Internet, Reti e Sicurezza

Internet, Reti E Sicurezza (Università degli Studi di Camerino)

Università di Camerino

Internet, Reti e Sicurezza

Corso di “Internet, Reti e Sicurezza” tenuto dal prof. Fausto Marcantoni

COSCI TOMMASO

AA 2023/2024

/* Capitolo 1: Networking & The Internet */

//Cos'è Internet?

- Una rete che collega una serie di calcolatori sparsi geograficamente.
- Una rete a commutazione di pacchetto

I link di comunicazione collegano tra di loro questi terminali, che però non sono collegati direttamente ma tramite dei dispositivi di commutazione detti **Router**: questi comunicano tra loro attraverso delle **route e/o path**, intesi come itinerari da far compiere al pacchetto.
Ovviamente, non esiste un percorso dedicato tra terminale e terminale, ma più di essi condividono un intero cammino o una parte --> **commutazione di pacchetto**.

L'accesso ad Internet viene gestito attraverso gli **Internet Service Provider** (ISP), come Vodafone, Fastweb, Telecom Italia ecc.

In Internet parliamo di **applicazione distribuita**: un'applicazione costituita da due o più processi e servizi che girano su sistemi distinti e connessi da una rete di comunicazione. Questi processi sfruttano servizi tipo:
- **infrastruttura di comunicazione**: che consente di messaggiare e il trasferimento dati
- **due tipi di servizi** alle applicazioni distribuite: servizio affidabile (connection-oriented) e servizio non affidabile (connectionless)
- nessuna garanzia del tempo richiesto per spedire i dati

//Protocollo

È un insieme di **regole** formalmente descritte, definite al fine di favorire la comunicazione tra uno o più entità. È **rigido**, definisce come funziona lo scambio di messaggio e il loro ordine e formato. Possono essere **linee guida, Normative e +/- Standard**.

//Struttura della rete

In ordine di importanza la rete di suddivide in:
- **Network edge**: applicazioni e host
- **Network core**: routes & network of networks (parte più importante)
- **Physical media**: communication links & cabling (spesso fatto da elettricisti certificati)

--> Network edge: Connection-oriented service
Obbiettivo: trasferire dati tra due sistemi (end-system)

Generalmente costituito da **handshaking** e un **setup state**: devo comunicare dei parametri che servono a stabilire la connessione, una volta concluso l'handshaking inizio la trasmissione dei dati.
La trasmissione viene fatta attraverso il protocollo **TCP**:
- definisce **l'affidabilità**: come procedono i byte nel percorso, tramite acknowledge
- definisce il **flusso:** per evitare un overflow
- **controllo della congestione**: tener controllo delle informazioni inviate e delle informazioni ricevuto

--> Network edge: Connectionless service
Obbiettivo: trasferire dati tra due sistemi
Non ho handshaking e il setup state: continuo ad inviare dati anche se il destinatario non riceve i

pacchetti.
La trasmissione viene fatta attraverso il protocollo **UDP (User Datagram Protocol)**: è molto più semplice e consuma meno risorse, ma non definisce affidabilità, non controlla il flusso e non controlla la congestione

Tutte queste informazioni sui protocolli sono conservate sull'**IETF**.

--> Network core
Commutazione di pacchetto (packet switching): non c'è circuito, ma pacchetti che vengono inviate sulla rete e vengono instradate con i router a destinazione. Ho problema della congestione che comporta attesa.

Mandare dei pacchetti da mittente a destinatario, i pacchetti vengono inviati ai commutatori di pacchetto (router). Non c'è una strada unica per un determinato un pacchetto: sarà compito del protocollo prendersi carico della strada del pacchetto.

Commutazione di circuito (circuit switching): quando viene stabilito circuito per arrivare a destinazione, una volta collegato ho molta libertà. Una volta definita lo state posso fare quello che voglio. In questo caso prenoto un circuito da A e B attraverso degli apparati e funzionava in 2 modi:

- Divisione di frequenza (FDM)
- Divisione di tempo (TDM)

Router: apparati complessi che consente due o più reti su internet. Sceglie il percorso migliore da sorgente a destinazione. Attraverso il **destination addres**, il router determina il **next hop**.
Tuttavia possiamo creare dei **circuiti virtuali**, che impostano un percorso sempre fisso che può essere associato ad un pacchetto.

Router svolge **store&forward**: il router riceve prima un pacchetto prima di poter cominciare a trasmettere e controllare che sia arrivato tutto e poi capire qual è la strada da percorrere. Quando svolgo questa operazione innesco un ritardo.
quindi nel router dovrò avere dei componenti: buffer in ingresso e buffer di uscita. Zone di memoria che costano e che permettono di attendere il tempo necessario per la trasmissione. Sono fattori che dipendono dalla congestione della rete, dalle perfomance del router.
Obbiettivo Forwarding: girare i pacchetti da un router ad un altro

Architettura di internet

Rete Locale: rete casalinga, rete di azienda che avrà uno o più collegamenti ad un'area utente.
Area Accesso: che mi permette di collegarmi al **backbone** (centro di internet)
Le modalità di accesso possono essere: accessi residenziali, aziendali, e wireless.

Sistemi autonomi

Sistemi autonomi (Autonomous System): un insieme di hosts, routers e reti fisiche controllate da una singola autorità amministrativa. Ogni AS è identificato dalla IANA, e in Italia da, registro.it.

Ogni AS deve però affidare ad uno o più routers il compito di comunicare con il mondo esterno. Quindi vengono costituiti due protocolli:

- **Interior Gateway Protocol**: per instradare informazioni internamente all'AS
- **External Gateway Protocol**: per instradare informazioni all'esterno dell'AS

Problemi di rete

Attraversando i vari router, vengono indotti dei ritardi:
- **elaborazione**: ritardo che subisce il pacchetto nel momento in cui viene elaborato dal router, per controllo dei bit. Questo è fatto nel calcolo del pacchetto per controllare che sia giusto. Devo vedere il path e devo capire su che porta inviarlo
- **accodamento**: ritardo che una volta elaborato il pacchetto e calcolato il next hop viene messo in buffer dove ci sono altri pacchetti. Il protocollo TCP prevede un timeout in caso di troppo tempo di queue
- **trasmissione**: tempo di trasferimento, in base al peso del pacchetto.
- **propagazione**: tempo che viene usato per la propagazione tra uscita e ingresso.

la cui somma costituisce il ritardo del nodo.

Il ritardo del **nodo** è quello della **somma** di tutti i ritardi: non c'è una formula matematica

Larghezza di band

Numero di bit che possono essere trasmessi in una determinata quantità di tempo.
10 milioni di bit al secondo --> 10Mbps

L'ampiezza di questa banda dipende dal mezzo trasmissivo è legata strettamente alla frequenza che posso modulare all'interno del mezzo.

Prestazioni di una rete

Bitrate: la velocità con la quale posso indurre segnali elettrici nel mezzo di trasmissione
Throughput: portate della rete, non riuscirò mai a sfruttare la portanza massima della rete ma ottengo sempre un po' di meno. Inoltre in rete ci sono vari router che hanno diverse velocità, ma avrò sempre quello più piccolo (collega ai ritardi).
Limite massimo che posso ottenere nel trasferimento di dati nel link tra sorgente e destinazione.
Latenza: intervallo di tempo tra input al sistema e la disponibilità dell'output. Parametro usato per calcolare il tempo che un pacchetto impiega per andare e tornare.
Quindi il vero Throughput è la combinazione con latenza e banda.

Nelle Traceroute ci sono delle **CDN** (Content Delivery Network): sono quei siti che vengono distribuiti in tutto, il mondo su vari server, per velocizzare le risposte dei client.

// Protocol Layering: il funzionamento della rete

Le reti sono molto complesse, con molti apparati hardware e software: dobbiamo trovare un modo semplice e efficiente per far si che la rete funzioni in maniera corretta.
La stratificazione permette di far eseguire azioni specifiche ad ogni livello: possiamo cambiare un livello senza cambiare gli altri; posso utilizzare le stesse informazioni dei livelli inferiori; la modularità mi permette la facile manutenzione e modifica.

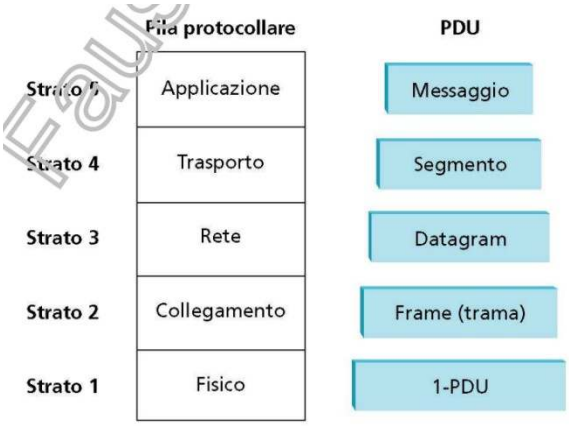
I principali livelli attualmente utilizzati (e esistenti) sono: **Application, Transport, Network, Link, Physical**.

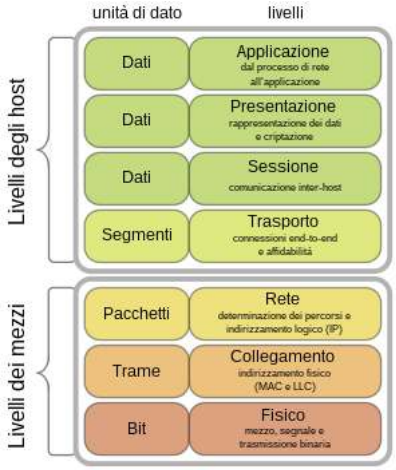
PDU (Protocol Data Unit): una stringa di bit processata per ogni livello n dove ci sono informazioni sul livello che stiamo utilizzando. È costituito da:

- **Header**: intestazione del livello. Queste informazioni mi dice come lavorare il payload
- **Payload**: contenuto dell'informazione

Ogni livello mettono a disposizione **interfacce** che mettono a disposizioni dei **servizi**, utili sia al livello successivo sia a quello precedente.
Servizio: insieme di operazioni primitive che un livello offre ad un livello **superiore**.

Questa pila di livelli viene definita come **architettura di rete**.





Funzioni di strati - Pila protocollare

Applicativo: mette a disposizione i protocolli che possono essere usati da qualsiasi applicazione
Trasporto: connection orinted con TCP non orientend con UDP
Rete: responsabile dell'instrdamento dei datagram. Dovrò capire a queli host sono collegato e devo utilizzare i vari protocolli per tracciare il path del paccheto.
Data link: collegamento nella rete. Informazioni per muovre il pacchetto da un nodo al successivo.
Fisico: i frame del livello due diventano degli 0 e 1 che vengono fisicamente trasmessi tramite dei mezzi fisici.

Esistevano anche i livelli di presentazione e di sessione che organizzavano in maniera generale i dati e la comunicazione tra parti --> **OSI.** Si sono definiti quindi degli **standard** al fine di garantire il funzionamento internazionale dei prodotti:
- standard **DE FACTO**: sono degli standard del TCP/IP non codificati da un ente standardizzante utilizzati al livello mondiale
- standard **DE JURE**: sono standard che seguono specifiche internazionali standardizzante da enti specifici
- standard **PROPRIETARIO**: sono standard usati in reti privare che generalmente sono incompatibili con sistemi differenti. Raramente le specifiche sono rese pubbliche.

Indirizzamento

Abbiamo 3 livelli di indirizzo
indirizzo fisico: indirizzo dell'host per la comunicazione, ogni macchina ha il suo
indirizzo IP: mi permette di essere raggiunto da tutto il mondo univoco in tutto il mondo
- versione 4: 32 bit e sono finiti, non più sufficienti.
- versione 6: 128 bit, molto più flessibile

indirizzo porte: applicazione e trasporto

/* Capitolo 2: Application Layer */

Nel nostro caso parleremo di applicazioni di rete: sono dei **processi distribuiti in comunicazione**; programmi che stanno nella rete che comunicano uno con l'altro: sono su degli host che si scambiano dei messaggi.
Ogni applicazione presenta un protocollo applicativo --> quindi il protocollo è una **parte dell'applicazione stessa**, che definisce il formato dei messaggi scambiati e il loro significato.
Trovandosi poi nello strato applicativo della pila protocollare riesce a dialogare con gli strati inferiori.

Ci sono 2 principali architetture: client

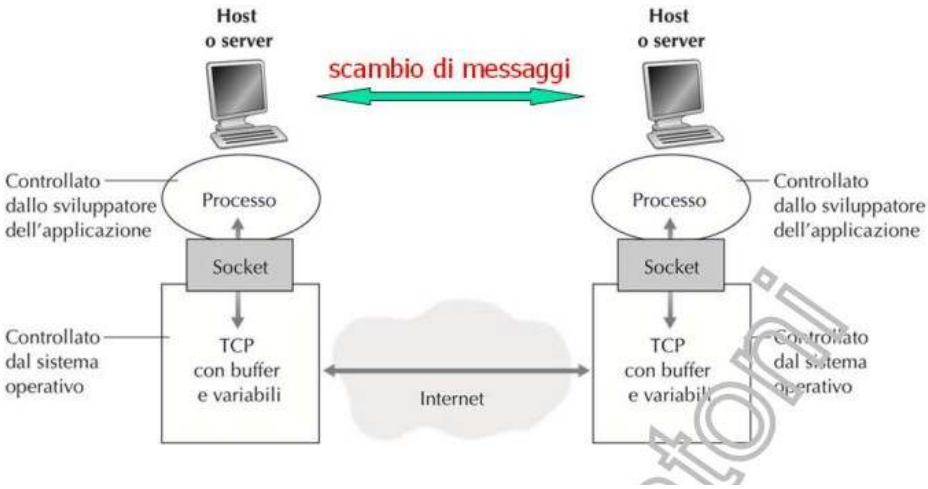
// Terminologia e concetti essenziali

- architettura **client-server**: per ciascuna coppia di processi comunicanti, se ne etichetta uno come client e uno come server
- architettura **peer-to-peer**: dove i due processi vengono si possono comportare come client e/o come server simultaneamente.
- **processo**: programma in esecuzione su un host
- sullo **stesso host** i processi comunicano mediante **meccanismi definiti dal SO**.
- processi su **host diversi** comunicano mediante meccanismi definiti dal **protocollo dello strato di applicazione** (application layer protocol)
- **processi comunicanti**: programmi in esecuzione su due sistemi e si scambiano dei messaggi
- **messaggi**: informazioni scambiate tra processi su due sistemi terminali attraverso la rete
- **socket**: interfaccia software che permette la comunicazione tra il livello Applicativo e quello di trasporto sottostante

PER VEDERE I PROCESSI ATTIVI --> *tasklist* sul CMD

Le applicazioni al loro interno hanno un **user agent**: interfaccia tra utente e l'applicazione di rete, sa come funziona un protocollo e permette di usarlo all'utente (il browser è l'user agent dell'HTTP).

Tutto funziona tramite **client-server**: normalmente il server è sempre attivo e accetta tante richieste; il client sollecita il server a fare azioni ed è quello che inizia il dialogo.
Esistono poi delle **API (Application Programming Interface)**: sono delle interfacce software per accedere ai servizi che vengono messi a disposizione.
Inoltre il processo identifica quello con cui vuole comunicare attraverso l'**Indirizzo IP** (individuo l'host su cui è l'altro processo) e il numero di porta (**port number,** Identifico il processo destinatario)



I protocolli di trasporto che un processo (o

applicazione) può usare sono 2:
TCP (Transmission Control Protocol): servizio orientato alla connessione, detto connection-oriented, che viene ritenuto più sicuro ed affidabile
UDP (User Data Protocol): protocollo connectionless, che è sicuramente più veloce ma non garantisce che il messaggio arrivi.

// Specifiche del Protocollo Applicativo (informazioni valide per tutti i protocolli)
- **tipo di messaggio**: se è una richiesta o una risposta
- **campi del messaggio**: in base alla request o alla reply
- **semantica del messaggio**: significato delle informazioni dei campi
- **regole**: regole del messaggio sul come e quando avviene lo scambio di messaggio

il comando per visualizzare i protocolli in uso è netstat

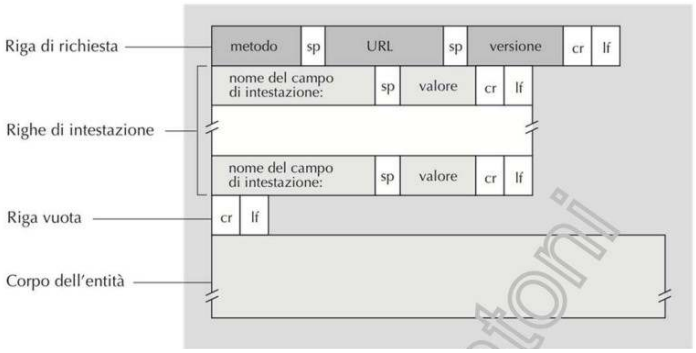
--> Protocolli di livello applicativo: HTTP (Hyper Text Transfer Protocol)

È il protocollo Web per eccellenza e funziona esclusivamente con l'architettura **Client-Server**: infatti il client attiva la connessione e richiede i servizi, il server accetta la connessione risponde alla richiesta ed è quello che poi la chiude.
È un **protocollo generico**, poiché è indipendente dal formato dei dati con cui vengono trasmesse le risorse di solito: un file HTML che fa riferimento ad altri oggetti della pagina tramite il rispettivo URL (Uniform Resource Locator).
È **stateless**: infatti il server non è tenuto a mantenere informazioni che persistano tra una connessione e la successiva. In altre parole, se un client manda due richieste identiche, il server risponderà 2 volte con la stessa risposta.

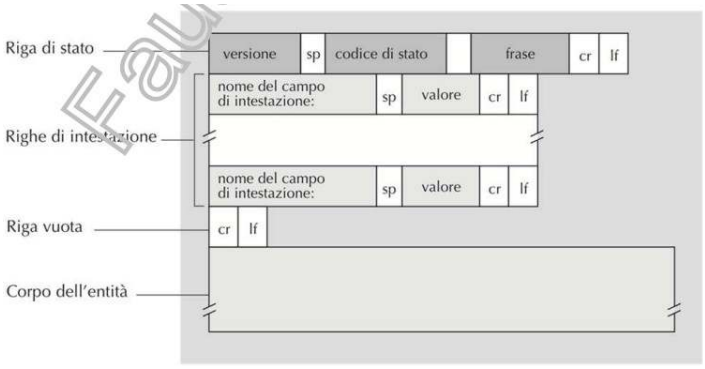
La connessione http può essere:
- **persistente**: tutte le richieste/risposte avvengono nella stessa sessione TCP
- **non persistente**: ogni richiesta/risposta avviene in una sessione TCP a sé (default)

Come calcolare il **response time** di un server HTTP?
Unità RTT (Round Trip Time): tempo necessario per il viaggio di andata e ritorno di un piccolo pacchetto.
Si consideri che: per iniziare la connessione TCP serve **1 RTT**, per l'HTTP request serve **1 RTT** e poi c'è il file **transmission time**.
Quindi il totale = **2RTT + il transmission time**.

Message http request



Message http response



// Cookies

Il cookie è un tipo di informazione che viene memorizzato da client, che verrà riutilizzato per ristabilire una connessione. Viene considerato come un "**tagliando**" per il diritto di accesso a quel contenuto: in questo modo anche se l'HTTP è stateless, il client può mantenere lo stato delle precedenti connessioni.

Sono facoltativi e possono essere disattivati, anche perché sono un pericolo per la Privacy: profilano gli utenti e raccolgono informazioni sulle abitudini e comportamenti, con l'obbiettivo di analizzare le attività in rete e fornire a ciascuno pubblicità mirate --> **Pubblicità Comportamentale**.

// Web Caching (server Proxy)

Il server Proxy è un'entità della rete (che si presuppone essere vicina ai client) che può rispondere ad alcune richieste HTTP evitando di inviarle direttamente al server.

Una volta che il client invia la richiesta, il proxy reagisce in 2 modi:
- ha l'oggetto richiesto e lo restituisce
- non ha l'oggetto richiesto, lo richiede al server e lo restituisce al client

I vantaggi ovviamente sono in primis la velocità, la diminuzione del traffico verso server lontani e il filtraggio dei pacchetti, che implica una maggiore sicurezza.

--> Protocolli di livello applicativo: Telnet

Telnet è un'applicazione standard di Internet: è un protocollo che permette un **login remoto**, implementando un modello di tipo server-client. Ciò permette ad un utente attestato ad una certa macchina di stabilire una connessione TCP con un server di login che si trova su un'altra macchina. La sua porta di default è la **23**.

Essenzialmente, Telnet rilancia i caratteri battuti sulla tastiera dall'utente direttamente al calcolatore remoto, come se la tastiera fosse connessa ad esso. Poi rimanda l'output della macchina remota direttamente allo schermo dell'utente.

Ora al suo posto si usa **SSH** (Secur Shell) poiché non è il massimo della sicurezza: i dati vengono inviati sottoforma di ASCII, quindi facilmente leggibili, compresi i dati sensibili.

--> Protocolli di livello applicativo: FTP (File Transfer Protocol)

È ovviamente un protocollo per il **trasferimento di file** da/verso un host. Usa il modello client/server e comunica tramite TCP su porta **20 e 21**: il client contatta il server sulla porta 21 che viene considerata di "**controllo fuori banda**", mentre il **trasferimento dei dati** avviene sulla porta 20.

Entrambe le connessioni vengono aperte dal client e per ogni file trasferito viene instaurata una nuova connessione. Spesso anziché usare FTP si scambiando file usando HTTP poiché anche in FTP, i dati transitano in ASCII (quindi tutto in chiaro, compresi username e psw).

// PASV-PORT

Attualmente usare FTP può dare problemi per l'avvento di NAT e Firewall. Sia **PASV** che **PORT** sono comandi utilizzati per la connessione dati. In particolare, il trasferimento dati può essere ostacolato dai Firewall. Quindi si è reso necessario aggiungere PASV, che non è altro che un modo di utilizzare il trasferimento dati anche con i Firewall.

--> Protocolli di livello applicativo: TFTP (Trivial File Transfer Protocol)

Rispetto al FTP, utilizza **l'UDP** sulla porta **69**. È detto Trivial proprio perché mette a disposizioni **poche funzioni**: è molto semplice, non conosce il concetto di directory, non prevede autenticazioni ed ha un tempo di utilizzo molto limitato.

La modalità di traferimento dati di TFTP è di **2 tipi**:
- **ASCII** (netASCII)
- **binario** (OCTET)

I pacchetti UDP inviati sono a lunghezza fissa di **512Byte** e la trasmissione viene considerata conclusa quando viene ricevuto un pacchetto < 512Byte.

--> Protocolli di livello applicativo: SMTP (Simple Mail Transfer Protocol)

// Internet Mailing System

Le 3 componenti principale del sistema di mailing Internet sono:
- **user agents**: strumento che legge e scrive la posta elettronica. Lettore con la posta elettronica (diverso dal browser, che mi fa vedere il messaggio). Anche la gestione della posta elettronica viene fatta tramite user agent (es Outlook)
- **mail server**: ha le mailbox gestite tramite DB che contengono i messaggi dell'utente. Vi è anche una coda dei messaggi e ovviamente vi è un protocollo SMTP che fa sia la spedizione sia la recezione.
- **SMTP:** protocollo principale a livello applicativo per la posta elettronica

Il protocollo SMTP viene usato per il trasferimento affidabile dei messaggi da client a server sulla porta **25**. Non fa altro che consegnare e memorizzare nel server di posta del ricevente. Ci sono poi diversi protocolli di accesso alla posta del server, come **POP, IMAP e HTTP** (gmail, outlook...).
Il protocollo della mail è un **push protocol**: quindi mi avvisa solo quando ho qualcosa da riferire, non lo tengo sotto controllo continuamente.

--> Protocolli di livello applicativo: POP-IMAP (Post Office Protocol – Internet Mail Access Protocol)

Il protocollo **POP3** lavora sulla porta **110** (default e non criptata) e sulla porta **995**, per connettersi in maniera sicura. Questo protocollo consiste essenzialmente in fase di **autorizzazione** e di **transizione** (se la prima fase è avvenuta con successo).

Il protocollo **IMAP** invece lavora sulla porta **143** (default e non criptata) e sulla porta **993**, per connettersi in maniera sicura.

// POP3 VS IMAP

POP scarica le mail sul computer, le cancella dal server e si disconnette
IMAP rimane connesso e si limita a leggere il contenuto (risparmia tempo su messaggi grandi)

POP assume che solo un cliente sia connesso ad una determinata mailbox
IMAP permette connessioni simultanee alla stessa mailbox

IMAP permette di creare, modificare o cancellare mailbox sul server e permette di creare cartelle condivise tra utenti.

IMAP permette di fae ricerche di messaggi sul server senza doverli scaricare tutti.

POP invia anche le password in testo, rendendone facile l'intercettazione
IMAP cripta anche le password.

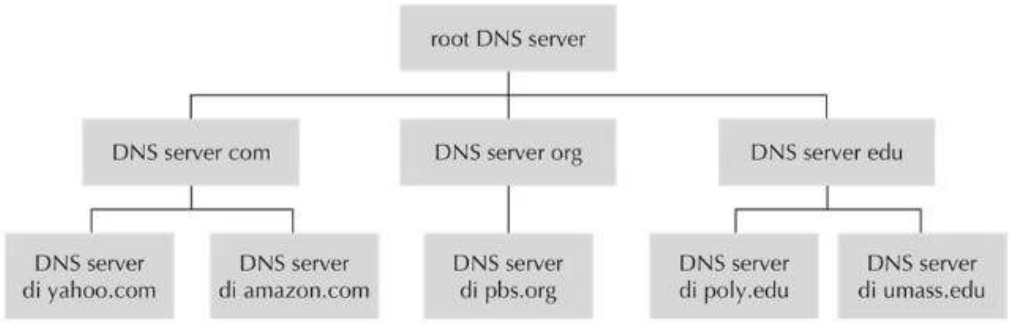
--> Protocolli di livello applicativo: DNS (Domain Name System)

Il protocollo della mail è un **push protocol**: quindi mi avvisa solo quando ho qualcosa da riferire, non lo tengo sotto controllo continuamente. Come faccio a individuare gli host sulla rete? Utilizzo anzitutto l'indirizzo IP che sono univoci in tutto il mondo (non più sufficienti). Sistema che associa un nome mnemonico ad un indirizzo IP.

È quello che fa il **DNS**: database distribuito su tutta la rete e **traduce** gli hostname in indirizzi (e viceversa). Non ce n'è uno solo in tutto il mondo poiché porterebbe diversi problemi. Allora viene decentralizzato, in modo che ogni computer può ricevere il problema

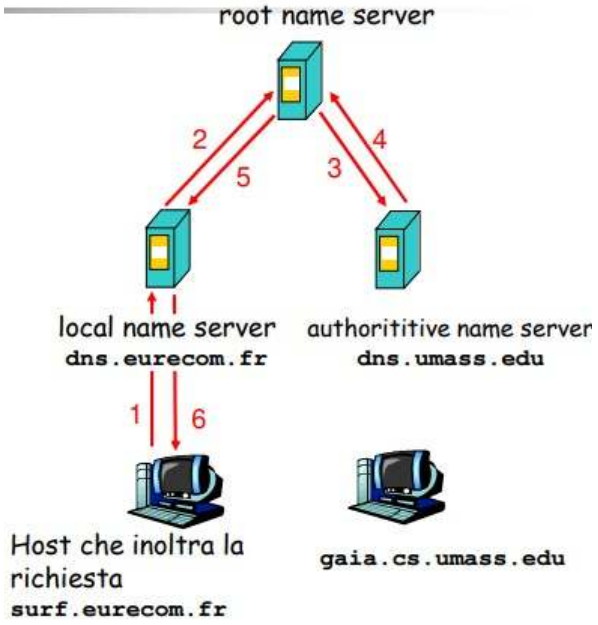
DNS primario (master) ha i dati effettivi e ogni tot di tempo il primario fa il trasferimento di zona con il **DNS secondario** (slave), che ne contiene una copia. Ogni qualvolta un amministratore modifica le informazioni sul DNS primario, i secondari scaricano la zona automaticamente.

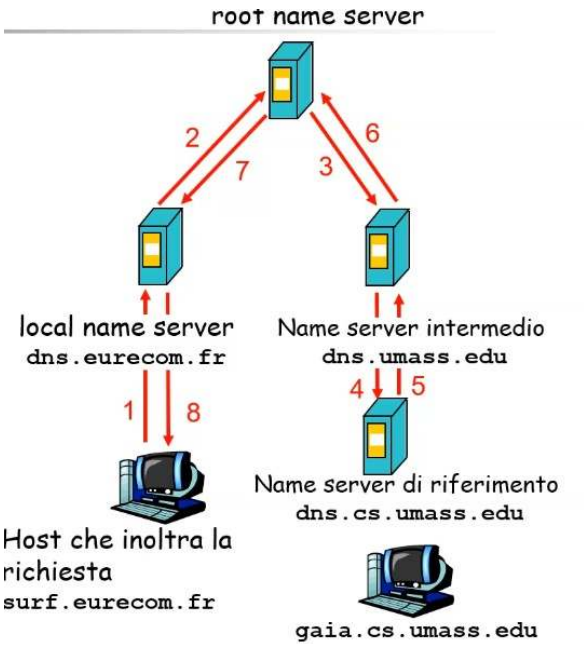
Gerarchia di server DNS



Route name server: apici della gerarchia, quello a cui mi riferisco ogni qualvolta faccio una ricerca (sono circa 1000 nel mdondo). Poi ci sono i top-level domain (.com) e per ultimi gli authoritative (facebboik.com)
Gli **FQDN** un nome che viene registrato con la registrazione di un host.
Il DNS server usa il protocollo **UDP** ed ascolta sulla **porta 53** poiché sono informazioni semplici (qual è l'indirizzo di questo ww.unicam.it? risposta semplice), mentre il trasferimento zona utilizzano il **TCP**.

Come funziona la richiesta?



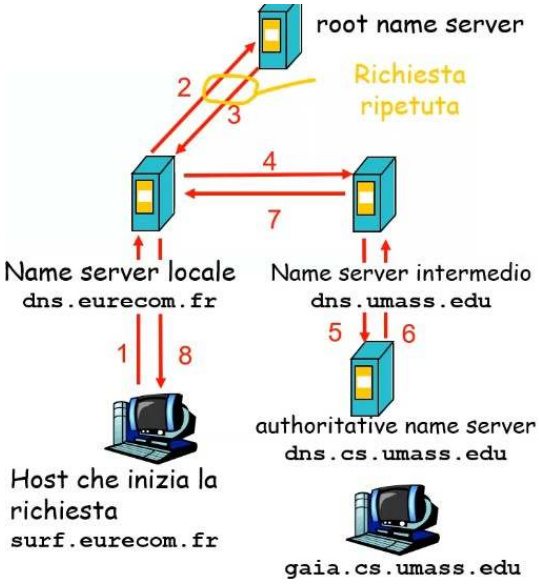


// Richiesta ricorsiva VS Richiesta ripetuta

Nel **primo caso** si trasferisce il carico della traduzione al name server di contatto.
Nel **secondo caso** il name server di contatto risponde con l'indirizzo del prossimo name server da contattare (non conosco questo nome, prova a chiedere all'altro)

Il DNS presenta anche una Cache: ogni volta che provo un indirizzo, ma se l'indirizzo già visitato ce l'ho nella cache senza scomodare il DNS. Presenta un timeout, dell'ordine di alcuni giorni, dopo la quale l'indirizzo viene cancellato.

*Ipconfig /displaydns per vedere il contenuto della cache
ipconfig /flushdns per cancellare la cache*



Com'è fatto il database del DNS?
Non è un database, ma sono dei file di zona. Dentro ci sono scritti come sono i record con formato **RR: (nome, valore, tipo, ttl)**. Esistono diversi tipi

Tipo A: nome e indirizzo dell'host
Tipo NS: nome e DNS secondario (nome del name server di riferimento)
Tipo MX: valore è il nome di un mailserver
Tipo CNAM: alias. Il computer che ha il tipo A si chiama anche così. (utile nel caso di più name server sulla stessa rete)

Protocollo DNS

Generalmente da una richiesta (query) e risposta (reply).

--> Protocolli di livello applicativo: SNMP (Simple Network Management Protocol)

È un protocollo usato diffusamente per gestire reti: **monitorare e configurare dispositivi di rete**.
Utilizza il modello Client- Server:

- **SNMP Station** (manager): è il modulo che emette una richiesta e che ha il suo **MIB**
- **SNMP Agent**: modulo che formula la risposta.

Ogni dispositivo in una rete gestita tramite SNMP, mantiene un database locale di informazioni per la gestione, noto come **Management Information Base** (MIB).
Consiste in poche e semplici operazioni: GET, GETNEXT("WALK"), GETBULK, SET, TRAP.

/* Capitolo 3: Transport Layer */

Un **protocollo di trasporto** è responsabile della comunicazione logica tra processi applicativi posti su diversi host. I due protocolli utilizzati in internet sono:
- UDP (User Datagram Protocol) -> servizio inaffidabile senza connessione
- TCP (Transmission Control Protocol) -> servizio affidabile orientato alla connessione, con controllo di congestione e di flusso

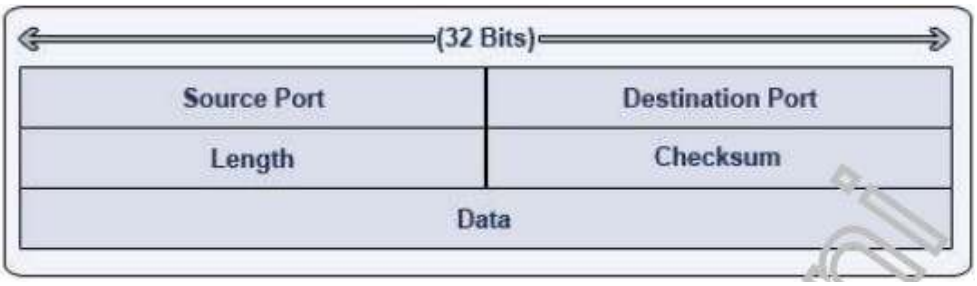
Lo scambio di informazioni in internet avviene tramite **pacchetti**: in particola parliamo di **segmenti**. Questi segmenti contengono generalmente 3 elementi e vengono inviati sulla rete come **Datagramma**:
- **# porta di origine/destinazione**
- altri campi di intestazione
- messaggio

Ogni processo ha un **socket**: identificazione unica per la trasmissione logica, porta attraverso la quale i dati passano dalla rete al processo e viceversa.
Questi socket possono avere funzioni di **demultiplexing** e/o **multiplexing**: il primo dal lato ricevente, che esami determinati campi del msg ricevuto e decide a quale socket del ricevente consegnarlo, e il secondo dal lato mittente, che raduna i diversi dati da diversi socket e li incapsula in un unico pacchetto da spedire in rete.

Sia il TCP che l'UDP fanno MPX e DMPXing.

--> Protocolli di trasporto: UDP (User Datagram Protocol)

L'UDP è un protocollo connection-less, che non fa l'handshake, ed è quindi più veloce del TCP poiché riduce notevolmente la congestione della rete. Inoltre presenta un'intestazione della dimensione di **8byte** contro i 20 del TCP.



I campi speciali dell'intestazione del segmento indicano il socket a cui consegnare il processo --> **Source** Port & **Destination** Port, entrambe con 16bit (0-65535).
Presenta poi il campo **lenght**, che mi definisce il messaggio, e il **checksum**: in particolare calcola la somma di tutte le parole da 2 byte contenute nel messaggio, fa il complemento ad 1 della somma e poi invia il risultato nel segmento UDP al ricevente. Il ricevente calcola la somma di tutte le parole del **contenuto nel segmento + il checksum**. Se non si sono verificati errori il risultato deve essere uguale ad una sequenza di 1.

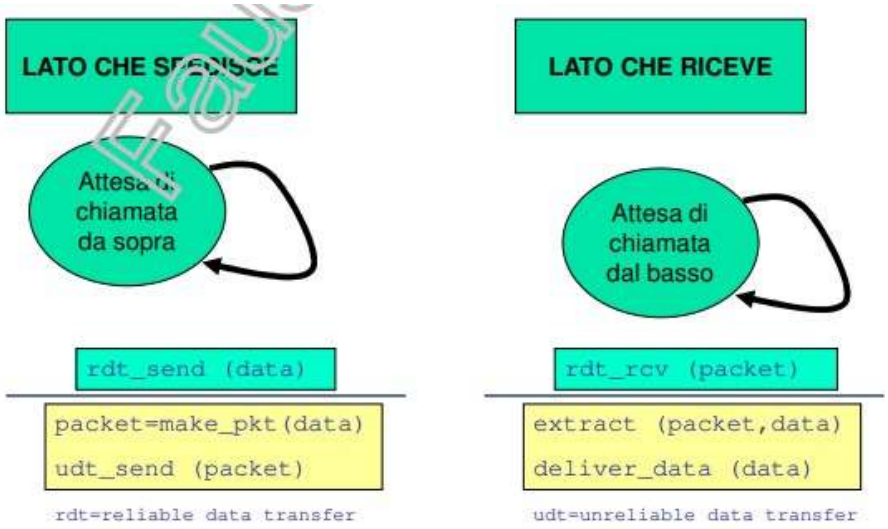
Nella maggior parte degli utilizzi nell'**application layer** viene usato il TCP, poiché fornisce molti servizi legati alla stabilità e all'affidabilità della connessione. Mentre l'UDP viene usato

prevalentemente dove è richiesta **responsiveness** (attesa molto breve9 e che permette una perdita di pacchetti che non compromette la connessione --> streaming video

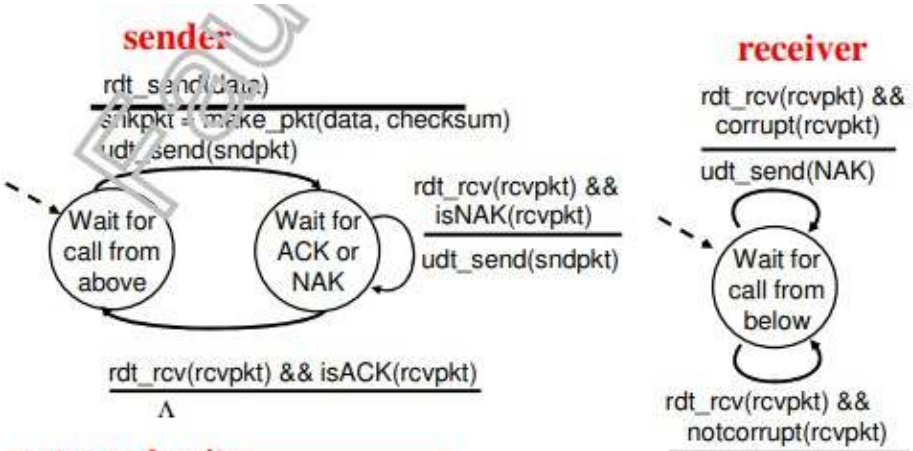
// Principi del trasferimento dati affidabile – (Reliable Data Transfer)

Spesso anche se i protocolli adottati sono sicuri e affidabili, i livelli sottostanti su cui si appoggiano potrebbero non essere affidabili. Ci sono quindi 4 modi (circa) di risolvere il problema

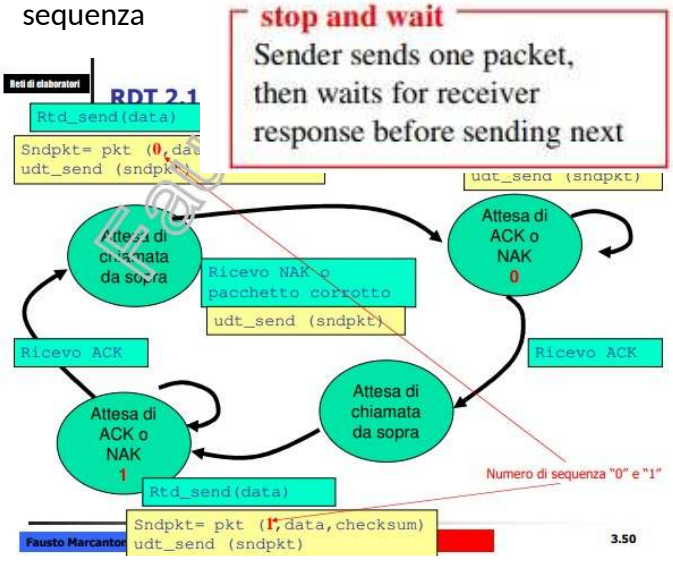
- **RDT 1.0** – Il canale sottostante **completamente affidabile** ……

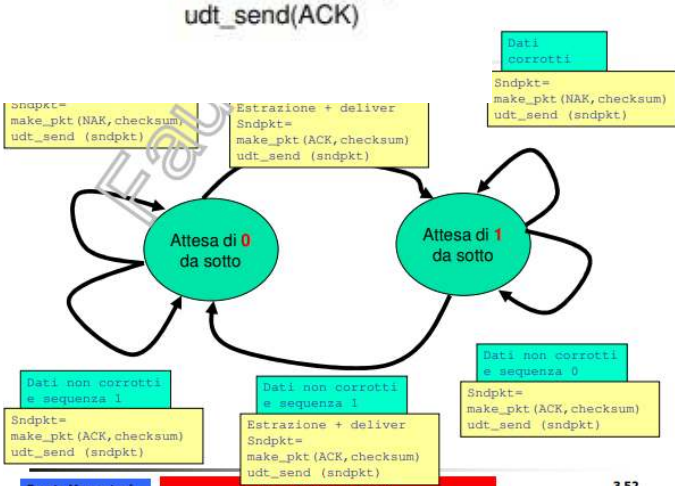


- RDT 2.0 ……



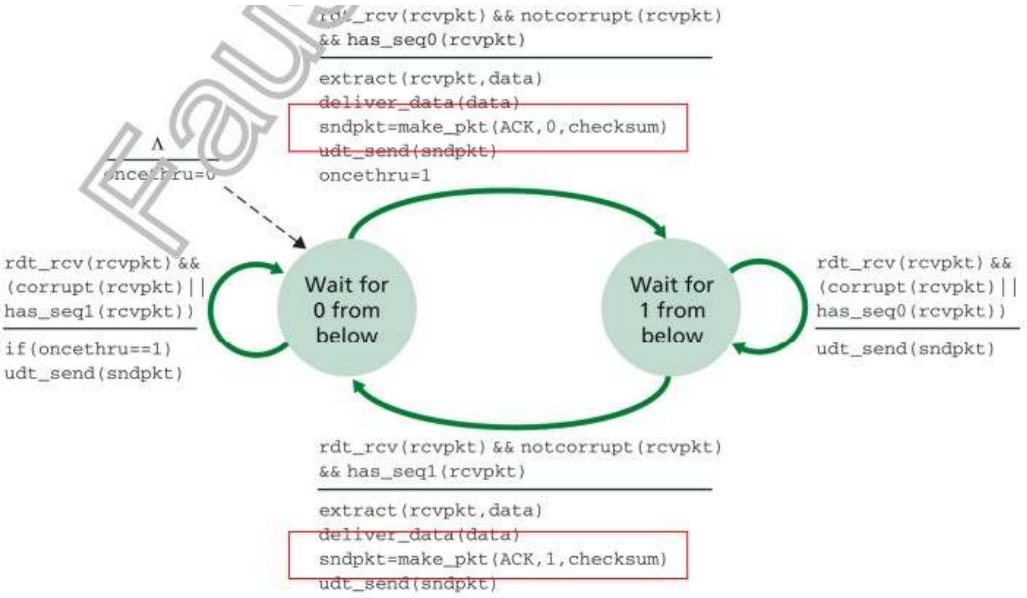
- **RDT 2.1** - Numero di sequenza



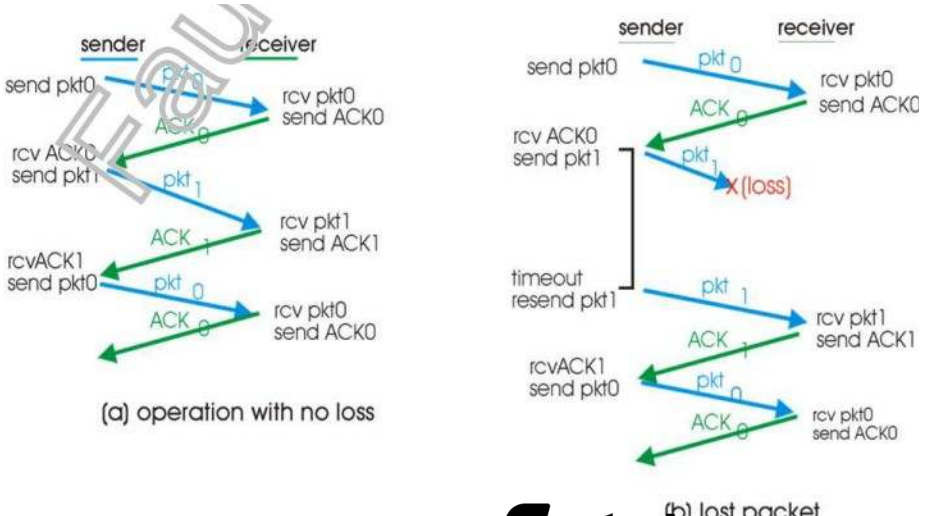


inserito nel pacchetto dati ……
Il ricevente invia feedback di error detection-feedback-retransmission.

- **RDT 2.2** – Inclusione seq number nel pacchetto di un messaggio ACK ……

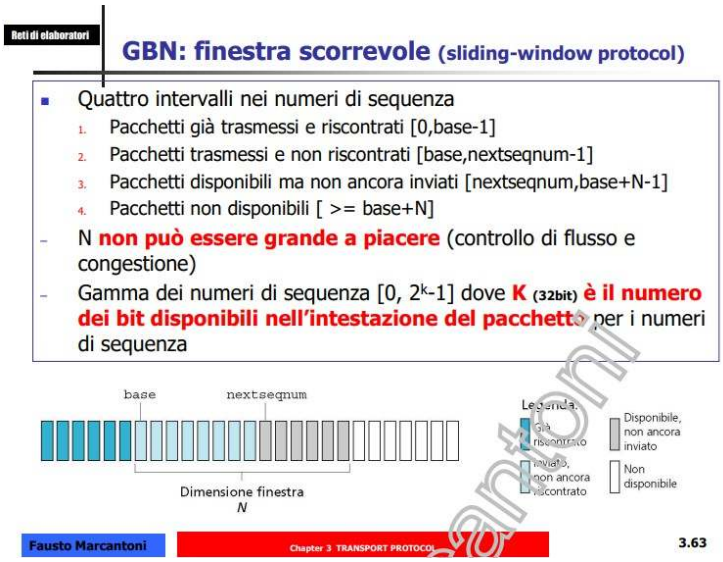


- **RDT 3.0** – Introduzione TimeOut ……



// Protocollo di Pipelining & Go-Back-N

Perché inviare un solo pacchetto per volta? Inefficiente dal punto di vista del tempo impiegato nello scambio di informazioni --> Introduzione della finestra scorrevole



Presenta il **riscontro cumulativo**: un riscontro di un panchetto N implica l'ACK anche di tutti quelli precedenti ("fino a qui ho capito tutto"). In caso di **timeout**, il mittente rispedisce tutti i pacchetti di tipo 2 inviati ma non confermati dall'ACK.

* http://computerscience.unicam.it/marcantoni/reti/applet/GoBackProtocol/goback.html *

// Selected Repeat Protocol

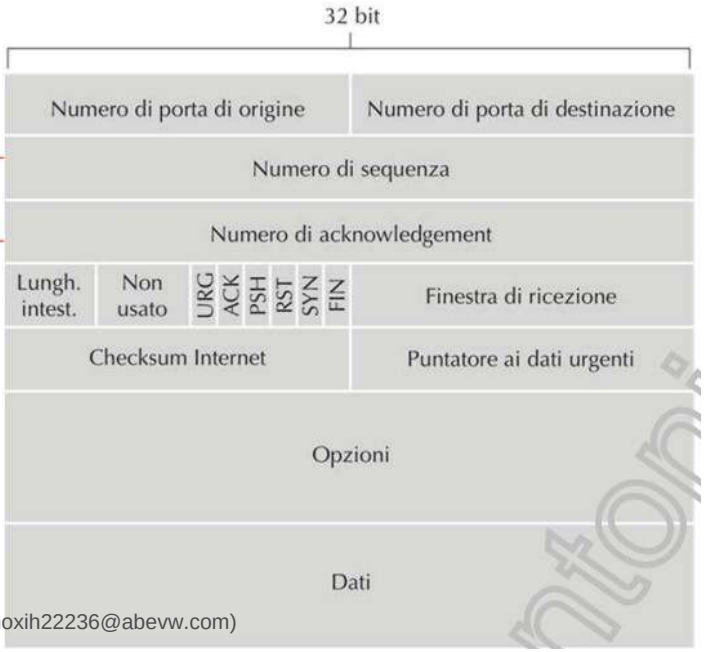
Il selective repeat non fa altro che ritrasmettere **solo** i pacchetti dei quali **non ho ricevuto l'ACK** e non tutto quanto.

* https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/ *

--> Protocolli di trasporto: TCP (Transfer Control Protocol)

Il protocollo TCP è **connection-oriented**: ciò prevede che si faccia un handshake prima di iniziare lo scambio di informazioni. Inoltre è **point-to-point**: una configurazione che prevede un solo sender e un solo receiver.

Il TCP ha un **buffer di recezione** e quindi si necessita la specifica del **MSS**: è il limite di dati



che possono essere trasferiti in un segmento TCP. Questo viene poi "aggiustato" dal **MTU**: il Max Transmission Unit che è la dimensione massima dei dati che possono essere trasmessi nel link layer (ethernet ha MTU 1500byte).

Presenta **6 flags**, bit che possono valere 0 o 1:
- **RST**: bit di reset per gravi errori
- **PSH**: il ricevente deve passare subito il segmento al layer sopra
- **URG**: marca il contenuto del segmento come urgente
- **FIN & SYN**: si usano per chiudere e aprire connessioni.

Inoltre vengono istanziate altre variabili:
- **ISN**: è il Initial Sequence Number, un numero generato in modo pseudocasuale all'avvio della connessione, compreso tra 0 e $2^{32} - 1$
- **MSL**: è il Max Segment Lifetime, ovvero il tempo per cui il segmento resta in rete.

// Procedure di timeout e ritrasmissione

Il timeout, in questo caso, deve essere maggiore del RTT (Round Trip Time, tempo richiesto affinché il segmento faccia andata e ritorno).
L'RTT non si può calcolare né tantomeno stabilire a priori, ma lo si può approssimare:

$$\textbf{estimatedRTT} = (1-\alpha)\ \textbf{estimatedRTT} + \alpha\ \textbf{sampleRTT}$$

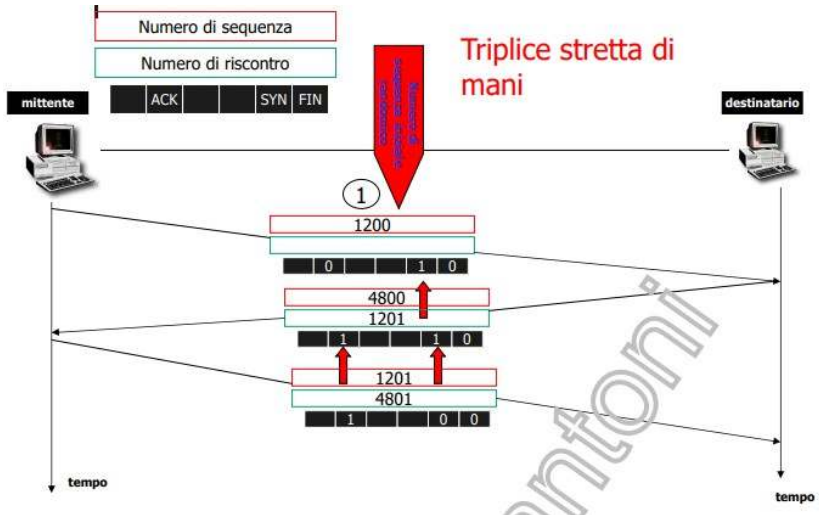
&

$$\textbf{devRTT} = (1-\beta)\ \textbf{devRTT} + \beta\ (\textbf{sampleRTT} - \textbf{estimatedRTT})$$

dove l'estimatedRTT è la media pesata dei valori misurati, il sampleRTT è quello misurato ad ogni andata/ritorno, α=0,125 & β=0,25.

Il **Retransmission Time Out (RTO)** è il tempo entro cui la sorgente si aspetta di ricevere un riscontro. Anche questo non può essere un valore statico predefinito, poiché dipende da diversi fattori: infatti, si calcola dinamicamente di volta in volta e deve essere maggiore del RTT.
In generale **RTO = estimatedRTT + 4devRTT**, con un risultato tra 200ms e 60sec. (PAG 21)

// 3-way handshake in TCP (apertura connessione)



- Il mittente invia il SYN seq=x;
- il destinatario riceve il SYN segment e invia il SYN seq=y con l'ACK=x+1;
- il mittente riceve il SYN+ACK segment e invia l'ACK=y+1;
- il destinario riceve l'ACK segment.

Per chiudere la connessione si utilizza lo stesso metodo utilizzando, però, la flag **FIN** (4-way handshake).

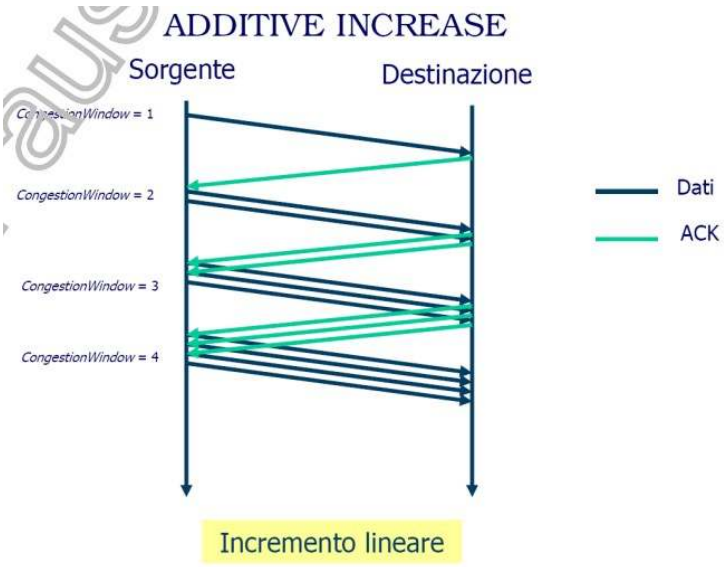
// Controllo congestione

Il controllo della congestione della rete, diverso dal flusso di rete, non deve essere effettuato attraverso l'immissione di altrettanti "pacchetti di controllo" all'interno della rete. Ci sono diversi scenari che si possono incontrare nella realtà, che tengono conto anche delle prestazioni della rete (vedi slide).

Nel TCP il mittente limita il ritmo di trasmissione tenendo conto della **finestra di congestione**: ovvero la quantità di dati riscontranti da un host durante la connessione. In particolare, il TCP non può inviare dati ad un rate **maggiore** della finestra della congestione: questa cambia **dinamicamente** proprio in base alla congestione della connessione.

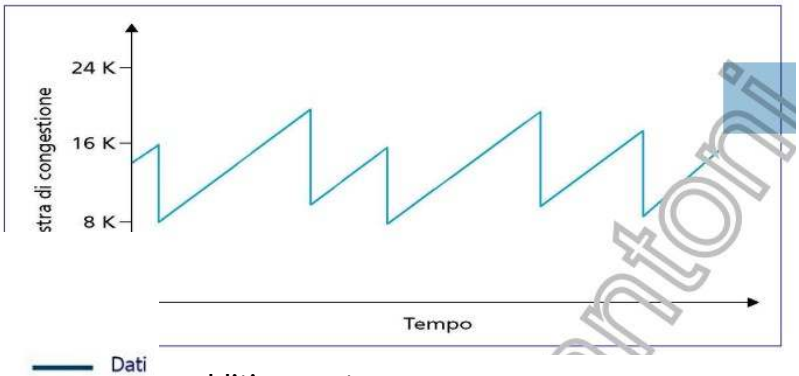
Questo cambiamento viene percepito dal mittente in base agli eventi di perdita (timeout/ACK duplicati), che utilizza diversi algoritmi per cambiare dinamicamente suddetta finestra:

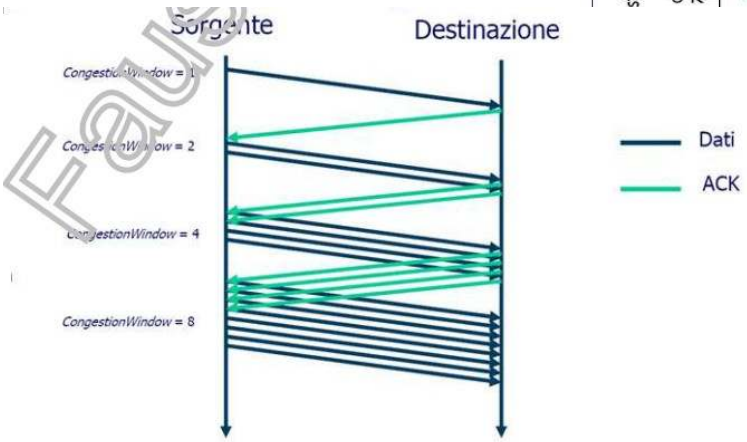
1) Congestion Avoidance (AIMD)
2) Partenza Lenta (Slow Start)
3) Fast recovery (non trattata)



Sia la AIMD che la SLOW START tengono conto di due variabili importanti:
- **Congwin**: la finestra di congestione corrente
- **Threshold**: il limite tra incremento moltiplicativo e incremento additivo

In particolare al di **sotto** della threshold si adotta la **slow start**, dove la Congwin cresce geometricamente; al di **sopra** della threshold si adotta la **congestion avoidance**, dove la Congwin cresce additivamente.





Il decremento avviene dimezzando l'attuale finestra di congestione ad ogni riscontro di perdita.

Nella Slow Start, la dimensione del CongWin viene prudentemente

CongestionWindow = CongestionWindow * 2
All'arrivo di ogni ACK

impostata ad **1** all'inizio della connessione. Questa viene poi incrementata in modo esponenziale ad ogni arrivo di un ACK.

Abbiamo quindi due tipi di comportamenti diversi, per lo stesso fenomeno di perdita. Vengono considerati quindi il **TCP TAHOE** e il **TCP RENO**: il primo taglia incondizionatamente la finestra di congestione ad 1 MSS ed entra nella fase di SLOW START sempre; mentre il TCP RENO elimina la fasi di partenza lenta ed effettua un ripristino rapido (fast recovery).

/* Capitolo 4: Network Layer */

Nasce per la necessita di collegare dalla necessità di collegare e comunicare tra di loro reti di calcolatori (come AS): alla base del collegamento, ci sono i Router. Questi possono essere **interior router e exterior router** (interni/di frontiera) e devono in qualche modo parlare allo stesso modo, utilizzando protocolli adatti.

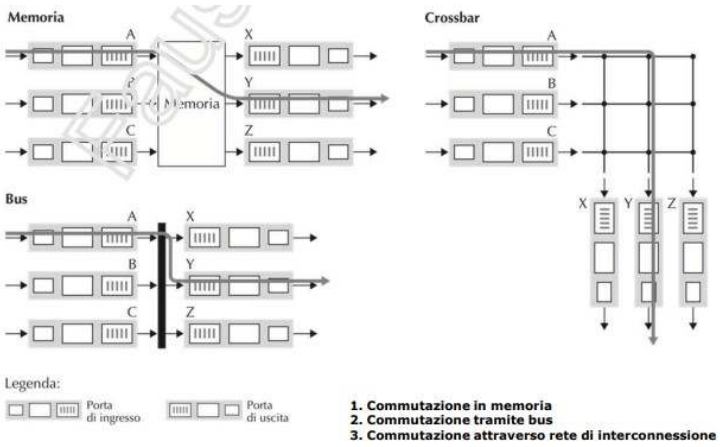
// Router

Il router è un dispositivo che lavora al **livello 3** (livello di rete) e non è altro che un **commutatore**: prende i pacchetti che arrivano e, attraverso l'algoritmo di routing, trova la porta di destinazione per mandarlo al next stop. Si comporta in **2 modi** diversi:
- in alcuni casi **analizza l'indirizzo logico**, che riesce a sapere se l'indirizzo è della propria rete o no, in caso non fosse della rete locale fa l'inoltro
- sa anche il **next hop**, il passo successivo dove inviare il pacchetto. In base al protocollo il router sa o tutta la topologia di rete o solo i suoi vicini.

All'interno troviamo: **porte ingresso, porte di uscita, il commutatore e il processore di instradamento**. L'implementazione della funzione di instradamento viene oramai quasi sempre in modalità hardware per renderla più efficiente possibile.

- **Porte in ingresso**: buffer che, in base alle tabelle di inoltro, determina la porta di uscita.
- Prefix length: parametro utilizzato per la ricerca nella tabella --> **Prefix Matching**, processo con cui sui si confronta un indirizzo IP con un insieme di indirizzi di rete.
- **Commuttore**: funziona in 3 modi diversi: tramite **memoria**, tramite un **bus**, tramite un **commutatore**. In base alla scelta della modalità cambia la velocità e il dispendio di risorse.

Una volta arrivato alla **porta di uscita** c'è l'accodamento, anche se questo può portare a ritardi e quindi a perdite di pacchetti. Si possono instaurare delle policy per impostare la priorità e la velocità di instradamento: spesso viene considerata la regola "chi prima arriva, prima esce" (FIFO).

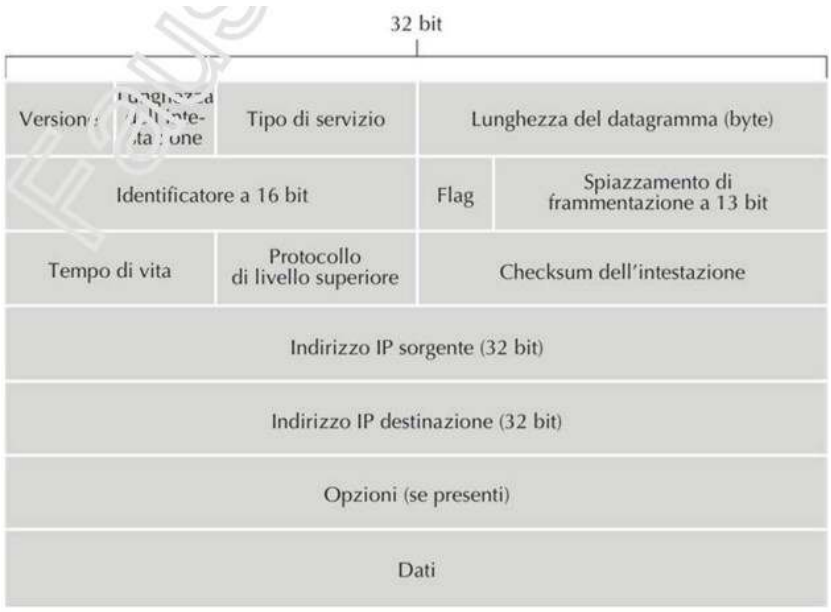


--> IP – Internet Protocol

L'**Internet Protocol** è nato proprio per la comunicazione, con l'obbiettivo di consentire l'instradamento dei pacchetti in rete. Questo indirizzamento è **universale**: infatti non esistono 2 host che hanno lo stesso indirizzo IP (su internet)

L'IP trasferisce un **datagramma**, non più un segmento, e cerca il **best effort** dei pacchetti, cercando di portare a destinazione il pacchetto e tenerlo il meno possibile in rete. Inoltre può **frammentare** i pacchetti se richiesto, per una migliore efficienza, che verranno poi ricomposti dall'host ricevente.

// Formato dei datagrammi IPv4



Numero di versione: 4 bit che specifica come deve essere trattano il datagramma
Lunghezza intestazione: 20 byte
Tipo Servizio: 8 bit dove posso mettere identificatore che specifica la priorità di come trattare il pacchetto
Lunghezza datagramma: 16bit che indica quanto è lungo tutto il datagram (compreso il payload)
Identificatore (offset di frammentazione): 16+3+13 bit utili per la frammentazione
Tempo di vita (TTL): campo che mi consente per il timeout del pacchetto, ogni volta che viene attraversato da un router questo viene decrementato di 1 e, se 0, viene dichiarato morto.
Protocollo di livello superiore: protocollo con cui aprire il payload per leggere le informazioni (6 per TCP, 17 per UDP)
Checksum: solito campo per il controllo del raggiungimento della destinazione
Indirizzo IP sorgente e destinazione: entrambi da 32 bit, scritti in binario
Opzioni & Payload dati

// Schema di indirizzamento

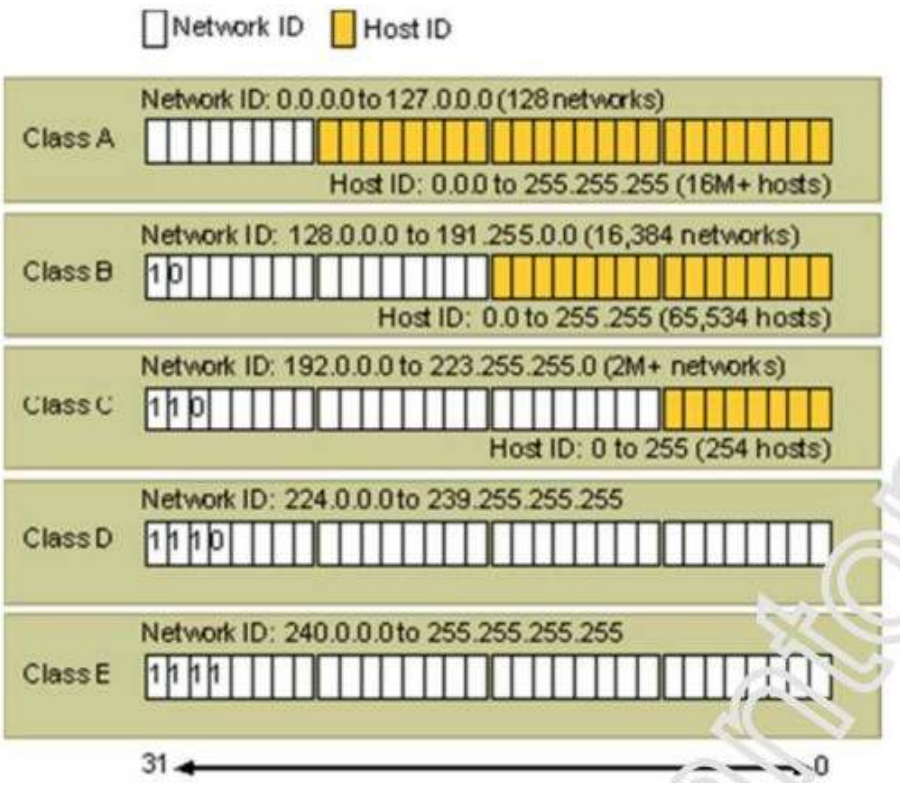
Gli indirizzi IP sono unici su tutta la rete e all'inizio ogni indirizzo era associato ad un solo ed unico host. L'indirizzo IP è fatto dalla **Net.ID + Host_ID**.
La notazione è ovviamente binaria, fata da 4 gruppi di 8 bit che indicano i 4 numeri decimali che identificano l'indirizzo IP.

L'**IPv6** invece è composto da **128bit** e sono coppie di ottetti in esadecimale. Il formato del datagramma differisce di molto da quello dell'IPv4, in quanto diversi campi sono stati eliminati e parte della sicurezza è insita nell'indirizzo.

Interfaccia di rete: punto di connessione tra host e router, si parla di parte fisica. I **router ha** più di un'interfaccia di rete, **l'host può** avere più di un'interfaccia.

// Evoluzione degli Indirizzamenti

1981 --> Indirizzamento Classfull (Archeologia!!!)
Questo indirizzamento è facile da comprendere e implementare: si basa sulla suddivisione in 5 classi degli indirizzi basate sui primi 4 bit dell'indirizzo IP.



(!) Questo può capitare all'esame: basta rircordarsi i numeri (0, 128, 192, 224, 240) ed essere veloci nella conversione decimale-binario.

1993 --> CIDR (Classless Inter-Domain Routing)
Dato che l'indirizzamento Classful è troppo rigido, si abbandonano le classi.
Gli indirizzi sono gestiti in modo efficiente per fare rounting: l'indirizzo IP diventa

IP = <prefisso, suffisso>

dove il prefisso è la **rete** di appartenenza e il suffisso è **l'host** connesso.

// NetMask (Maschera di rete)

Nasce dalla necessita (vedi sopra) di dividere la parte di rete dalla parte host nell'indirizzo. Sono **32bit** disposti con x "1" seguiti da y "0".

Operando poi un **AND logico** possiamo estrarre la parte di rete da quella host.

255.255.255.0
11111111.11111111.11111111.00000000
x.y.x.0/24

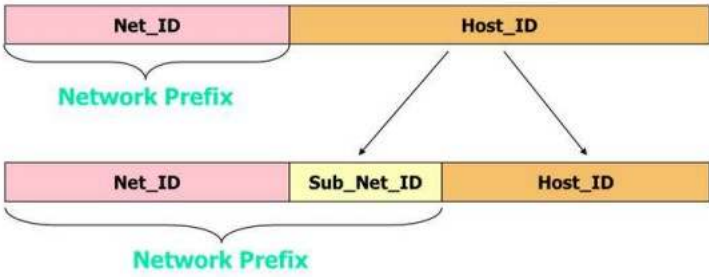
// Indirizzi IP particolari

- **0.0.0.0** --> Indirizzo di avvio
- **127.0.0.1** --> Loopback (localhost) che permette di comunicare con la propria macchina come se fosse un altro host nella rete
- **Net_ID.(tutti 1 in Host_ID)** --> indirizzo broadcast, mando pacchetti a tutta al rete Net_ID.

- **Net_ID.(tutti 0 in Host_ID)** --> indirizzo di rete, indica la rete
- **255.255.255.255 (tutti 1)** --> broadcast locale

// Subnetting

Il subnetting è la tecnica di suddividere la rete raggruppando TOT indirizzi host formando logicamente una sottorete.



Esistono **2** tipi di subnetting:
- **maschera Fissa** (Fixed Length Subnet Mask), in caso di subnetting statico
- **maschera Variabile** (Variable-length Subnet Mask), in caso di subnetting dinamico

ESEMPIO

Risaliamo alla rete dato l'indirizzo

193.205.92.150 /25

con **/25** indichiamo la subnet mask, che presenta appunto 25 "1" e, per differenza, 7 "0".

Procediamo quindi con l'AND logico:

indirizzo	11000001.11001101.01011100.10010110
mask	11111111.11111111.11111111.10000000
AND	11000001.11001101.01011100.10000000
in dec = rete	**193.205.92.128**

// IP e la frammentazione dei pacchetti

Nell'header IP, ricordiamo i campi:
- **Protocol**: sono 8 bit che è la chiave per poter leggere il payload, poiché specifica di che protocollo di tratta e come leggere i dati
- **Checksum Header**: sono 16 bit che servono al controllo di errori attraverso il parity check, attraverso il completamento a 1.
- **il secondo gruppo di 32bit**: questo è riservato alla frammentazione, identificazione, flag e fragment offset.

La frammentazione è quel processo di suddivisione di un frame in ulteriori frame della grandezza del mezzo di trasmissione sottostante (definito dal Max Transmission Unit). Il suo riassemblaggio viene eseguito solo una volta a destinazione.

// Campi di frammentazione

Identificazione	16 bit	Individua univocamente il frammento Tutti i frammenti hanno lo stesso ID number
Flag	3 bit	Primo bit riservato Secondo bit = 1 non si può frammentare (messaggio di errore ICMP) Terzo bit = 0 frammento è l'ultimo del datagramma o il solo
Offset di frammentazione	13 bit	Fornisce la posizione del frammento nel datagramma originario **misurata in unità di 8 byte**. (La dimensione del payload di ogni frammento è un multiplo di 8 byte)

//

Problematiche

- Maggiore **overhead** di trasmissione con l'impiego di molte risorse non necessarie
- Soggetto facilmente attaccabile tramite **Denial Of Service** (DOS)

--> DHCP – Dynamic Host Configuration Protocol

Gli indirizzi IP privati possono essere usati fa chiunque, ma non devono oltrepassare il router di frontiera. Quelli privati vengono dati all'amministratore che poi li passa agli host nella rete.
I problemi più comuni nella gestione di una rete sono: duplicazione, renumbering, scarsità e gestione degli indirizzi IP.

Quando serve usare DHCP? Quando ci sono molti client, ci sono portatili/mobile...
I vantaggi sono quelli di fornire settaggi **automaticamente** e in modo che ogni client appena collegato è **subito operativo**.

// BOOTP & DHCP

- **BOOTP**: meno recente che sfruttava il disco e il boot up della macchina per definire in maniera automatica il proprio indirizzo IP. Era usato essenzialmente nel settore diskless, per scaricare immagini di memoria per stazioni di lavoro da un server proprio attraverso il BOOTP.

- **DHCP**: la differenza è che l'IP è soggetto a un lease, cioè un prestito dell'ip per un periodo di tempo limitato.
Le informazioni nel DHCP sono: indirizzo IP, Subnet mask, Gateway, DNS, altri valori opzionali.

È costituito da una richiesta e da una risposta e può lavorare in **3 modi**:
- **assegnazione manuale**: l'IP viene assegnato dall'amministratore per una corrispondenza univoca tra indirizzo MAC e indirizzo IP --> **reservation**. I vantaggi sono pochi, essenzialmente il maggiore controllo sulla rete.
- **assegnazione automatica**: nel momento in cui accendo un client, ricevono sempre lo stesso indirizzo IP dal server DHCP, quindi essenzialmente con un **lease time infinito**.
- **assegnazione dinamica**: all'IP viene dato un lease time (circa 1h) che una volta scaduto o gli viene riassegnato lo stesso o un altro in base alla disponibilità.

// 4 (+1) fasi di query

- **DHCP discover**: manda un broadcast alla porta 255.255.255.0:67 con indirizzo 0.0.0.0:68; viene fatta in broadcast per cercare più server DHCP per ricevere l'indirizzo (utilizzano protocollo UDP).
- **DHCP offer**: offerta di indirizzo dal server DHCP, che possono essere diversi con diverse offerte
- **DHCP request**: il client richiede l'indirizzo al primo che risponde
- **DHCP ack**: conferma dell'indirizzo da parte del server
- **DHCP release**: procedure di shutdown, dove viene rilasciato al server l'indirizzo IP allo spegnimento del Client.

Se non c'è un indirizzo IP disponibile un sistema windows ha indirizzo **169.254.0.0/16**.

DHCP relay
È un DHCP che non sta nella rete dove sono collegato: sono i router a cui sono collegato, che poi fanno la richiesta al DCHP relay per l'assegnazione dell'IP.

--> LAN – Local Area Network

Per definizione *IEEE*: una LAN è un sistema di comunicazione che permette ad apparecchiature indipendenti di comunicare tra loro entro un'**area delimitata** utilizzando un canale fisico a **velocità elevata** e con **basso tasso di errore**.

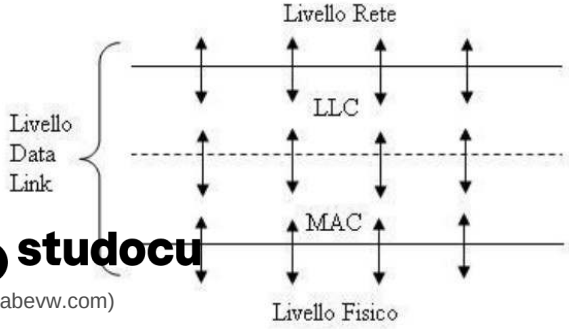
Alcune **caratteristiche**:
- generalmente non sono trasmissioni di dati continue
- tutte le macchine utilizzano lo stesso canale trasmissivo
- la trasmissione è di tipo Broadcast
- è economica, versatile nella modifica, facile nella manutenzione, capace di grossi carichi, duratura nel tempo

Alcune **complicazioni**:
- tutti gli host devono essere identificati (quindi indirizzati)
- dato il canale condiviso, occorre arbitrare l'accesso al suddetto canale

Attributi di una LAN:
- affidabilità
- flessibilità
- modularità
- espandibilità
- gestibilità (protocollo SNMP)

La **classificazione** delle LAN avviene sulla base:
- **Hardware**: il tipo di architettura, come topologia, mezzo trasmissivo, tecniche di trasmissione e metodi di accesso alla rete
- **Software**: il sistema operativo che definisce i protocolli di rete, cioè lo standard di comunicazione

// Data Link Control



È l'ultimo passo prima della trasmissione ed è quello di struttura del messaggio, secondo il formato previsto dal protocollo utilizzato sulla linea di uscita.

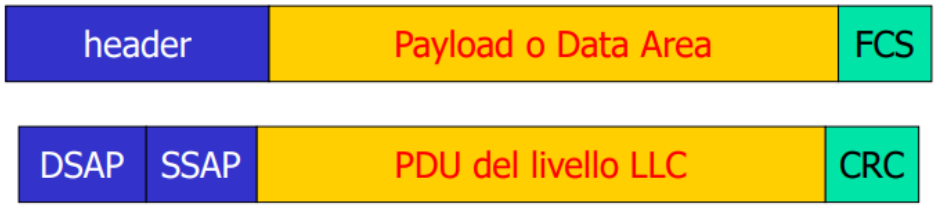
In **trasmissione**, riceve un flusso di dati e lo trasforma in frame, aggiunge eventuali intestazioni ed esegue l'algoritmo di controllo errore, lo passa al livello MAC e lo invia.
In **recezione**, prende il flusso in arrivo, controllo l'intestazione esegue il controllo dell'errore, elimina intestazione e end.frame e lo invia al livello superiore.

Inoltre il layer datalink si divide in:
- **LLC** (Logical Link Control): supporta servizi di alto livello, tipo gestione errori e controllo di flusso
- **MAC** (Medium Access Control): che si occupa della gestione dell'accesso al mezzo fisico inferiore

// MAC PDU - Frame

Come faccio a mettere in comunicazione diretta due calcolatori senza disturbare gli altri, che comunque ricevono una copia di tutti i dati?

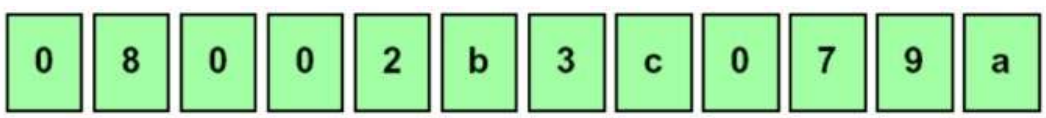
Devo assegnare a ciascun host un numero identificativo --> **MAC** (**M**edia **AC**cess address)



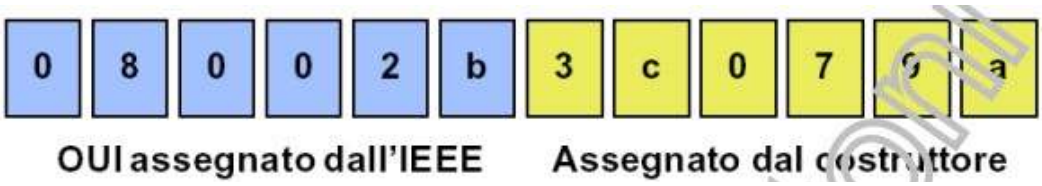
I campi principali di una MAC PDU sono:
- **DSAP e SSAP**: destination/source SAP (Service Access Point)
- **Payload** dei dati
- La **FCS** (Frame Check Sequence): è un **CRC** (Cyclic Redundancy Code) su 32 bit per il controllo integrità

// Indirizzo MAC

Sono lunghi **6 byte**, cioè 48bit e sono formattati come **6 coppie di cifre esadecimali**.

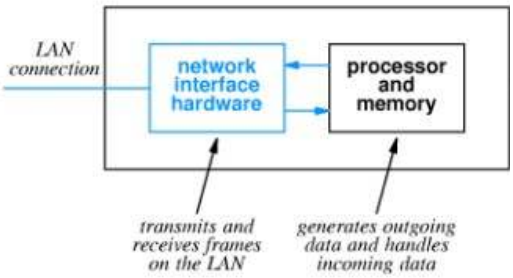


La sua struttura si suddivide in due parti grandi 3 Byte ciascuna:
- i 3 byte più significativi sono detti **OUI** (Organization Unique Identifier): detto anche vendor code, sono i codici associati ai produttori di schede di rete e sono standardizzati dall'IEEE
- i 3 byte meno significativi sono una **numerazione progressiva** decisa dai produttori



// Scheda di rete

La scheda di rete si occupa della trasmissione e ricezione dei dati e lavora indipendentemente dal computer.



L'indirizzo fisico deve essere **UNICO** nella LAN (non in internet). Questo normalmente è **statico**, cioè assegnato dal costruttore, ma può essere anche configurabile o dinamico.

Gli indirizzi sono di 3 tipi:
- **Unicast** = singolo host
- **Multicast** = gruppo di host
- **Broadcast** = tutte le stazioni (ff:ff:ff:ff:ff:ff)

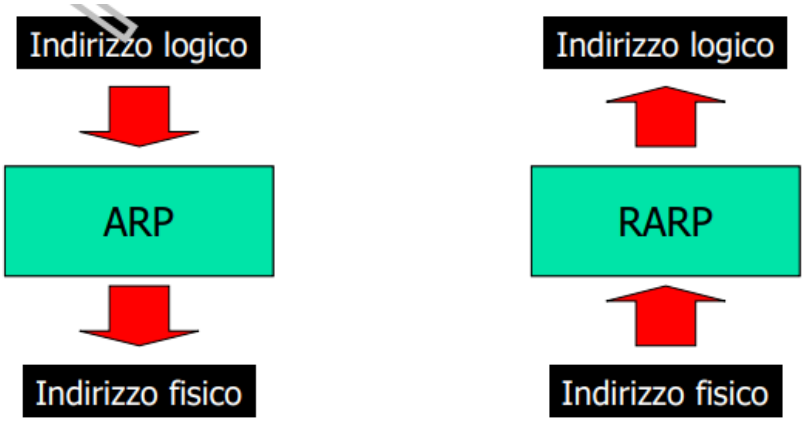
// Indirizzi di gruppo

Sono indirizzi che servono principalmente a fare **Neighbor Discovery** e hanno 2 modi di impiego:
- **Solicitation**: la stazione richiede un servizio e manda un mesaggio multicast con l'indirizzo del servizio; le stazioni che offrono tale servizio rispondono
- **Discovery** (o Advertisement): la stazioni che offrono un servizio inviano a cadenza regolare un messaggio multicast per informare del servizio offerto; chi è interessato al servizio risponde

NDP (Neighbor Discover Protocol): è un protocollo utilizzato e configurabile per gli indirizzi IPv6

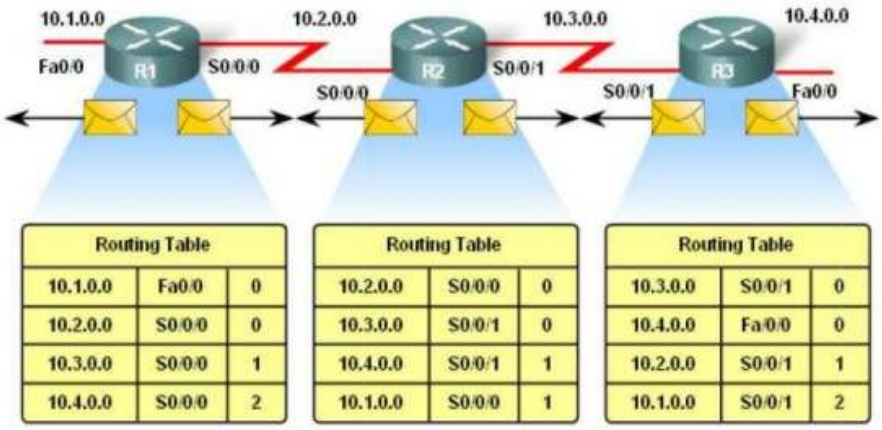
// ARP - RARP

L'ARP (Address Resolution Protocol) è un protocollo incorporato dalla famiglia TCP/IP che è responsabile della **traduzione** dell'indirizzo IP in indirizzo fisico (MAC) e il RARP viceversa.



--> Routing & Instradamento

L'obbiettivo del routing è quello di fare in modo che i pacchetti arrivino a destinazione. Per fare ciò vengono introdotte le **tabelle di instradamento**: la tabella è un "database" memorizzato in un router o in un host, che elenca le rotte di destinazione di una data rete e in molti casi una **metrica** di tale rotta. La metrica identifica di solito il costo associato ad una determinata route in termini di tempo, per capire quali di queste convengano o meno.



I record nelle tabelle sono composti da:
- **indirizzo** della rete di destinazione
- **netmask**
- **interfaccia** su cui inoltrare
- indirizzo del **next hop**

Il routing può avvenire in **2 modi**:
- **Diretto** --> se l'host mittente è sulla stessa rete dell'host destinatario. Di solito coinvolge i livelli 1 e 2 o, più in generale l'hardware address (MAC address)
- **Indiretto** --> se gli estremi della comunicazione di trovano su reti differenti. Dovrò passare in un router intermedio e utilizzero l'indirizzo IP (coinvolge layer 1,2 e 3).

+ Come faccio a sapere se due host si trovano sulla stessa rete? Attraverso gli indirizzi IP e il Network Prefix Match (AND logico tra indirizzo e maschera).

+ La tabella di routing è **sempre** presente in tutte le macchine con IP: di solito contiene almeno il next hop migliore

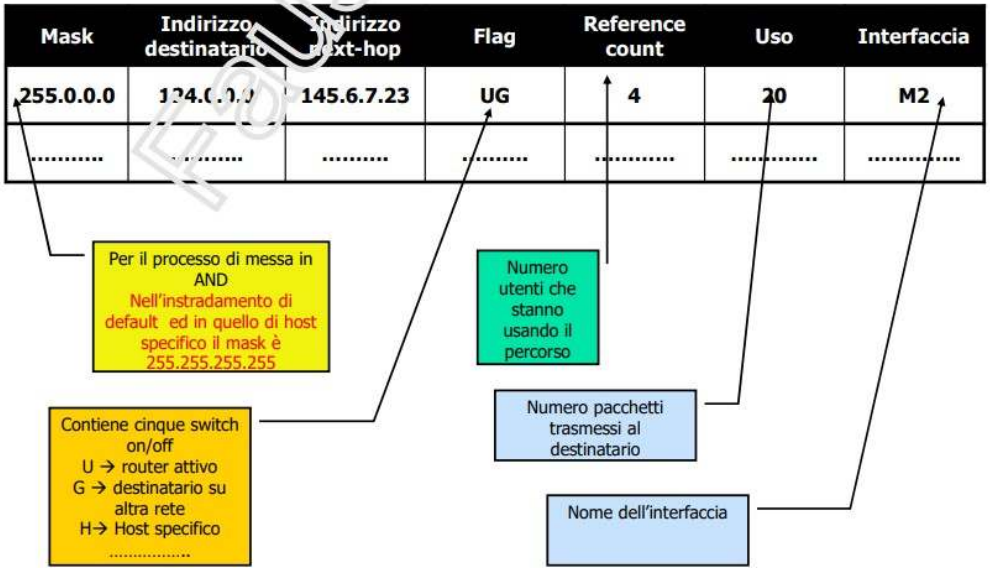
+ Il next hop è configurato in una sola direzione, i percorsi sono tipicamente **asimmetrici**.

// Rotte

Esistono 3 tipi di rotte:
- **Diretta**: address range corrispondenti alle interfacce router
- **Statica**: route configurate staticamente dall'admin
- **Dinamica**: reperite tramiteun protocollo di routing

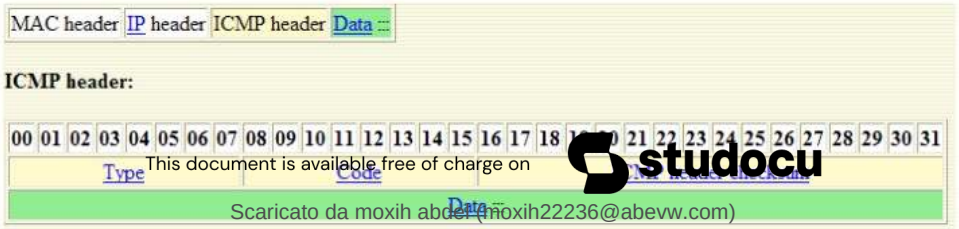
Negli end system e in gran parte dei router deve essere **sempre** presente una **default route**, quando non si sa dove andare.

La tabella si costruisce nel seguente modo:



--> ICMP – Internet Control Message Protocol

Questo protocollo è implicito nel TCP/IP e permette di inviare messaggi di errore. Infatti i 2 protocolli sono



interdipendenti: l'IP utilizza ICMP per mandare messaggi di errore e l'ICMP utilizza IP per trasportare i suoi messaggi.

L'header dell'ICMP è di **32bit** e contiene:
- **Type**: valore numerico che specifica il formato dell'informazione
- **Code**: specifica di errore
- **Checksum**: classico controllo con complemento a 1

I messaggi contenuti che possono essere trasferiti attraverso l'ICMP sono:
- **messaggi di errore**: destinatario irraggiungibile, mittente rallentato, limite di durata superato, reindirizzamento
- **richiesta di informazioni**: richiesta di eco e eco di risposta, richiesta e risposta timestamp, richiesta di mask e risposta, sollecito e notifica di router

//Caso traceroute

L'ICMP viene usato anche per fare **Traceroute**: si tratta di inviare sequenzialmente una serie di datagrammi e attendendo una risposta da essi.

1) Al primo datagram, il traceroute setta **TTL = 1** del datagram, in modo che giunto al router, questo scarti il datagram, decremento il contatore TTL e invia un **ICMP di time exceeded**.

2) Dopo aver scoperto l'indirizzo del router successivo, il traceroute imposta il **TTL = 2** in modo da raggiungere il 2° router. Questo viene decrementato dal primo e, una volta arrivato al secondo, diventa 0 e viene inviata di nuovo una ICMP di exceeded time.

Così facendo riusciamo a captare informazioni su tutti i router del percorso.

--> Autonomous System

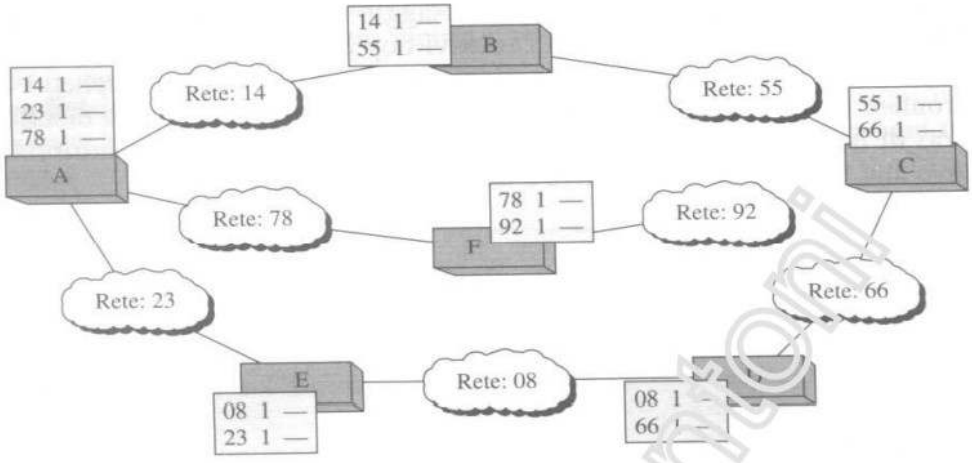
L'internetwork è in realtà un'unione di più reti, che possono essere gestite in maniera indipendente: questo ha portato a suddividere l'internetwork in aree diverse, ognuna delle quali si trova sotto uno stesso dominio amministrativo --> **Autonomous System**.

Questi sistemi autonomi possono essere collegati tra di loro e ognuno di loro presenta degli **Interior Gateway Protocol** (IGP) e degli **Exterior Gateway Protocol** (EGP).
Sarà compito degli **ASBR** (Autonomoues System Border Router) far comunicare l'interno dell'AS con gli altri AS e più in generale con il resto della rete.

Esistono quindi 2 tipi di protocolli di routing:
--> **Intra-AS (IGP)**, che si occupano di instradare datagrammi tra nodi dello stesso sistema:
- **distance vector**: RIP/RIP-2 & IGRP/EIGRP
- **link state**: OSPF/OSPF-2 & IS-IS

--> **Inter-AS (EGP)**, che si occupano di instradare datagrammi tra sistemi:
- **BGP** Border Gateway Protocol (attualmente usato)
- **EGP** (formalmente usato)

* i protocolli link state trovano il percorso migliore attraverso il **numero di hop** da attraversare
** ogni **tot** secondi, il router invia la tabella di routing ai vicini



T=0

A			B			C			D			E			F		
14	1	.	14	1	.	55	1	.	08	1	.	08	1	.	78	1	.
23	1	-	55	1	-	66	1	-	66	1	-	23	1	-	92	1	-
78	1	-															

T=1

A			B			C			D			E			F		
14	1	.	14	1	.	55	1	.	08	1	.	08	1	.	78	1	.
23	1	-	55	1	-	66	1	-	66	1	-	23	1	-	92	1	-
78	1	-	23	2	A	14	2	B	55	2	C	14	2	-	14	2	A
55	2	B	78	2	A	08	2	D	23	2	E	78	-	A	23	2	A
08	2	E	66	2	C							66	2	D			
92	2	F															

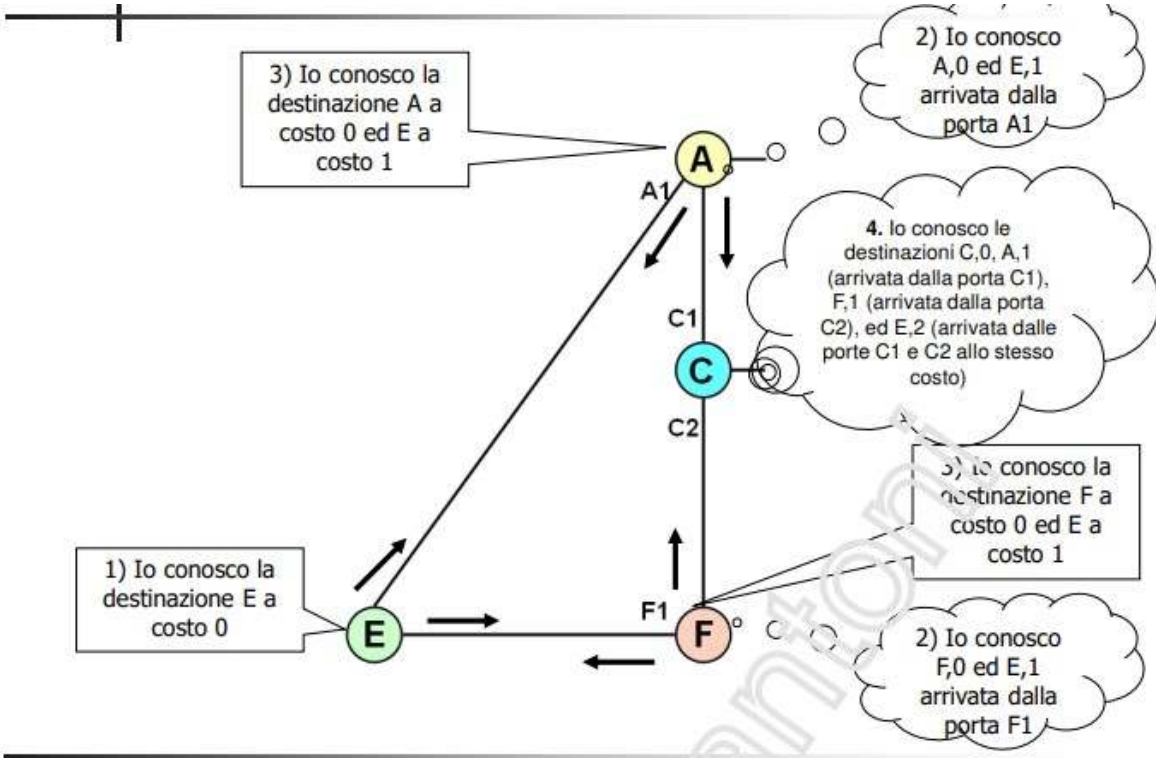
--> Algoritmi di Instradamento

Esistono 2 principali categorie di algoritmi:
- **statici**: basati su tabelle manuali, i cui path cambiano raramente
- **dinamici**: la topologia, i percorsi e i costi della rete possono variare al variare del carico

Questi vengono suddivisi in altre 2 categorie:
- **sensibili al carico**: questi cambiano la metrica a seguito della congestione del link in questione
- **insensibili al carico**: sono quelli usati attualmente, che non riflettono l'attuale livello di congestione

In base a come viene calcolato il percorso sono suddivisi in:
- **decentralizzati** (distance vector): il calcolo viene effettuato in maniera iterativa e distribuita, poiché non c'è una conoscenza globale della rete, ma solo dei vicini
- **globali** (link state): in questo caso si ha a disposizione tutta la mappa della rete tramite degli algoritmi --> **flooding** della rete: invio simultaneamente dati su tutte le porte di uscita

- Distance Vector



// PRO & CONTRO – Distance Vector

+ semplice da elaborare
+ richiede poche capacità di calcolo, di memoria e di elaborazione

- potrebbe portare a loop
- convergenza lenta
- non possono essere usati tanti hop (RIP ne ha max 15)

// PRO & CONTRO – Link State

+ ho una mappa completa della rete
+ non è suscettibile agli errori

- consumano tantissima banda
- non sono facilissimi da configurare
- richiedono capacità elaborative elevate

// Link state – Algoritmi utilizzati

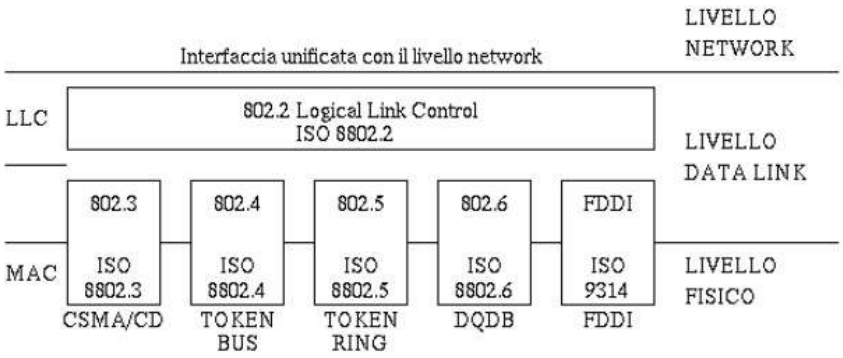
Gli algoritmi utilizzati dal Link State sono definiti come **SSSP** (Single Source Shortest Path) e sono:
- Algoritmo di **Dijkstra**
- Algoritmo di **Bellman-Ford**

Sono simili, l'unica differenza sostanziale è la **metrica**: Bellman-Ford può gestire anche pesi/costi **negativi**, mentre il Djikstra solo costi **positivi**.

/* Capitolo 5: LAN - Local Area Network (again) */

In una LAN, nel momento in cui un host nella rete trasmette, esso diventa proprietario di **tutto** il mezzo trasmissivo (problema di sicurezza non indifferente agli albori di LAN).

Gli elementi principali di una LAN sono i protocolli Standard, come **IEEE 802**, e il cablaggio strutturato da **EIA/TIA 568**.
L'IEEE 802 definisce i protocolli standard, tra cui ricordiamo **l'802.3** per l'Ethernet e **802.11** per il Wireless LAN. Inoltre ha suddiviso il livello Data-Link in:
- **Logical Link Control** (LLC) che è comune a tutte le LAN ed è l'interfaccia unificata verso il livello network
- **Media Access Contro** (MAC) che è specifico per ogni LAN e risolve il problema della condivisione del mezzo trasmissivo



Siccome la complessità delle reti porta ad essere soggetti a fenomeni di interferenza in grado di generare dati casuali e modificare quelli già presenti. Quindi sono stati sviluppati de controlli di trasmissione:
- **controllo di parità**
- **somme di controllo**
- **Controllo a Ridondanza Ciclica** (CRC)

// 2 tipi di collegamento

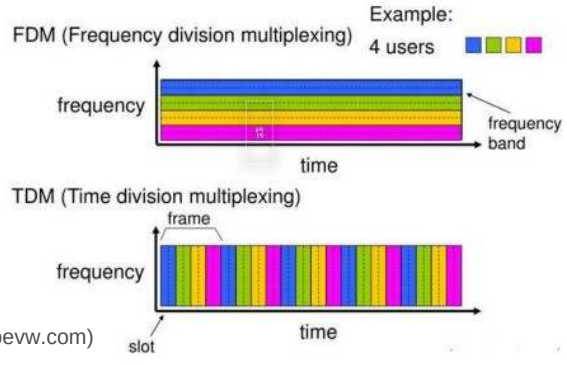
- **Broadcast**: più stazioni trasmittenti e riceventi connessi allo stesso canale condiviso
- **Punto-a-punto**: costituito da un trasmittente a un'estremità ed un unico ricevente all'altra

in caso di connessione il cavo dovrebbe essere il CAVO CROSSOVER (domanda d'esame)

// Protocolli ad accesso multiplo

Sono protocolli richiesti in un'ampia varietà di configurazioni di rete, comprese le LAN cablate, quelle wireless e le reti satellitari. Questi sono suddivisibili nelle seguenti categorie:
- Protocolli a suddivisione di canale --> **TDM e FDM**
- Protocolli ad accesso casuale --> **Slotted ALOHA**
- Protocolli a rotazione --> **Polling, Token-Passing**

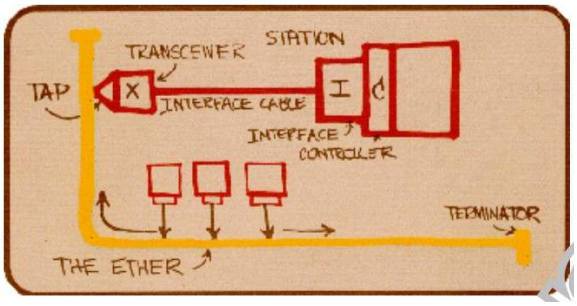
// Tecniche di allocazione



- **Statica**: il mezzo trasmissivo viene “partizionato” in base al tempo e alla frequenza. Quindi tramite Frequency Division Multiplexing **FDM** e Time Division Multiplexing **TDM**
- **Dinamica**: il canale tramissivo può essere assegnato a **turno** (durata limitata di tramissione) o a **contesa** (ogni sorgente prova a trasmettere indipendemente dalle altre)

Ethernet II Frame Structure and Field Size					
8 Bytes	6 Bytes	6 Bytes	2 Bytes	46 - 1500 Bytes	4 Bytes
Preamble	Destination Address	Source Address	Type	Data	Frame Check Sequence

L’unità di informazione del livello data link è detta **trama** o frame e permette di separare il flusso dei dati in **unità più piccole** e quindi più facilmente controllabili tra i **64 e 1518 byte**.



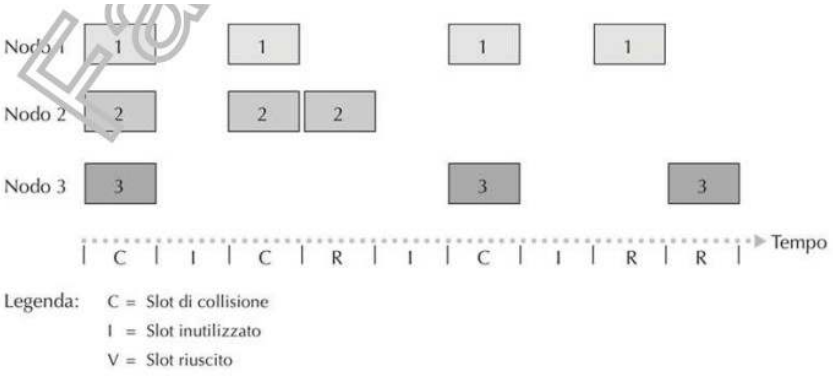
Ci sono diversi tipi di frame ethernet, il più importante è **l’IEEE 802.3 CSMA-CD**

Questo è un abbozzo fatto da Metcalfe nel 1976. C’è un solo BUS e non è deterministico, nel senso che non c’è regole per stabilire quando “parlare”. Ad oggi l’Ethernet è la tecnologia cablata più diffusa, ma si è passati da una topologia a BUS ad una a **stella**.

Questo perché la topologia a BUS comporta alcuni problemi:
- l’interruzione del BUS porta all’interruzione di tutta la rete;
- aggiungere un end system comporta un grosso intervento e il fermo della rete;
- un frame immesso sul BUS si propaga in entrambe le direzioni.

// Slotted ALOHA

Questo protocollo consente a un singolo nodo di trasmettere continuamente pacchetti alla massima velocità del canale, **quando è il solo nodo attivo**.



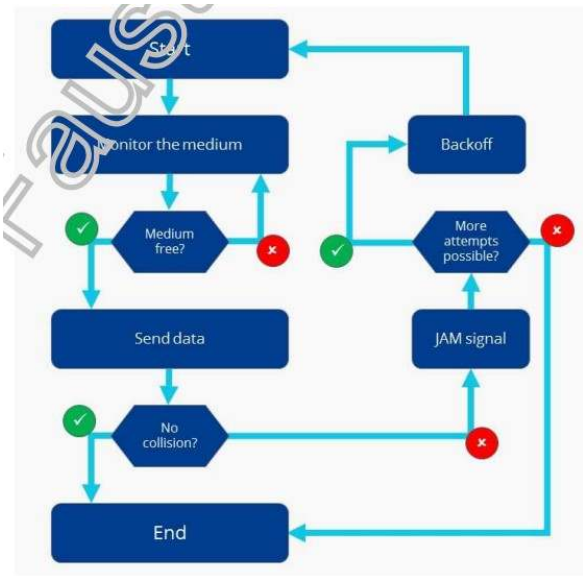
I nodi 1, 2 e 3 collidono nel primo slot. Il nodo 2 trasmette nel quarto slot, il nodo 1 nell'ottavo e il nodo 3 nel nono.

In questo modo le trasmissioni venivano fatte in modo **casuale**, dove ogni nodo una volta rilevata una collisione poteva decidere **indipendentemente** quando ritrasmettere. Lo slotted ALOHA prevede che comunque tutti i nodi si **sincronizzino** a partire dall'inizio di uno slot.
Successivamente viene introdotto il **CSMA-CD**, per sincronizzare il tutto.

// CSMA-CD

Protocollo distribuito di controllo --> **Carrier Sense Multiple Access**: il **Multiple Access** indica l'elevato numero di computer che sono connessi; il **Carrier Sense** indica che il computer prima di trasmettere testa il mezzo trasmissivo per vedere se c'è già una trasmissione in corso.

Tuttavia le collisioni **non sono scongiurate**: se due terminali iniziano a parlare contemporaneamente si crea una collisione e ci si ritrova nella situazione tipo il parlarsi sopra di persona (ci si parla sopra anche nel riprendere la discussione).
Il CD sta per questo --> **Collision Detect**, il rilevamento degli errori avviene intuendo le differenze del segnale trasmesso nel BUS: segnale diverso --> collisione.



Essenzialmente, al momento della collisione viene trasmesso un segnale speciale detto **jamming** (sequenza di 32bit) che avvisa tutti i terminali di attendere nella trasmissione per un **tempo** x **randomico**.
Se la collisione ricapita anche dopo questo tempo x, si raddoppia il tempo di attesa --> **Exponential Back-off**.

// Dominio

- **Dominio di Broadcast**: porzioni di rete in cui il broadcast riesce a raggiungere tutti gli host
- **Dominio di Collisione**: porzioni di rete in cui si verifica una collisione

Il tempo di rilevamento per una collisione è **α = lunghezza collegamento/lunghezza pacchetto**

// APPUNTI SICUREZZA PRIMA LEZIONE

Penetration Test: tecniche di sicurezza per controllare le potenzialità delle reti e dei sistemi

(crack). Esistono diversi e vari tipi di penetration test:
- External Testing: senza sapere nulla dell'infrastruttura, con strumenti esterni provi a vedere ciò che succede nell'azienda ()non il migliore
- Internal Testing: entro dentro l'azienda e poi cercare di osservare
- Targeted Test: ministratore IT che chiede un test su un determinato sistema
- Blind testing: conosci solo il nome dell'azienda e poi cercare di risalire ad altri dati
- Double Blind testing: quando l'amministratore IT non sa che vengono attuate prve
- Social Engineering

Principali servizi di sicurezza:
- Confidenzialità: restricted, confidential, secret, top secret
- Integrità: far in modo che i documenti siano firmati
- Non ripudio: firma elettronica
- Controllo degli accessi: controllo a due fattori, OTP
- Disponibilità: metodo di recupero per file online
- Rintracciabilità: su internet rimane traccia di quello che si fa, tutte le azioni che violano la sicurezza
- Autenticazione: man in the middle, una terza persone può mettersi in mezzo tra host e client

Tecniche psicologiche: spesso non servono oggetti fisici per ingannare le persone, basta la psicologia. La maggior parte delle persone reagiscono tutti allo stesso modo, soprattutto a stimoli base. L'ingegneria sociale la si usa là dove on si arriva con i normali strumenti di intrusione informatica.

Principio dell'influenza di Cialdini: la gente fa cose che gli altri stanno facendo, poiché le persone sono influenzate dalle persone che seguono --> Influencer: sono coloro che condizionano anche in modo fittizio il nostro fare.

Si cerca la **fiducia** nella vittima: la fiducia è un atteggiamento che cambia la valutazione, che permette la visione da parte della vittima sempre positiva. Devo portare la vittima a fidarsi dei contenuti nelle mail.

"Chi difende deve proteggere un perimetro sconfinato. Mentre chi attacca deve solo trovare un punto vulnerabile."