cookie nella risposta
- Campo cookie nella richiesta
- il browser salva i cookie nella risposta e li reinvia quando viene richieste lo stesso oggetto
- il sito web contiene un db con i cookie

Funzionamento:

- il client fa una richiesta al server
- il server genera un ID e lo associa al client e lo salva nel database
- quando il client farà una richiesta al server, la risposta verrà personalizzata in base al **COOKIE** e rimarrà cosi fino a quando il cookie non cambierà per volontà del client

Web Caches (proxy servers)

Obiettivo: non generare traffico se si ripete la stessa richiesta

- l'accesso al web è fatto tramite un cache server (proxy)
- se la richiesta è in cache viene ritornato al client
- se invece non è presente si fa la richiesta al server e viene successivamente tornato

riduce il tempo di risposta e riduce il traffico

FTP – File transfer protocol

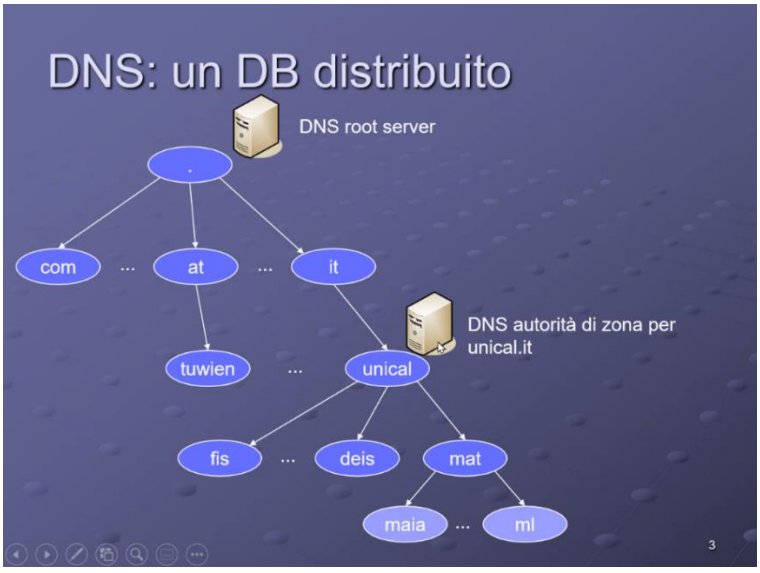
Trasferire file da e per un host remoto; **porta 21**

- il client contatta il server sulla porta 21 con TCP
- si possono navigare le directory sulla connessione
- una connessione dati separata viene aperta per mandare i file
- dopo che i file sono trasferiti si chiude la connessione

Naming e DNS

Un **DNS resolver** trova gli indirizzi IP partendo dal nome. Consulta altri DNS per trovare l'indirizzo. Se lo trova allora la conversazione può iniziare

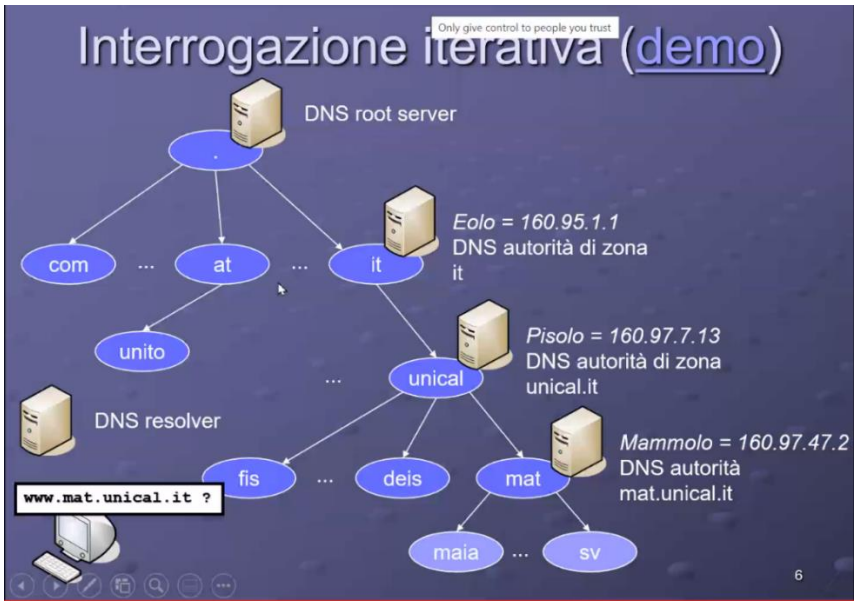
Cosa c'è dietro il resolver?



I server root hanno autorità di tutto. All'interno ci sono dei database che contengono file di testo, chiamati **file di zona.**
I file di zona all'interno contengono triple del tipo <nome, tipo, valore>

- **maia.mat.unical.it. A 160.97.47.242**
- **mat.unical.it. MX ml.mat.unical.it**
- **160.97.63.242 PTR maia.mat.unical.it**

Interrogazione iterativa



Il resolver deve scorrere quasi tutti i server di zona, ecco perché iterativa. La quantità di dialogo è enorme ovviamente. Consulta il DNS e trova l'informazione di volta in volta nei vari server fino ad arrivare a quello richiesto.
Guarda l'immagine per capire dove si trova. Mammolo ha **mat.unical.it**, quindi dal root fino a li deve arrivare

Interrogazione ricorsiva: il carico di lavoro si server è molto più alto, ma è più efficiente. Viene controllato di volta in volta se è presente l'indirizzo del server richiesto e, se non c'è, viene mandato al server inferiore. Quando viene trovato, risalendo a cascata fino al root si ritorna la risposta

Cache Poisoning: se un server è infetto potrebbe dirottare il traffico richiesto per un indirizzo da un'altra parte. Questo perché nei DNS **non c'è garanzia di base sulla confidenzialità e autenticità delle query responses.**

Consulatre dns da cmd: nslookup

Livello trasporto – Layer 4

Quello che vediamo all'interno del layer 4 **viene gestito dai sistemi operativi.**

- È lo strato di interfaccia con il **livello 3**
- Inaffidabile:
  - I pacchetti possono morire per 3 motivi: **congestione della rete, congestione delle destinazione e corruzione dei pacchetti**
  - L'arrivo dei pacchetti non è sempre nello stesso ordine.

Come si utilizza il mezzo trasmissivo?

- Send(ip1, ip2, data[]): un array di byte da mettere sul filo
- Data[] Receive()

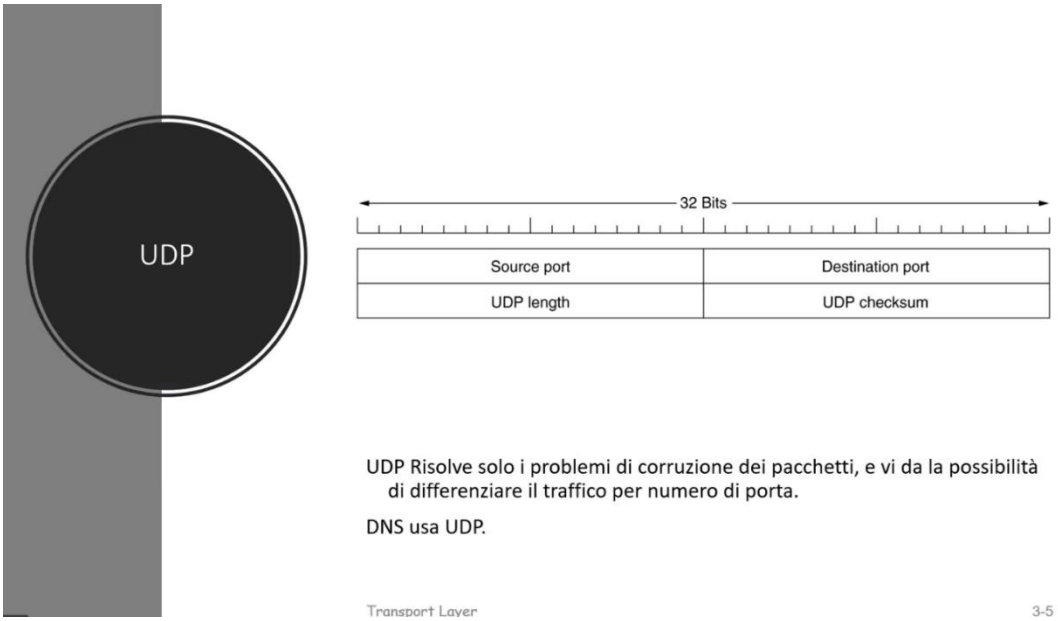
- Connect(), Disconnect()

I protocolli di trasporto sono due

- Dns usa UDP
- Risolve il problema della corruzione dei pacchetti, ma non gli altri due.

In particolare, il **livello applicazione** fornisce *ipdestinatario:porta e ipsorgente:porta*. Viene messo in un array di byte e mandato tramite il layer di trasporto

Non c'è distinzione tra **socket listening e socket connesso.** Infatti in UDP c'è solo il **listening**. Quando fai receive hai garanzia che ti entri un datagramma nella dalla porta di ascolto selezionata.
Quindi, **in UDP può arrivare roba da chiunque,** cosa che non avviene in TCP



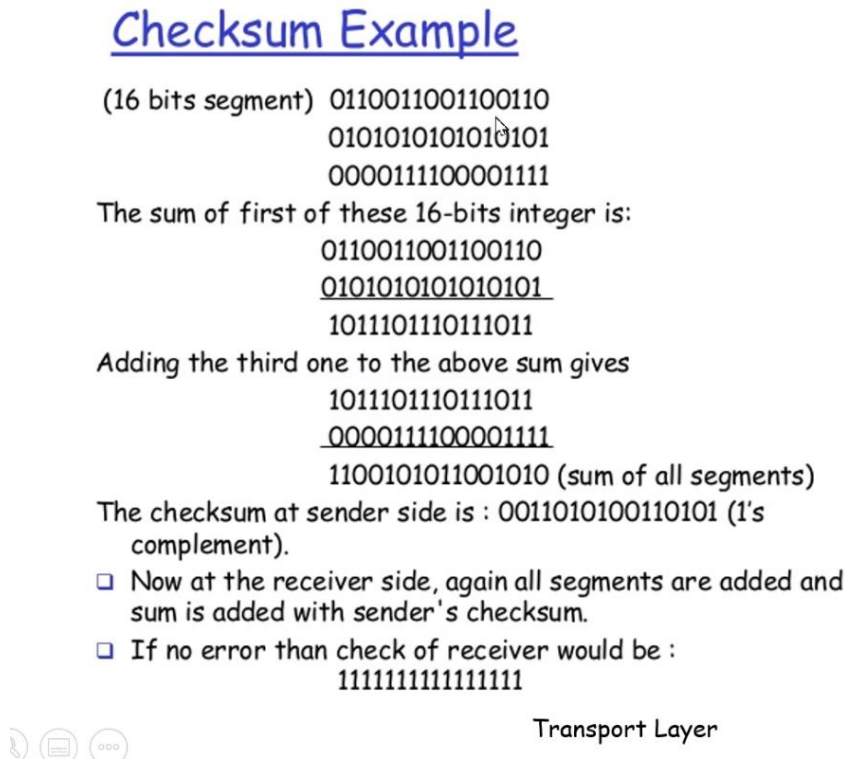
In UDP lo smistamento è basato su **destination port**.

A cosa serve la **checksum**? – la **checksum** è calcolata all'interno di un algoritmo; serve per capire se **il contenuto della stringa sia stata alterata e ne controlla l'autenticità;** il risultato della checksum si basa sul contenuto della stringa (simile al codice fiscale e la lettera di controllo finale)

La checksum è usata quando i **dati sono corrotti**. Ci possono essere casi dove i dati corrotti non influenzano il funzionamento del servizio. Nel caso dello *streaming* meglio avere **dati corrotti piuttosto che il totale blocco del servizio.**

La checksum viene calcolata dal sistema operativo; visto che in UDP c'è la possibilità che il pacchetto si corrompa, se il contenuto è alterato e la checksum è diversa **il sistema operativo scarta automaticamente il pacchetto perché corrotto.**

UDP checksum



1) Si somma la prima riga con la seconda
2) Si somma la seconda con la terza
3) Si prende il risultato e si fa la **not (il complementare)**
4) Si manda il tutto
5) Il ricevente fa la somma di tutto quello che arriva e tiene la somma da parte.
6) Si fa la somma tra la somma messa da parte e la checksum ricevuta.
7) Se questo la somma tra questi due fa tutto 11111111 allora è **corretto**.

Se il datagramma si corrompe in modo tale che **la checksum risulti uguale** siamo fregati. **Attenzione:** la checksum è **inutile** se la **manomissione del pacchetto è volontaria;** sicuramente qualcuno che manomette il pacchetto lo manometterebbe in modo tale da avere la stessa checksum oppure modificherebbe direttamente la checksum.

Importante **ECC: Error Correction Code**

- Vengono usati numeri abbondanti e non 16bit.
- Puoi trovare la corruzione e correggere l'errore

Se invece di usare la checksum si utilizaz **ECC** in base ai valori calcolati lato A e lato B si può trovare l'errore. Attenzione perché se si esagera nella riparazione potrebbe esserci ridondanza eccessiva.

TCP

Più sicura e affidabile di UDP. Ci sono alcuni problemi da risolvere; bisogna stare attenti a:

- Corretta numerazione del tutto
- Corretto ordine di consegna

Quando un pacchetto viene consegnato in TCP, c'è una risposta chiamata **ACK.** Quando la prima volta questo pacchetto non viene consegnato si aspetta un **timer di timeout** di circa 5 secondi e le prossime volte è più corto. Se allo scadere la risposta **ack** è arrivata al mittente **il pacchetto è ricevuto,** altrimenti si manda ancora.

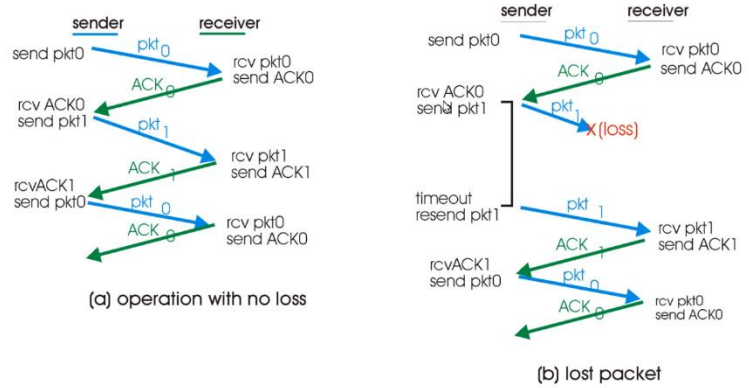
Risposta persa: quando si perde la risposta di ritorno, si potrebbero avere dei duplicati perché il mittente potrebbe mandare ancora una volta il pacchetto. Quando questo succede, **la ricevuta di ritorno va ripetuta** ma il mittente non saprà che il pacchetto duplicato è scartato.

Come faccio a capire che il pacchetto è già ricevuto? *Pacchetto per pacchetto o numerare i singoli byte.* Con questo numero posso capire se il pacchetto è già stato ricevuto o no; per comodità anche gli **ack** avranno un numero.

Implementazione Stop & Wait:

- **A** manda un pacchetto
- **B** riceve il pacco
- **B** manda la risposta
- **A** riceve la risposta

Stop & Wait in azione

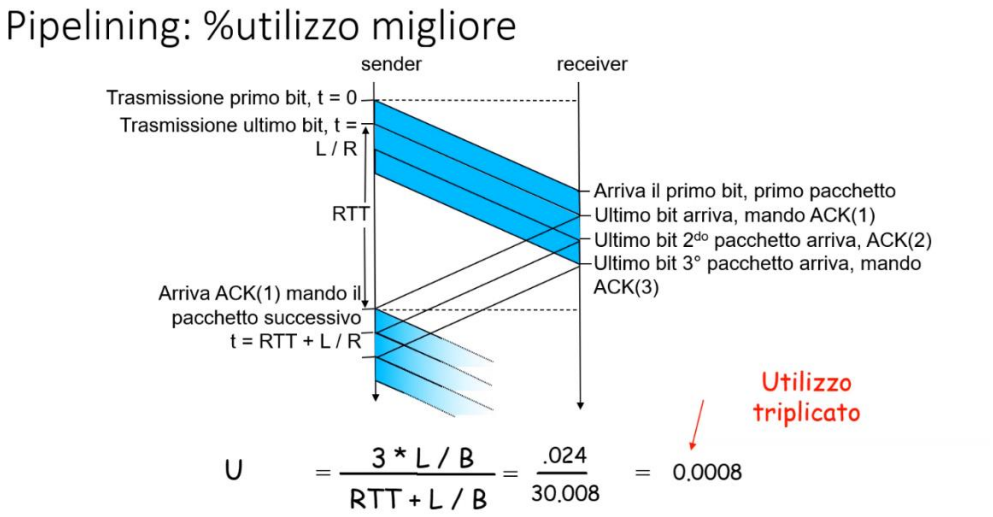


Problemi:

- Non è efficiente a smaltire i dati in partenza
- I pacchetti vanno mandati uno alla volta

TCP assomiglia più ad un protocollo a **finestra scorrevole** (**pipeline**): *più pacchetti in volo ancora da essere confermati*

- Go back N
- Selective repeat

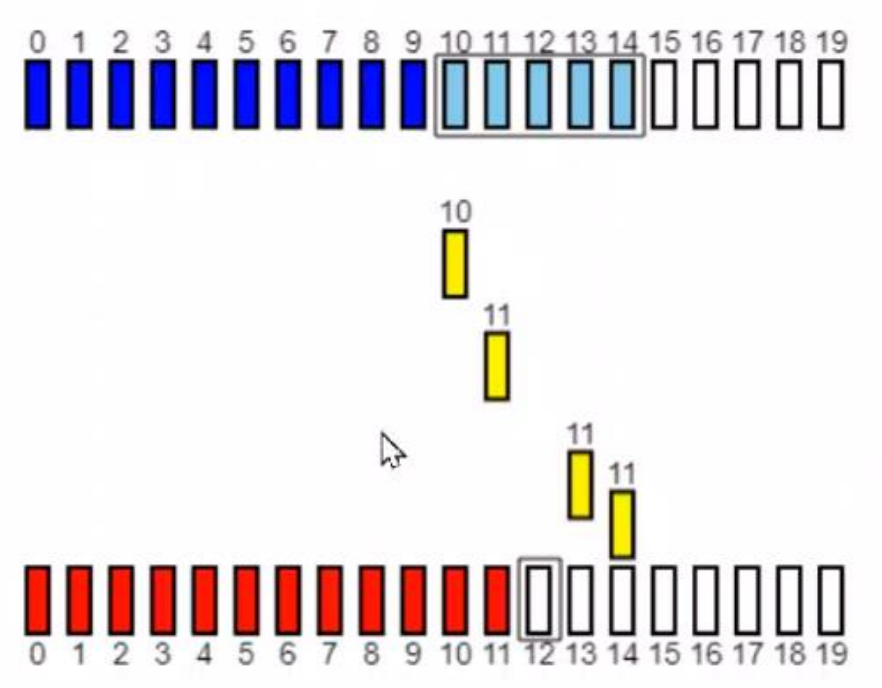


Cose positive:

- I pacchetti mandati sono di più
- Più efficiente
- Il tempo morto è molto di meno
- Più trasmissioni

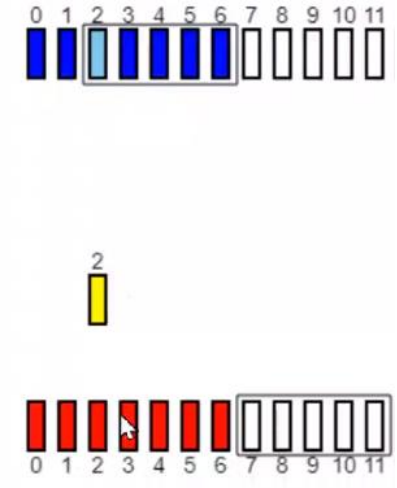
Go Back N

Funziona cosi: *chi riceve i pacchetti, conferma ogni qual volta ne riceve uno; quando non vengono ricevuti alcuni, si ferma e non riceve altri e* ***chiede la riconferma*** *di quelli non ricevuti. Quando li riceve, al mittente basta ricevere un pacchetto per confermare i precedenti, visto che se è arrivata la conferma ad esempio del pacchetto 4 allora anche 1 2 3 sono per forza arrivati, altrimenti lo avrebbe saputo dal ricevente.*

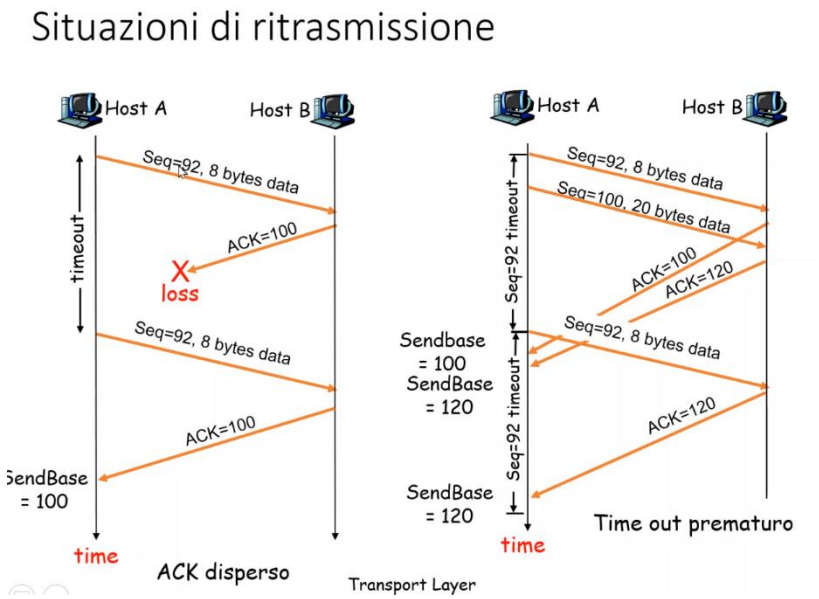


Selective Repeat

I pacchetti ricevuti vengono bufferizzati. Ci sono delle finestre di invio e ricezione sia per mittente che destinatario. Succede che, quando un pacchetto non viene ricevuto, viene informato il mittente e quindi bisogna attendere la corretta spedizione di quel pacchetto per andare avanti e spedirne altri. Anche se il precedente non è ricevuto, quelli successivi vengono bufferizzati. Allora, quando tutti sono ricevuti, si sposta la finestra di invio e ricezione avanti per spedire gli ulteriori pacchetti.



Go back n tcp

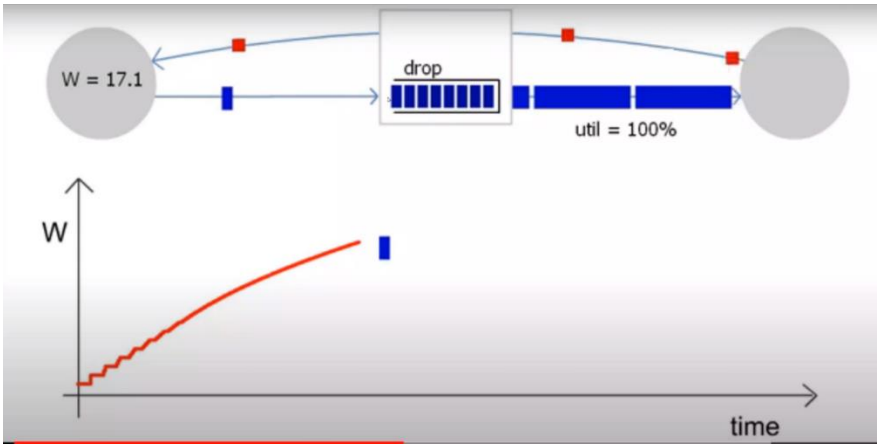


Cosa sono i numeri di sequenza e di ack?

- **N sequenza:** oofset del primo byte dnei dati
- **N ack:** numero del prossimo byte da ricevere

Controllo di flusso: Se A manda troppi dati a B e B non riesce a gestirli in tempo, potrebbe esserci cogestione della rete e alcuni pacchetti potrebbero essere droppati. Viene

in aiuto il **controllo di flusso** di TCP. In partica, il **buffer di invio di A** viene ridimensionato in base al **buffer di ricezione di B**.



Altra soluzione: **controllo di congestione –** Si parla inizialmente con finestre d'invio piccole e man mano di aumenta il throughput; quando si verifica una perdita di diminuisce il throughput e si ripete dall'inizio il processo.

2 implementazioni: **Reno e Tahoe.**

In tcp si usano sia il controllo di congestione che il controllo del flusso.

Problemino: ogni perdita per il controllo di congestione viene interpretato come un problema di congestione; ma se la perdita fosse dovuta alla rete in generale? Allora si avranno pessime performances su link con packet loss da disturbo (vedi wifi)

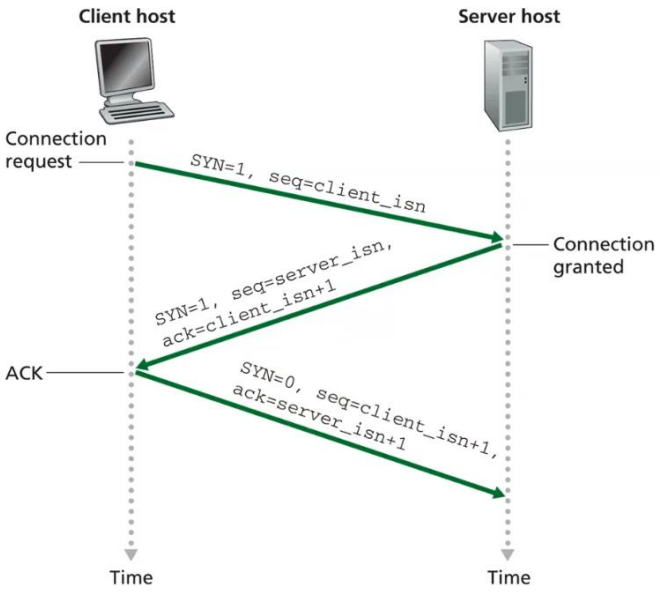
Come si sceglie il tempo di timeout in tcp

$$\texttt{EstimatedRTT} = (1 - \alpha)*\texttt{EstimatedRTT} + \alpha*\texttt{SampleRTT}$$

Alpha = 0.125

Handshake

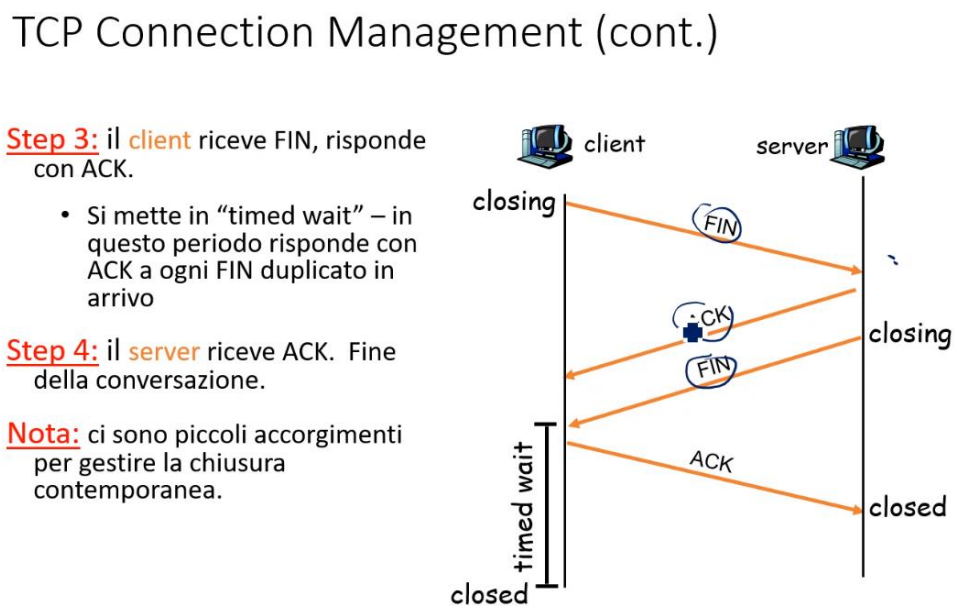
È l'inizio di una **connessione TCP.**



- Il client chiede al server di aprire una connessione a partire da un certo numero di sequenza
- Il server dice al client che è pronto ad aprire una connessione con quel numero di sequenza
- Viene mandato un secondo ack dal client al server per dire che ha ricevuto la risposta e confermare che lui sappia i numero di sequenza

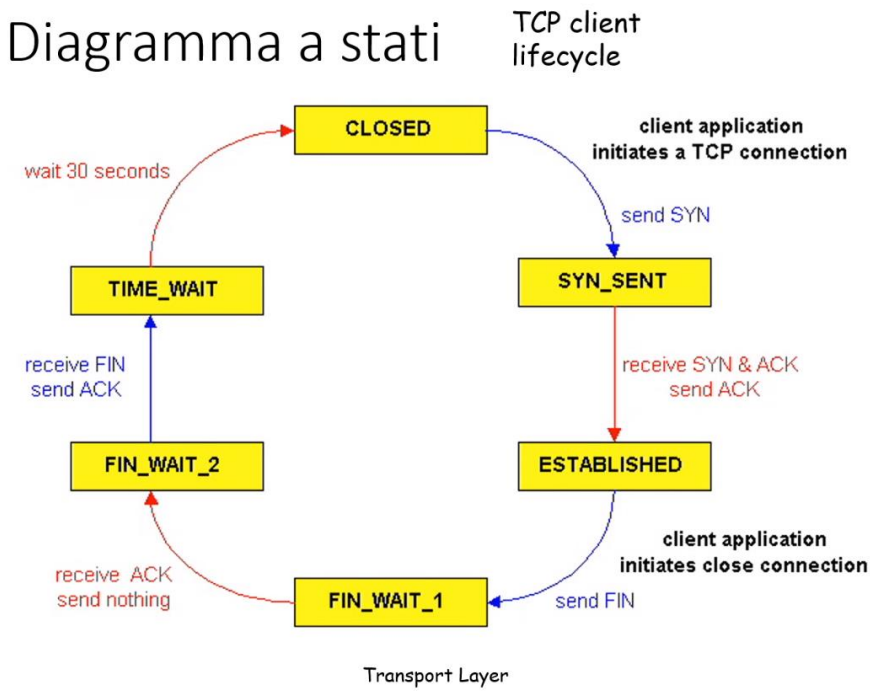
Il numero di sequenza iniziale non parte mai da 0 per evitare attacchi man in the middle; sapendo che i pacchetti iniziato da 0 potrebbero fare una tcp injection. **Il numero di sequenza vero è quello raw.** Il sequence si **randomizza**

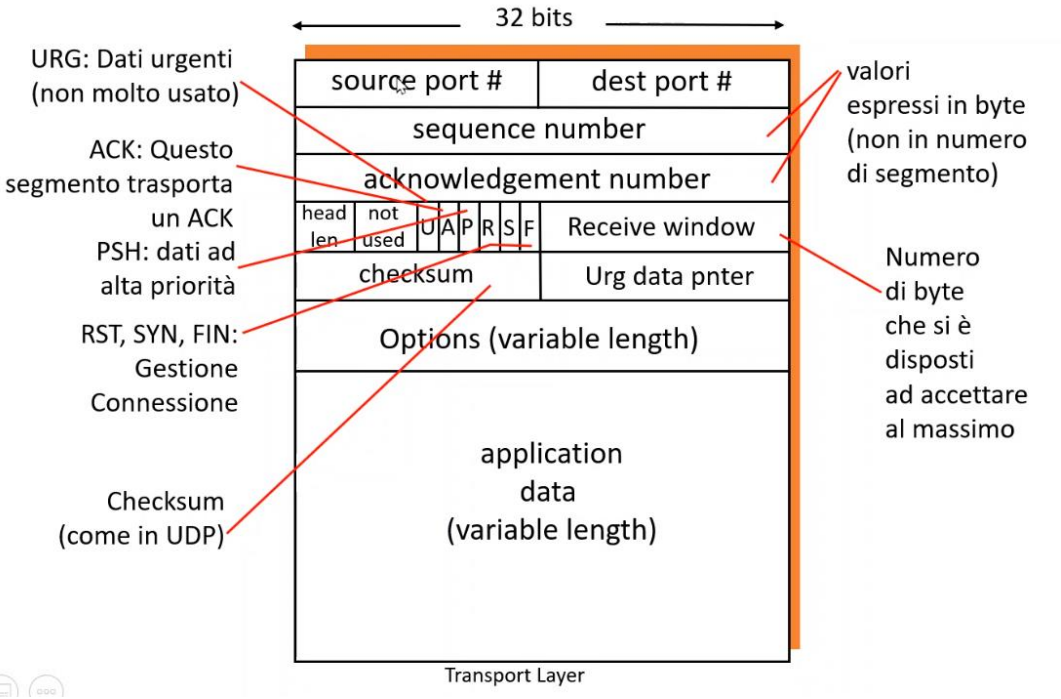
Fine conversazione TCP:



- Il client manda il FIN(fine conversazione)
- Il server manda l'ack e manda anche lui il fin
- Ora il server aspetterà anche lui l'ack finale del client.

Qui è stato deciso di mettere un timeout finale per il server altrimenti potrebbe rimanere per sempre in attesa dell'ack del client





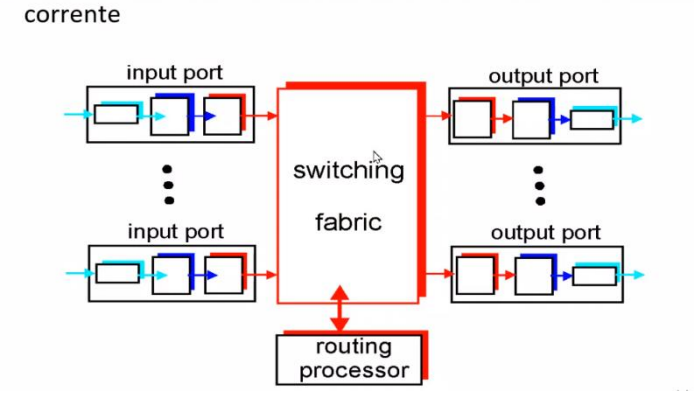
Livello Network – livello 3

Ci troviamo sullo strato 3.

- Trasporta i datagrammi da A a B

Funzioni di un router

- Fa girare algoritmi di routing (RIP, OSPF, BGP)
- *Inoltra* i datagrammi in base alla sua tabella di routing corrente



- I segmenti TCP vengono spezzettati se serve in datagrammi IP
- Dall'altro lato vengono ricostruiti
- I nodi intermedi intervengono solo a livello rete
- I router guardano dentro le intestazioni e decidono cosa fare

Importante: ora ci sono gli end point e i router intermedi che entrano come attori. All'interno dei **routers** ci sono **control plane e data plane**

- Uno fa le cose primarie
- L'altro fa le cose secondarie

Ci sono anche delle **routing tables** che vengono consultate per l'instradamento.

- **Data plane(switching fabric):** controlla dento le routing tables e instrada il datagramma con meno latenza possibile
  Spesso implementato in hardware
- **Control plane (routing processor):** avvengono calcoli e rielaborazioni. I calcoli servono a programmare il data plane per fare le cose correttamente

Come avviene il forwarding?

Serve conoscere **IP sorgent**e e **IP destinazione**. Il protocollo è importante inserirlo, **TCP ha codice 6.**

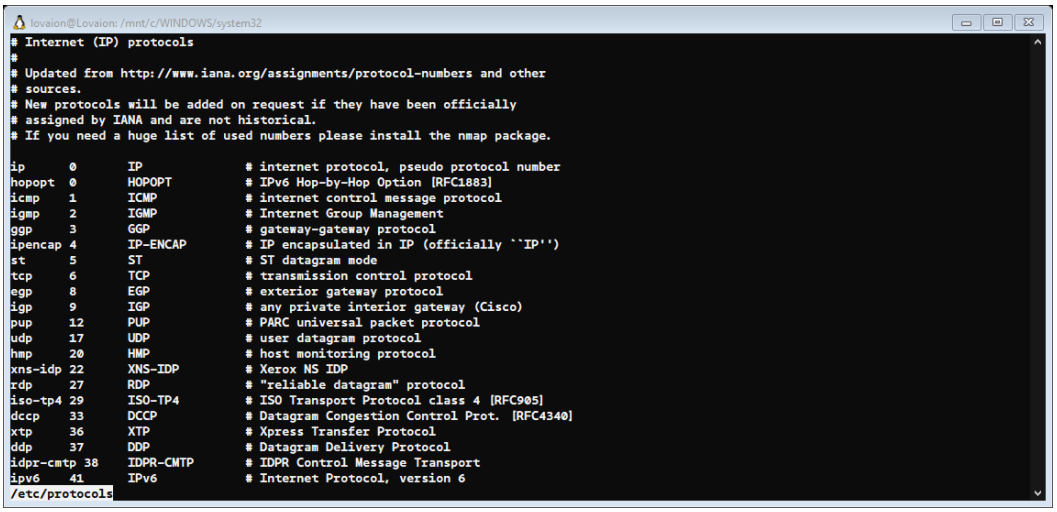


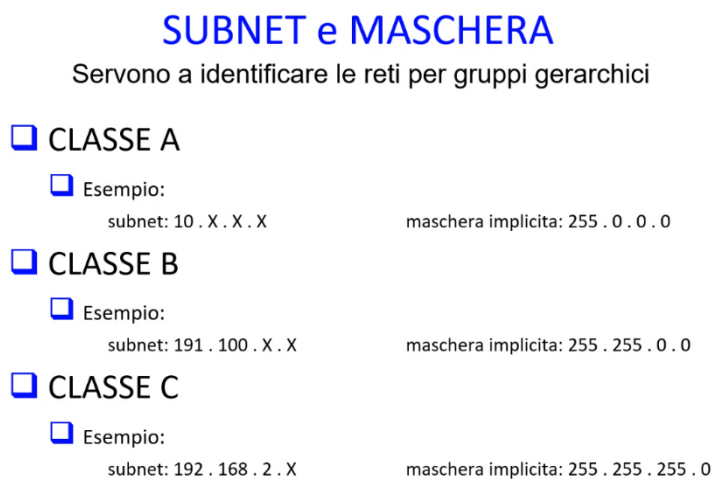
Figura 2: less /etc/protocols

Per consultare la tabella di routing:

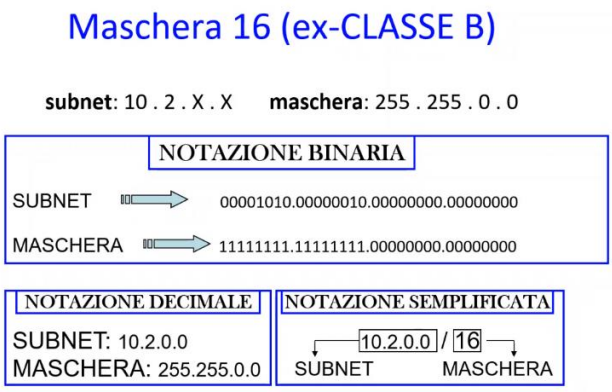
- print route Window
- route -n linux

Capire ora subnet mask e gateway

Le **subnet** sono gruppi di indirizzi. In base alla subnet si può capire gli indirizzi che ne fanno parte. In particolare servono a **identificare le reti per gruppi gerarchici.** La maschera indica **quali byte sono fissi e quali possono variare.**



- classe A: il primo byte è fisso, gli altri variano
- classe B: i primi 2 byte sono fissi, gli altri variano
- classe C: i primi 3 byte sono fissi, gli altri variano



Le reti non sono sempre cosi, perché ci possono essere anche 23 27 28 non per forza **8 16 24**

Indirizzamento:

Gli ISP (Internet Service Provider) compra roba.

Come avviene l'assegnazione dell'indirizzo IP ad ogni singola stazione?

ogni stazione deve avere dei parametri:

- **subnet mask**: definisce la dimensione della sottorete IP . 255.255.255.0
- **gateway predefinito:** è l'indirizzo di un router o dispositivo di routing che collega una rete locale a internet

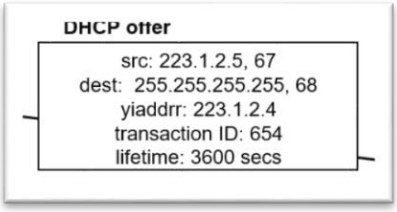
Gli **indirizzi broadcast** sono riservati a reti locali**, non si possono usare pubblicamente.**

Configurazione statica e dinamica dell'interfaccia di rete:

- statica: legge da file e imposta
- dinamica: fa richiesta ad un server DHCP e gli viene fornito un indirizzo IP valido.

Spiegazione configurazione dinamica

- il server DHCP dev'essere raggiungibile in broadcast
- il client produce un datagramma broadcast; quello che esce è un DHCP discover
- il server DHCP risponde con **l'offerta**



- il client genera un'offerta ufficiale richiedendo quella ricevuta
- il DHCP gli da l'ip

NAT

Gli indirizzi IP sono finiti, quindi non è possibile dare un indirizzo IP privato a tutti.

Cosa fa il NAT? – è un trucco per fare in modo che si utilizzi un IP pubblico per parlare in rete utilizzando però il proprio indirizzo privato ed essendo riconoscibile sulla rete.

Come avviene?

Il router domestico ha 2 interfacce: **domestica** e **pubblica**

Inizio:

- quando la conversazione inizia, il router riceve un pacchetto da parte di un host nella rete
- cambia le informazioni del pacchetto, in particolare IP sorgente e anche la porta alcune volte
- quando la riscrittura avviene, si aggiorna anche una tabella chiamata **nat translation table,** dove si tiene a mente chi ha mandato il pacchetto e su quale porta
- il server che voleva essere contattato dall'host della rete riceve il datagramma vedendo come mittente il router domestico, e la risposta viene mandata a lui
- il datagramma viene controllato, si analizza la nat translation table e viene rimandato il pacchetto al mittente originale.
- La gestione delle porte è fondamentale per non avere problemi di incongruenze

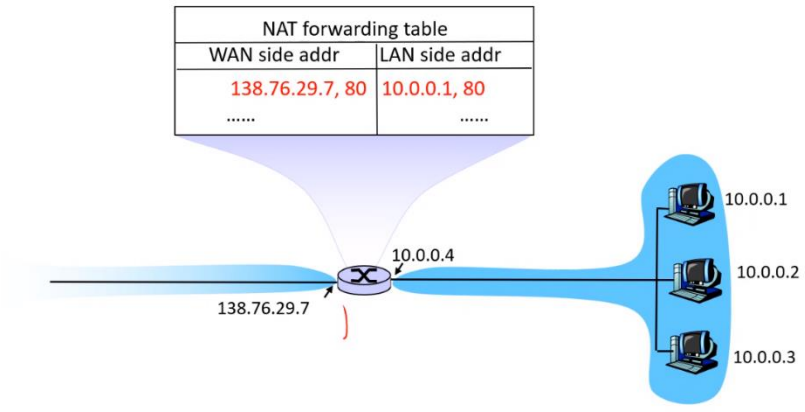
Problemi col nat

- 65534 porte utilizzabili simultaneamente al massimo

Se voglio un servizio raggiungibile dall'esterno?

Nel senso, se io 10.0.0.3 voglio che un servizio sia raggiungibile dall'esterno, come faccio a farmi raggiungere?

Entrano in gioco le **port forwarding table**, dove vengono mappate le porta in listening di un qualche host che vuole essere raggiungibile dall'esterno.



Cosi se arriva una conversazione sulla porta 80 per l'indirizzo pubblico del router, la conversazione viene forwardata al 10.0.0.1

Problemi NAT e P2P

A è dietro un NAT mentre **B** no. Quando B chiama A, Bob fa capire ad A che la sta chiamando, e quindi, visto che c'è un NAT, A chiama B sotto il cofano, senza essere visto. Questa cosa si chiama **callback**.

ICMP protocollo di controllo

È senza numero di porta; i pacchetti vengono intercettati e processati prima di essere smistati. Echo request, echo reply, destination unrechable.

Loop per colpa delle tabelle di routing

Se la tabella di routing è configurata male, ci potrebbero essere dei loop per i pacchetti. Si utilizza il **TIME TO LIVE (TTL)**; Parte con un valore grande e diminuisce ogni volta che viene palleggiato. Quando questo TTL arriva 0, si ferma.

Firewall

Triangolo della sicurezza:

- **Confidenzialità**: i dati non devono essere visualizzabili da fuori e da altri
- **Integrità**: i dati devono rimanete integri e non manomessi
- **Disponibilità**: manomettere la disponibilità di un servizio

Firewalling: porte taglie fuoco; ci sono differenze tra **router** e **firewall**

- **Router:** decide dove instradare le conversazioni in input
- **Firewall:** in base a delle regole al suo interno decide se il pacchetto può essere accettato o no

Gli scopi del firewall sono:

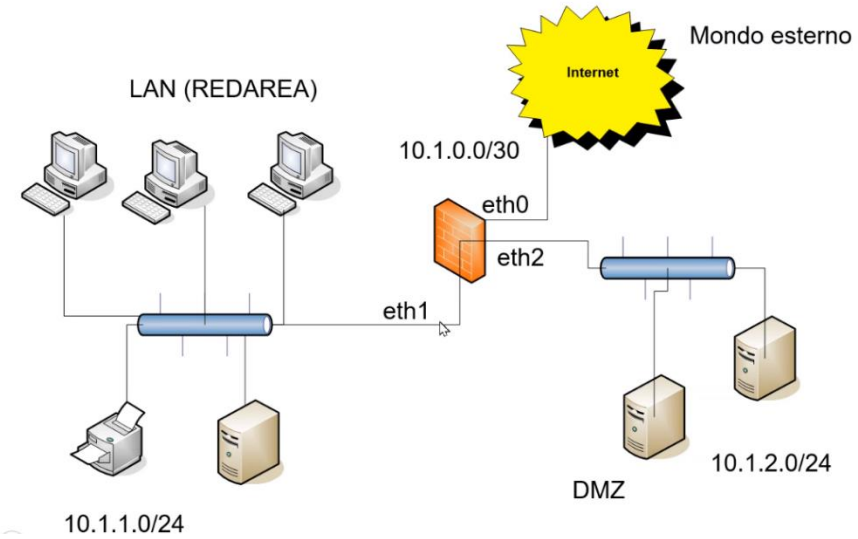
- I servizi che devono essere visibili all'esterno devono essere visti
- Deve impedire agli utenti della rete di usare alcune applicazioni
- Proteggere da attacchi DOS e DDSO
- Evitare che server o client interni danneggino danni a terzi
- Minimizzare i danni se qualche macchina è infetta

Termini tecnici

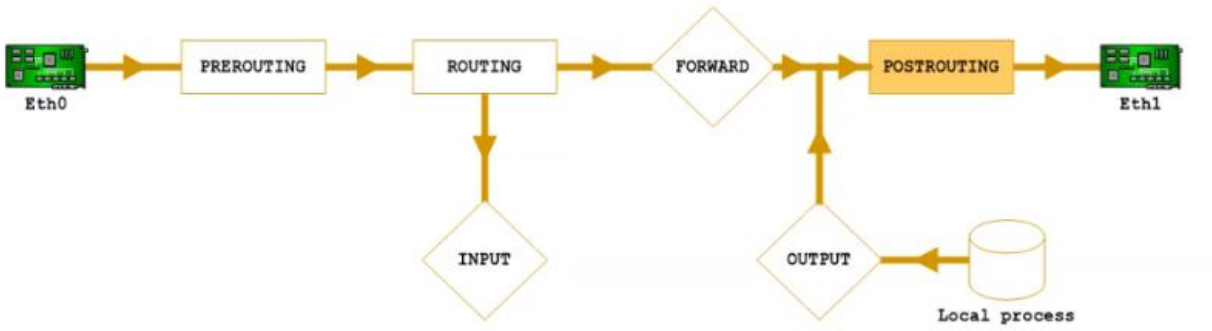
- Personal firewall: software per filtrare il traffico
- Firewall perimetrali: firewall *a monte* della rete che appartiene ad un dominio di collisoine
  layer 2 3 e 4

Tipi di firewall

- **Stateless:** filtrano pacchetto per pacchetto
- **Stateful:** usano il traffico precedente e lo tengono in memoria per decidere il destino di un pacchetto



Configurazione del firewall



Bisogna capire **IPTABLES**

- **Code**
  - **Ogni coda** ha un insieme di regole per capire quali pacchetti vanno avanti o meno
- **Prerouting:** modifica indirizzi destinazione
- **Postrouting:** riscrittura indirizzi sorgente

Catene: lista di regole (i rombi)

- **Input**: datagrammi destinati al firewall
- **Output**: datagrammi in uscita dal firewall
- **Forwarding**: datagrammi che passano per il firewall, ma sono destinati ad altri

Definizione delle policy: le policy sono quelle regole che vengono chiamate in causa quando non c'è una regola per uno specifico datagramma; quindi, vengono controllate quelle. E' bene mettere le policy in drop inizialmente e gestire poi i datagrammi con le varie catene.

- ***Iptables -p INPUT DROP***
- ***Iptables -p OUTPUT DROP***
- ***Iptables -p FOWARD DROP***

Screen creazione catene:

Come creare una nuova catena:

```
iptables –N nomecatena
```

```
iptables -N landmz
#dalla scheda di rete eth0 alla scheda di rete eth1
iptables -N laninet
#dalla scheda di rete eth0 alla scheda di rete eth2
iptables -N dmzinet
#dalla scheda di rete eth1 alla scheda di rete eth2
iptables -N dmzlan
#dalla scheda di rete eth1 alla scheda di rete eth0
iptables -N inetdmz
#dalla scheda di rete eth2 alla scheda di rete eth1
iptables -N inetlan
#dalla scheda di rete eth2 alla scheda di rete eth0
```

Aggiungere regole:

Aggiungere una nuova regola

```
iptables –A nomecatena condizioni –j decisione
```

decisione=un nome di catena, oppure DROP, REJECT, ACCEPT

```
iptables -A FORWARD -i eth1 -o eth2 -j landmz
iptables -A FORWARD -i eth1 -o eth0 -j laninet
iptables -A FORWARD -i eth2 -o eth0 -j dmzinet
iptables -A FORWARD -i eth2 -o eth1 -j dmzlan
iptables -A FORWARD -i eth0 -o eth2 -j inetdmz
iptables -A FORWARD -i eth0 -o eth1 -j inetlan
```

Configurazione finale:

Dalla LAN ad Internet

```
iptables -A laninet ! -s 10.1.1.0/24 -j DROP
iptables -A laninet -p tcp --dport ftp -j ACCEPT
iptables -A laninet -p tcp --dport http -j ACCEPT
iptables -A laninet -p tcp --dport https -j ACCEPT
iptables -A laninet -m state ESTABLISHED,RELATED -j ACCEPT
iptables -A laninet -p tcp -j REJECT --reject-with tcp-reset
```

Da Internet alla LAN

```
iptables -A inetlan -d ! 10.1.1.0/24 -j DROP
iptables -A inetlan -s  10.1.2.0/24 -j DROP
iptables -A inetlan -s  10.1.1.0/24 -j DROP
iptables -A inetlan -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A inetlan -p tcp -j REJECT --reject-with tcp-reset
```

Quando si mette **reject** il firewall da una risposta al mittente (**connection refused ad esempio**)

Modulo state: è un modulo di iptables. Si usa nella creazione di regole di firewalling in base a come il Kernel li ha marcati.
Esempio: *established, related* quando una connessione è già cominciata e sta tornando una risposta.

Un firewall statefull accetta o droppa in base allo storico, come detto sopra qui.

I pacchetti possono essere **marchiati** dal kernel:

- **NEW:** il primo di una nuova conversazione
- **ESTABLISHED:** consultando la tabella delle conversazioni esiste nella conversazione
- **RELATED:** pacchetto *nuovo* ma legato ad una connessione esistente
- **INVALID:** il pacchetto non ha stato o è invalido

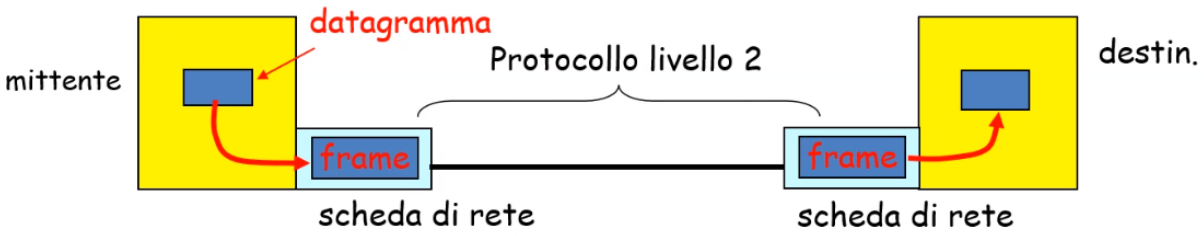
Il firewall prende le regole in ordine di scrittura

Le regole non stateful non usano la tabella degli stati.

Evitare attacchi bruteforce in SSH: non utilizzare la porta 22 per SSH. Si può usare **LIMIT** per impedire troppi tentativi in un lasso di tempo troppo breve.

Cose riguardo layer 2

Parliamo direttamente utilizzando le **interfacce di rete.**



I datagrammi qui prendono il nome di **frame**. Il frame viaggia sul link da buffer a buffer. Ci sono diversi tipi di link:

- **punto – punto:** due interfacce comunicanti in modo diretto. Non utilizzano qualcuno per comunicare.

- **link broadcast:** gruppi di vicini. Wifi e reti Ethernet sono un esempio. Nei link broacast **tutti vedono tutti;** sono identificabili tramite il **mac address.**

Il **mac address** viene generato in base alla **marca del prodotto. L'indirizzo IP** in base alle **subnet**

La tabella arp associa ai *vicini* il MAC ADDRESS. **Address Resolution Protocol.** Praticamente fa una mappatura **dell'indirizzo IP** e **indirizzo MAC** di un terminale in una rete locale internet

Internet Address	Physical Address	Type
192.168.0.1	00-11-32-76-58-c2	dynamic
192.168.0.3	e8-03-9a-9a-30-ce	dynamic
192.168.0.9	ac-16-2d-22-a0-9a	dynamic
192.168.0.18	88-3d-24-15-70-2e	dynamic
192.168.0.19	00-11-32-6d-2d-93	dynamic
192.168.0.24	00-16-6c-d7-0c-6a	dynamic
192.168.0.48	d8-8c-79-53-a5-95	dynamic
192.168.0.86	28-77-f1-e9-4d-36	dynamic
192.168.0.99	a0-63-91-30-c8-ca	dynamic
192.168.0.255	ff-ff-ff-ff-ff-ff	static

Figura 3: tabella arp

Struttura del frame:

- **indirizzi:** mac address sorgente e destinazione
- **type:** codice del protocollo di livello 3 trasporotato
- **crc:** viene controllato dal ricevente; se errato viene scartato il frame. *– è uno spazio all'interno di un frame che serve a vedere se il frame è corretto*

I pacchetti wifi hanno un terzo **mac address** che è quello dell'access point usato per connettersi.

Layer 2

In questo layer **router e host** prendono il nome di **nodi;** i canali di comunicazioni che connettono dei nodi prendono il nome di **link.**

- **Link in rame**
- **Link senza fili**
- **Link via satellite**

Link broadcast e condivisione del canale

Essendo che tutti vedono tutti, ci potrebbero essere delle **interferenze** quando due interlocutori parlano assieme; appunto perché ora parliamo proprio di segnali elettric e sonori

- **Collisione:** quando un nodo riceve due o più segnali in contemporanea

Come si risolve il problema e come si gestisce l'accesso al **canale condiviso.**

- i nodi devono mettersi d'accordo su chi deve parlare prima
- le comunicazioni devono essere sullo stesso canale

3 tipi di protocolli di condivisione:

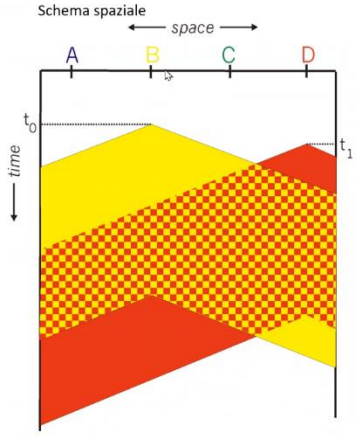
- **suddivisione del canale stesso (cellulari gsm)**
    - problema risolto a livello 1

- **ad accesso casuale (ethernet)**
    - su sopportano le collisioni

- **a turni (token ring)**
    - si trasmette quando si ha il token

Risoluzione accesso al canale condiviso:

CSMA (Carrier Sense Multiple Access): si sta in ascolto prima di trasmettere. Se il canale è occupato si ritarda la trasmissione. Questo **non funziona sempre.** Potrebbe accadere *che due stazioni vedano il canale libero in contemporanea.*



- *a t0 trasmette B perché vede libero*
- *a t1 trasmette D perché il segnale di B ancora non è arrivato*
- *c riceve un segnale con interferenza*

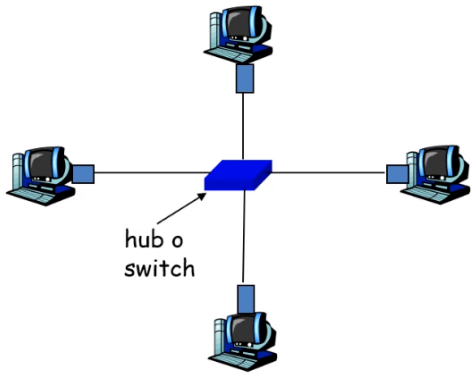
Mitigazione del problema

Si usano **CSMA/CD** (**collision detection**)

- Quando trova una collisione, la trasmissione viene **fermata subito** e il canale si libera
- Ora ci sono 2 strade:
    - Si ignora
    - Si ripete

Si utilizza il **backoff esponenziale**: ogni volta che una trasmissione fallisce oppure viene abortita per collisione, **si aumenta un contatore** che per comodità chiameremo **m**. quando la trasmissione viene abortita, prima di ritrasmettere si attende un tempo casuale **K**. Questo tempo è calcolato prendendo un numero casuale tra **0 e 2^m-1**; quindi in tutto **K^{512}bit-time**

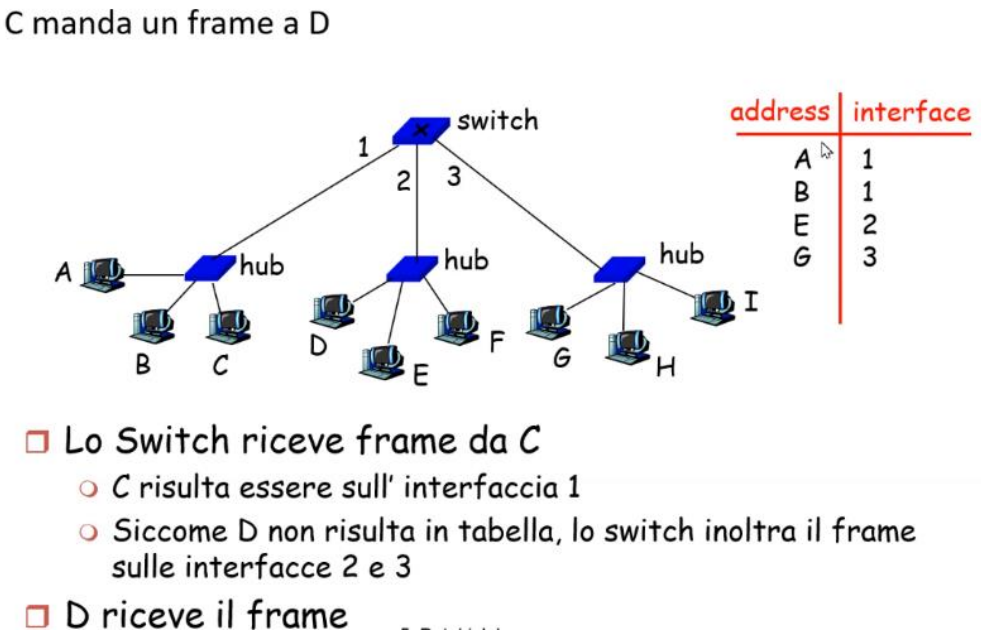
Topologia a stella



Hub:

- Viene ripetuto tutto quello che entra in ingresso
- È di livello 1
- Non ci sono buffer
- Non rileva le collisioni

Switch: sono intelligenti perché vedono all'interno del datagramma per capire dove instradarlo.



Lo switch ha una forwarding table che viene aggiornata quando capisce la posizione di un endpoint. Capisce l'interfaccia con la quale comunica e la salva insieme al mac address della stazione. Attenzione, la grandezza è finita.

Mac spoofing: attacco fatto per riempire le forwarding table ti uno switch e saturarlo per renderlo inefficiente e farlo funziona come un hub.

Ci sono 2 tipi di switch:

- **Managet**: configurabili
- **Non managet**: configurazione quasi impossibile

Negli switch **managet** è possibile configurare la **Virtual Local Area Network (VLAN)** – si possono implementare più domini di collisione virtualmente separati.

Con le VLAN si può avere una struttura complessa chiamata **Trunking** dove si può comunicare con switch di gruppi diversi.

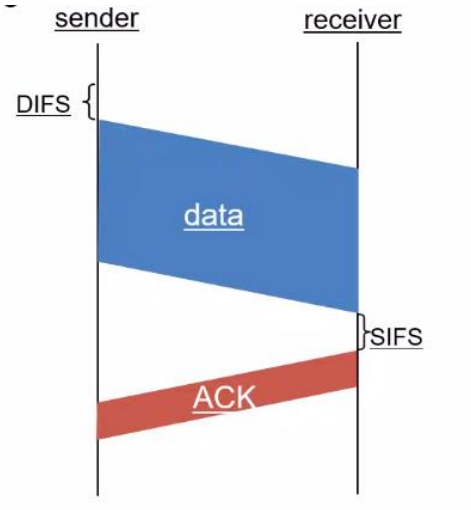
Problema delle stazioni nascoste

Quando si parla ad esempio tramite wifi, potrebbe essere che **per mancanza di segnale** il segnale possa essere visto come **libero.** Questa visione libera della stazione potrebbe portare ad una collisione. Nel link via cavo questa cosa non esiste.

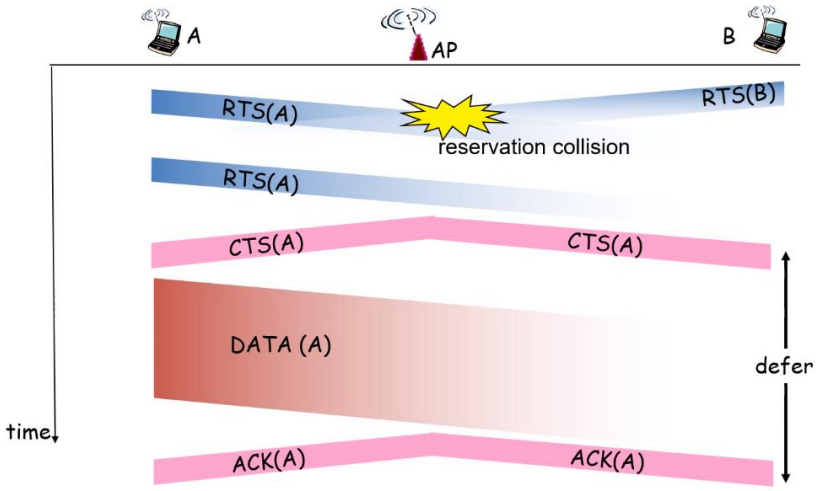
Collision Avoidance – CSMA/CA

Ci sono 2 modalità:

- **Trasmetti il frame subito:**
  - se il canale è muto, aspetti un tempo DIFS e trasmetti il frame per intero.
  - Se il canale è occupato avvia un timer backoff random
  - **----**
  - Il ricevente, dopo aver ricevuto il dato, aspetta SIFS e manda un hack



- **Prima di trasmettere chiedi l'autorizzazione: request to send, clear to send RTS-CTS**



Da qui in poi spiego questo esempio numerando i pasaggi

1) A e B mandano la **RTS** all'access point
2) Arrivano in contemporanea e quindi c'è una **collisione e vengono ignorate entrambe**
3) Vanno entrambe in **backoff esponenziale random**
4) **A** manda la RTS e A l'access point lo riceve
5) **L'access point** manda la **Clear To Send** (**CTS**) con parametro (A) che indica chi può mandare i dati
6) **A** manda i dati.
7) **L'access point** manda un ACK che I dati di a sono stati ricevuti

LAYER 1

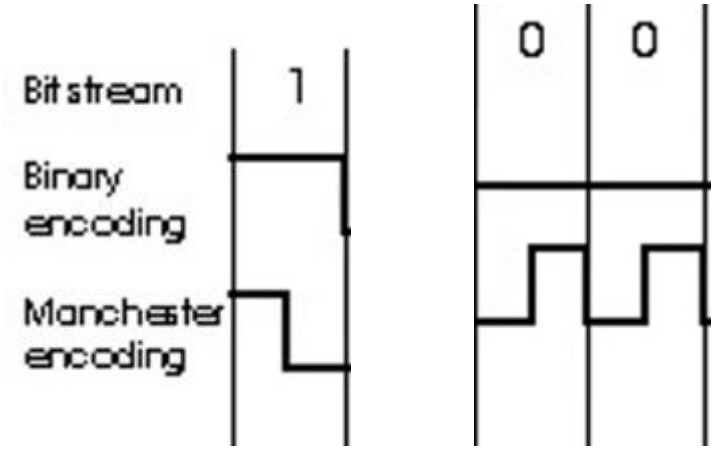
MAC: Medium access controller.

In questo layer il **frame** è una stringa di bit che vanno inviati.

- Modalità fisica dell'informazione
- Standardizzazione dei cavi, connettori e apparati
- Materiali usati

Ci sono meccanismi di conversione da **digitale ad analogico** in **trasmissione**, e da **analogico** a **digitale** in **ricezione.**

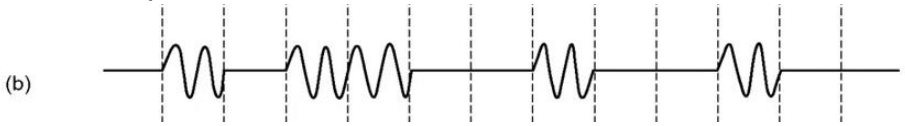
Nel cavo ci sono segnali analogici e alle estremità vanno trasformati in digitali.



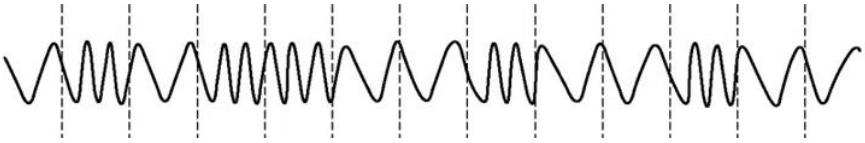
Manchester encoding: ha un limite di banda. Fa capire quando inizio un 1 e quando finisce ecc.

Meccanismi di modulazione

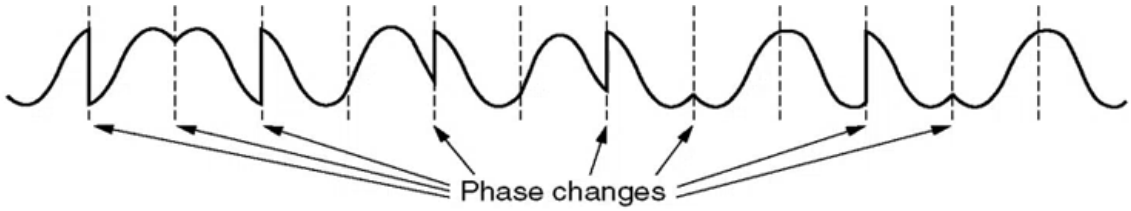
- **Modulazione in ampiezza:** quando c'è segnale è un 1 e quando c'è silenzio uno 0. La sinusoide ha ampiezza fissata.



- **Modulazione in frequenza:** si scelgono due frequenze. Una per lo 0 e un'altra per l'1.



- **Modulazione di fase:** si ha un'unica frequenza. Quando si vuole trasmettere un valore diverso dal precedente, si shifta a dx o a sx in base al valore di fase e in base a quello che si vuole trasmettere.



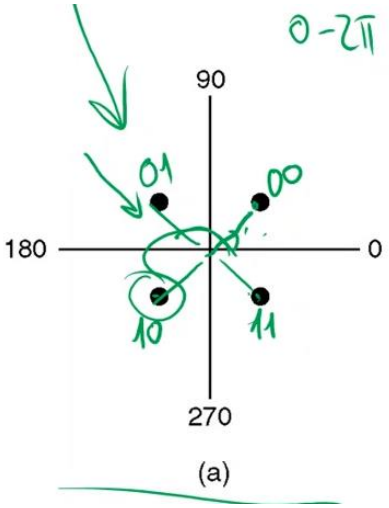
Combinando queste frequenze otterrei una modulazione multifase, **2 frequenze e 2 fasi**

Bitrate e baudrate

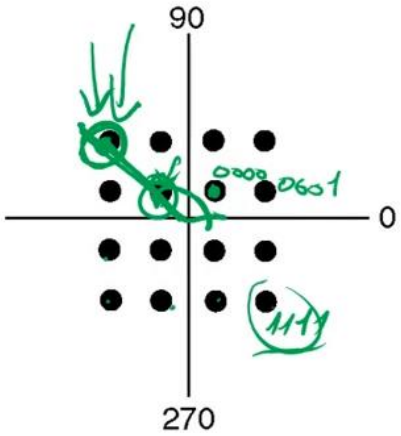
Una stazione trasmette a 10 baud/s; ogni simbolo è associato ad un codice a 2 bit**(QPSK)**; diciamo che trasmette a 20 bit/sec; bisogna sapere quanti simboli si stanno usando però.

In base al numero di simboli si hanno diversi modi

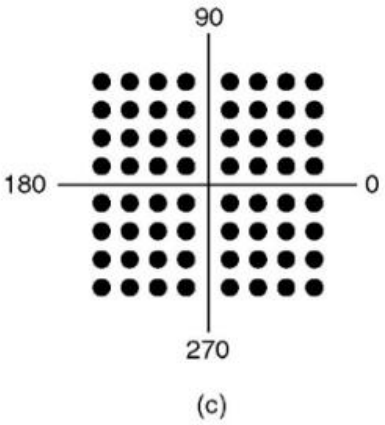
- **QPSK:** ogni intervallo di simbolo si possono trasmettere 4 cose diverse. Ogni puntino ha associato 2 bit.



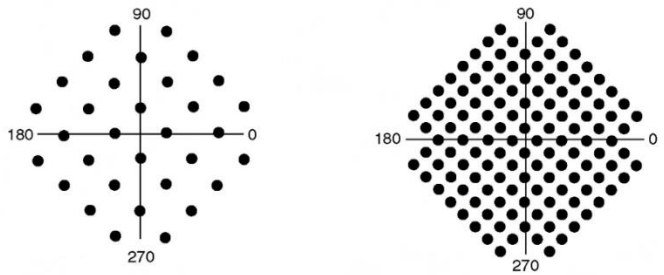
- **QAM-16:** ogni puntino ha associato 4bit. Utilizzato nel wifi e digitale terreste. Ci sono quindi 16 simboli in totale.



- **QAM 64**



(c)



(a) (b)

(a) V.32 for 9600 bps.
(b) V32 bis for 14,400 bps.

Modalità di trasmissione standard 802.11AC: MSC0 è il modo più semplice per trasmettere e prevedere BPSK che sarebbe la trasmissione a due simboli.

Aumentando il numero aumentano i simboli utilizzati e quindi il tipo di modulazione; **cambia anche il bit rate**. Con 16 simboli con un canale da 20megaHZ si arriva ad un throughput massimo di 26 Mb

Nelle reti wifi usando gli **stream spaziali** si può quadruplicare il valore.

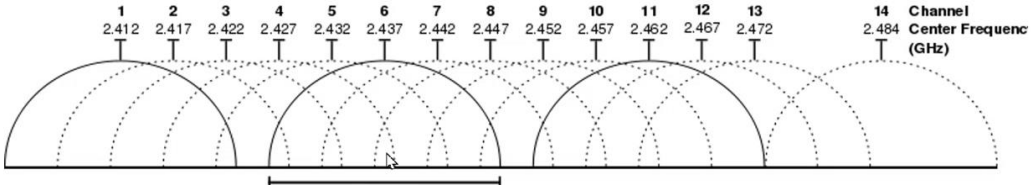
Stream Spaziale: si fa oscillare il campo elettromagnetico in verticale e diagonale

Ruolo della frequenza della sinusoide usata: Più frequenze hai più simboli puoi trasmettere più bitrate avrai.

Ruolo della frequenza del canale: lo spettro delle frequenze generali sono spalmate su più frequenze. Una sinusoide pura è quando il suo spettro è spalmato solo in un luogo

I canali si creano filtrando ogni segnale elettromagnetico che cade fuori un certo range.

Allocazione dei canali: se parlo col canale 6 sento solo lui. Però come vediamo ci potrebbe essere una sorta di interferenza perché i canali si sovrappongono



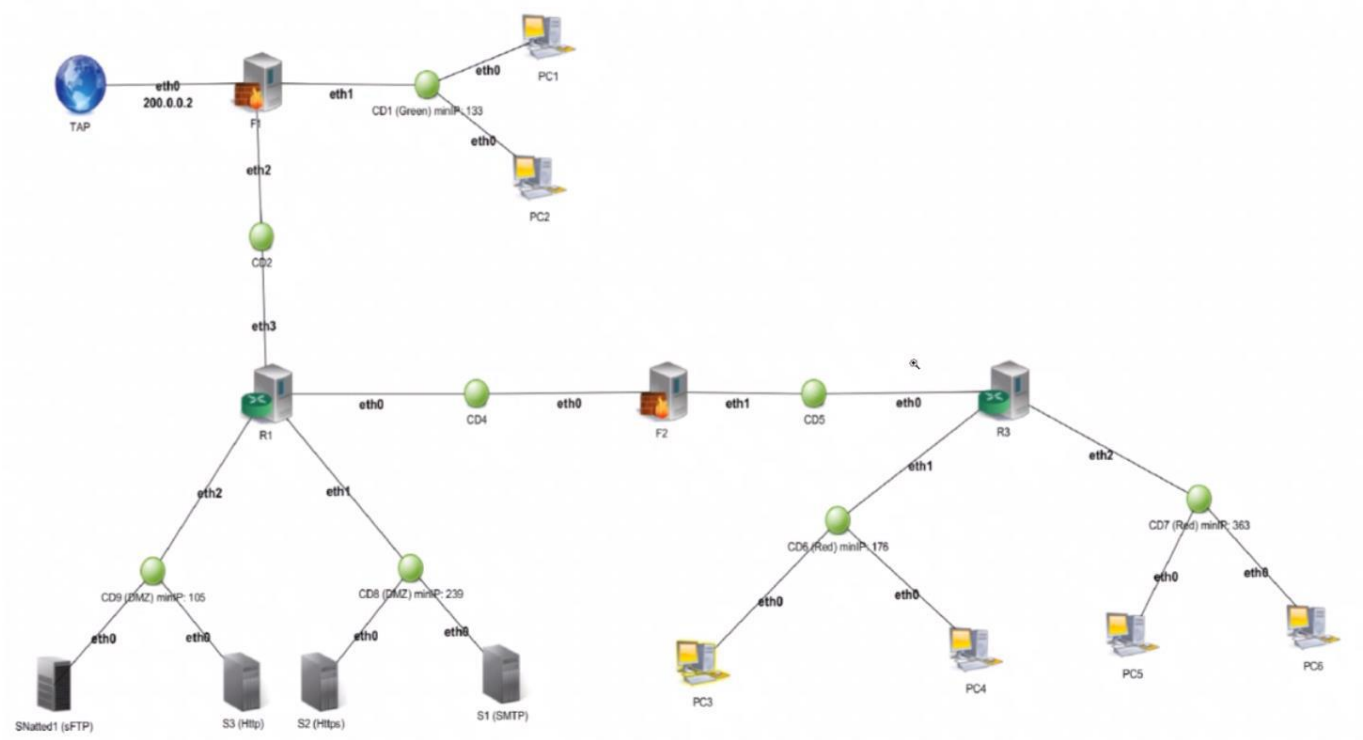
$\text{Tasso di trasmissione} = 2B \log_2 H$: dove **H è il numero di simboli** e **B l'ampiezza della bada**

Esagerando con i simboli potrebbe crearsi del **rumore di fondo (fruscio)** che ha uno spettro. Alcuni rumori di fondo possono essere naturali (come l'oscillazione delle molecole)

Bit di parità: è una piccola checksum che con una probabilità del 50% può individuare un errore e ripararlo

VDSL E CANALI.: I canali nelle frequenze alte viaggiano meno in km di quelli nelle bande basse. Il problema che abbiamo spesso con il wifi sono i limiti fisici e di lontananza dall'accesso point al pc.

Step GNS3 PER CALCOLARE TUTTO QUANTO E PASSAGGI PER LE VARIE COSE DA FARE



1) Identificare i **domini di collisione (il numero accanto ai pallini è il numero di indirizzi ip)**
    a. Dove manca il numero scriviamo 2
2) Assegnare la maschera ad ogni dominio di collisione: 2^X >= numero segnati + 2 (che sono network e broadcast)
    a. **Ad esempio:** 133 → 133 + 2 → 2^x >= 135. X = 8
    b. **Maschera =** BIT TOTALI – X. **32 – 8 =** 24. LA MASCHERA E' 24
3) Ordinare i domini di collisione in base alla maschera più piccola. Per comodità anche la divisione in aree la facciamo qui. Le aree sono **Green Red** e **DMZ**. Quelli senza aree li lasciamo li insieme senza aree.
4) Vanno ordinati ora i domini di collisione delle **singole aree** da maschera più bassa a più alta.
5) **Separare** le aree logicamente come se fossero sottoreti differenti. Calcolare, quindi, la maschera dell'area. Come si fa?
    a. Quando in un'area abbiamo solo 2 domini di collisione: *fai il minimo tra le due maschere dei domini e sottrai uno. Se è min(23,24) → allora farai 23 – 1 = 22.*
    b. Altrimenti fai una struttura ad albero. Parti disegnando la maschera più piccola e poi quella più grande. Se ad esempio avessimo 23 23 e 28: la **maschera 22** contiene 2 maschere 23, per la maschera 28 serve una maschera 27, per la 27 serve la 26, per la 26 serve la 25, ….., per la 23 **serve la 22**. Quale maschera contiene 2 maschere 22? La **maschera 21. L'area avrà maschera 21.**
6) **Ordinamento delle aree per maschera**
7) **Assegnare** network e broadcast. Si parte con 10.0.0.0 dalla prima area in cima.
    a. Esempio pratico: **Area Red** con CD7 (23) e CD6 (24).
    b. Si parte con CD7:
        i. Network: 10.0.0.0/23
        ii. Netmask**: 255.255.254.0**
        iii. Trasformiamo la netmask in formato ip: **11111111.11111111.11111110.00000000**
        iv. **Trovare il broadcast:** OR LOGICO TRA IL NETWORK E LA MASCHERA INVERTITA
        v. **NETWOKR IN BINARIO:** 00001010.00000000.00000000.00000000
        vi. **NETMASK INVERTITA:** 00000000.00000000.00000001.11111111
        vii. **BROADCAST:** 00001010.00000000.00000001.11111111 → 10.0.1.255

    c. **Ora tocca a CD6(24)**
        i. Per il network di CD6 dobbiamo prendere il **broadcast** del precedente e sommare un bit.
        ii. 10.0.1.255 → 10.0.2.0 (il 255 diventa 0 e passa ad 1 il prossimo numero)
        iii. E poi si procede regolarmente.
    d. Dopo aver calcolato anche CD6 bisogna calcolare il network e broadcast dell'area.
        i. Il network: rimane uguale al network del primo componente dell'area
        ii. La maschera è quella dell'area che abbiamo calcolato in precedenza

iii. Il broadcast va sempre fatto **network binario OR LOGICO maschera invertita**

e. **Quando si finisce questo e si passa ad un'altra area, si calcola il network sempre basandosi al broadcast dell'area precedente + 1**

Metodo calcolo veloce:

- Prendi il network (es 10.0.0.0)
- Prendi 32 – maschera (es 24) = 8
- Fai network + 2^8 - 1 *perché il network è già assegnato* (sommando partendo dall'ultima cifra) →10.0.0.0 + 255 = 10.0.0.255

Facciamo la prova con il metodo base:

- Network: 10.0.0.0/24
- Netmask: 255.255.255.0
- Broadcast: 00001010.00000000.00000000.00000000
  00000000.00000000.00000000.11111111

  00001010.00000000.00000000.11111111→ 10.0.0.255

Il resto delle cose riguardo al progetto ci sono sui file sul drive

Esercitazione comandi di Reti di Pacenza

Comandi per il debug di rete

Ping: ci consente di controllare un host A raggiunge un host B, indicando la latenza. Ci torna indietro *numero pacchetto, percentuale di pacchetto droppati, RTT minimo, RTT massimo e la deviazione standard.*

Ifconfig: vengono mostrate le informazioni sulle schede di rete presenti sulla macchina. Ci da informazioni se la scheda è UP o DOWN, quanti pacchetti sono stati ricevuti e inviati. Con ifconfig -a ci torna tutte le schede, anche quelle down. Per assegnare un indirizzo ip alla scheda si fa: ifconfig + nomescheda + indirizzo;

ip: mosttra le informazioni di route e ifconfig

Netstat: visualizza lo stato delle connessioni presenti sul pc. Mostra lo stato di tutti i **socket**

Telnet: è un comando che viene utilizzato per fornire all'utente sessioni di login remoto su vari host.

telnet + host + porta

Netcat: si usa per leggere e scrivere dati tramite internet.

- Netcat -l 12345 → avvia un socket server in ascolto su porta 123456
- Netcat ip + porta → avvia un socket client connesso su localhost su 12345

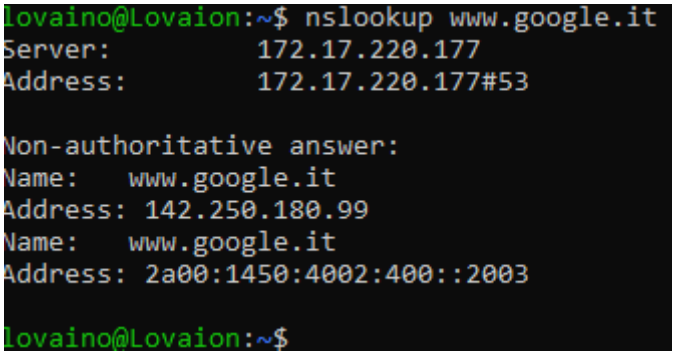
Wireshark: sessioni di wireshark mirato al link di questo oggetto.

Ethtool: serve a controllare i parametri di configurazione della scheda di rete.

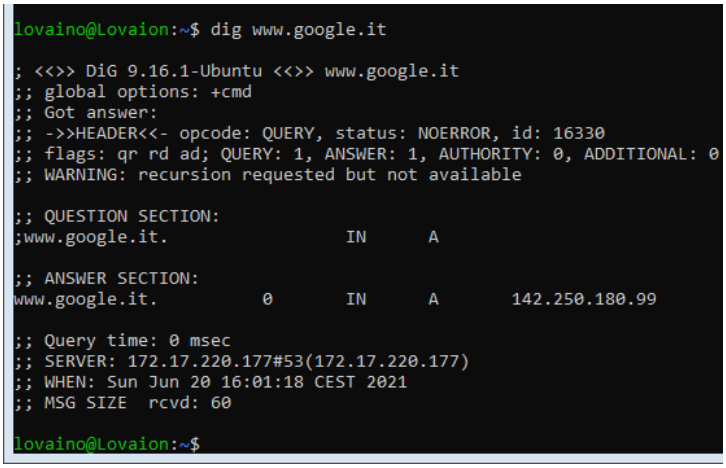
Ethtool + nomescheda

DIG/ NSLOOKUP: query su un server dns per la risoluzione di indirizzi ip e hostnam. Ottieni un dominio dall'ip e viceversa

Nslookup + indirizzo



DIG invece mostra più informazioni e ci sono diversi parametri da analizzare



IPERF: Misura le prestazioni della rete

- Iperf -c indirizzo (per client)
- Iper -s indirizzo (per server)

TCPDUMP: controllare se tra un router e un altro i pacchetti arrivano oppure no.

- **-nn** : non risolve sia nomi che porte.
- **-X** : mostra il contenuto dei pacchetti sia in esadecimale che in ASCII
- **-v**, **-vv**, **-vvv** : incrementa il numero delle infomazioni.
- **-c** : mostra solo un certo numero di pacchetti (es: -c 10).
- **-S** : stampa la sequenza di numeri assoluti
- **-e** : ottiene l'header.
- **-q** : mostra poche informazioni sul protocollo.
- **-w nomefile** : salva l'output in un file. Utile per poter essere analizzato successivamente. Aperto con wireshark, per esempio, può fornire informazioni più dettagliate e comprensibili

Route: mostra la tabella di routing.

- **Destination**: la destinazione del pacchetto
- **Gateway**: l'indirizzo della prossima scheda a cui verrà mandato il pacchetto
- **Ganmask**: la maschera
- **Flags**: il tipo di rotta. U è UP UG è up con gateway
- **Metric**: priorità della rotta in caso di uguaglianza
- **Ref**: numero di referenze all'interno di una rotta
- **Use**: numero di volta che è stata usata
- **Iface**: interfaccia

TRACEROUTE: il router manda un pacchetto all'interno del gateway. Lo manda fintanto che non si arriva allo step 6.