INTRODUZIONE

Definizioni utili

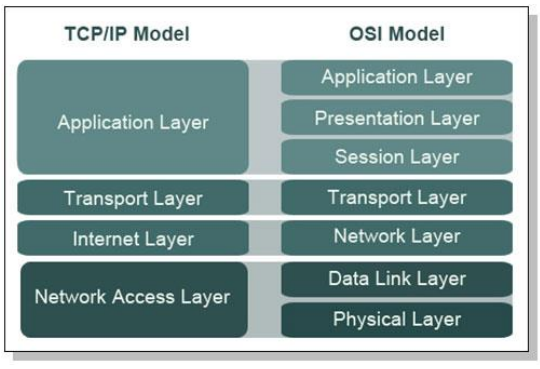
- **Trasmissione**: trasferimento di segnali da un punto a uno o più altri punti
- **Commutazione**: è il processo di interconnessione di due unità funzionali per il tempo necessario alla trasmissione dei dati
- **Segnalazione**: scambio di info che riguardano apertura, controllo e chiusura delle connessioni
- **Banda** (teoria dei segnali): ampiezza spettrale di un segnale/canale trasmissivo
- **Banda** (reti di telecomunicazione): bit/s
- **Capacità:** massima velocità (bit/s) trasmissiva di quel canale
- **Traffico offerto:** quantità di dati per unità di tempo che una sorgente cerca di inviare
- **Traffico smaltito (throughput):** parte di traffico offerto che viene consegnata effettivamente alla destinazione
- Throughput <= capacità canale Throughput <= traffico offerto Traffico offerto <= capacità canale
- **Rete:** insieme di nodi e canali che fornisce un collegamento tra uno o più punti per metterli in comunicazione
- **Nodo:** punto dove avviene la commutazione
- **Canale:** mezzo trasmissivo che collega i nodi (unidirezionale/bidirezionale, punto-punto, multi-punto)

Tipologia di reti

- Maglia completa: tollera bene i guasti ma al contempo è molto costosa per il grande numero di canali usati
- Albero: vulnerabile ai gusti, ma ha costi più bassi per via del basso numero di canali (n° canali < n° nodi); utilizzata nelle reti locali
- Stella: vulnerabile ai guasti (soprattutto il centro) ma ha un basso numero di canali; utilizzata nella rete telefonica con i doppini
- Maglia (mesh): non ha una topologia regolare, il calcolo dell'instradamento può essere complesso (molti percorsi alternativi), ma tollera bene i guasti e il numero di canali è a piacere di chi costruisce la rete
- Anello: unidirezionale o bidirezionale; usata per reti metropolitane
- Bus: c'è un solo canale condiviso da tutti

MODELLO ISO / OSI

ISO: International System Organization
OSI: Open System Interconnection



È un modello concettuale universalmente riconosciuto ed accettato ma che, a livello pratico, non si utilizza da nessuna parte; è solamente un riferimento teorico. È strutturato su 7 livelli, ognuno con il suo protocollo.
Nell'immagine seguente vediamo i 7 livelli del modello OSI e i loro corrispettivi nel modello TCP/IP.
I livelli superiori utilizzano i servizi di quelli inferiori: per inviare si scorre in giù la pila dei livelli, in su per ricevere. Il vantaggio di una struttura "a livelli" è proprio che il livello superiore utilizza quello inferiore, senza preoccuparsi di come questo venga effettivamente gestito dal livello inferiore.
Praticamente, per passare dal livello fisico a quello di applicazione, devo prima passare attraverso tutti gli altri che si sono in mezzo. Da notare che i livelli di applicazione, di presentazione e di sessione del modello OSI, sono riuniti in un solo livello di applicazione nel modello TCP/IP.

LIVELLO DI APPLICAZIONE

ARCHITETTURA CLIENT SERVER

Questa è composta da due parti: il client e il server. Il **client** è un host saltuariamente attivo (di solito) che inoltrano le richieste ai server. Importante sottolineare che i vari client (e i loro processi) NON comunicano direttamente tra loro, ma devono avvalersi del livello di trasporto sottostante per inviarsi i messaggi. Il **server** è un host perennemente attivo e risponde alle richieste dei vari client. In definitiva, l'host che avvia la comunicazione è detto client, mentre quello che viene contattato è detto server.
Nella maggior parte dei casi, un singolo server non è in grado di sopperire a tutte le richieste e perciò si uniscono più macchine che creano un unico server virtuale. Le maggiori applicazioni sono: WEB, FTP e posta elettronica.

ARCHITETTURA P2P

Viene sfruttata la connessione diretta tra gli host e perciò i server sempre attivi sono pochissimi, se non del tutto assenti. Il grande vantaggio è la **scalabilità**: un host può generare carichi sulla rete richiedendo un file, ma allo stesso tempo aggiungere capacità di servizio al sistema rispondendo alle richieste di altri peer.

Le applicazioni che "girano" su una macchina sono detti **processi**. Il processo mittente crea ed invia i messaggi, mentre quello destinatario li riceve ed invia le risposte (se necessarie). Di fatto agiscono uno come client e l'altro come server. I vari processi non comunicano direttamente tra loro, ma devono avvalersi del livello di trasporto sottostante per inviare/ricevere. I messaggi che si inviano, perciò, devono passare attraverso una **socket** (porta), la quale fa da interfaccia tra il livello applicativo ed il livello di trasporto.
Il progettista di un'applicazione può scegliere il protocollo di trasporto e lo fa in base ai servizi che i vari protocolli mettono a disposizione (trasferimento dati affidabile, throughput, temporizzazione, sicurezza). Perciò il servizio di trasporto deve essere affidabile, se vogliamo che tutti i pacchetti inviati vengano ricevuti correttamente (TCP), o non affidabile, ma così non abbiamo la certezza che il trasferimento di tutti i pacchetti sia avvenuto correttamente (UDP).
Il **Throughput disponibile** è la frequenza con la quale il processo mittente può inviare info al processo destinatario. Questo ovviamente può variare nel tempo e perciò il protocollo di trasporto di occupa di fornire un certo throughput disponibile garantito. Le app che hanno dei requisiti di throughput sono dette sensibili alla banda, mentre quelle che si adattano al throughput senza problemi sono dette elastiche.
Esaminiamo ora i servizi che Internet offre alle applicazioni, attraverso TCP e UDP.

SERVIZI DI TCP

Orientato alla **connessione** → i processi si scambiano messaggi a livello di trasporto prima di scambiarsi quelli richiesi dal livello di applicazione. Questo viene detto **handshaking**. Se va a buon fine viene aperta una connessione TCP tra le socket dei duo processi, altrimenti no.
Trasporto **affidabile** → i dati vengono inviati senza errori e nell'ordine opportuno, evitando anche duplicazioni di byte.
TCP include in più un controllo della **congestione**: se il carico della rete è troppo alto, il protocollo esegue una "strozzatura" del processo di invio. Questo, però, può diventare un problema per le app che richiedono un throughput minimo.

SERVIZI UDP

È un protocollo minimale: non orientato alla connessione, senza handshaking, non affidabile nel trasferimento dei dati e senza controllo della congestione (invia a qualsiasi frequenza).

Ma come fa un processo a indicare con quale altro processo vuole comunicare? Ogni processo è identificato da un **indirizzo IP** univoco dell'host e da una **porta**. Perciò un processo A per comunicare con il processo B deve conoscere l'indirizzo IP dell'host B e la porta che identifica il processo destinatario specifico.

Un protocollo a livello applicativo definisce come i processi di un'applicazione si scambiano i messaggi, ovvero:

- I tipi dei messaggi scambiati
- La loro sintassi
- La semantica dei vari campi che li compongono
- Le regole per determinare come e quando un processo invia/risponde ai messaggi

Bisogna distinguere tra applicazioni di rete e protocolli a livello applicativo: un protocollo a livello applicativo è solamente una parte di un'applicazione di rete.

PROTOCOLLO HTTP (HyperText Transfer Protocol)

È il protocollo al livello di applicazione del web. È implementato in due programmi, in esecuzione su macchine diverse, che comunicano tra loro scambiandosi messaggi HTTP. Uno fa il lato client (browser) e uno fa il lato server (server web). Questo protocollo utilizza TCP come protocollo di trasporto e perciò può avvalersi del trasferimento affidabile dei dati messo a disposizione. HTTP non si preoccupa di come vengono scambiati o recuperati i messaggi, lascia questo compito a TCP.
È un protocollo "senza stato" dal momento che i server HTTP non mantengono alcuna informazione sui client. Ma una serie di richieste in sequenza come viene gestita? L'applicazione usa un approccio con **connessioni non persistenti** se utilizza connessioni diverse per ogni richiesta, mentre connessioni **persistenti** se utilizza una sola connessione TCP. Questo protocollo può utilizzarle entrambe.
Trattiamo ora il formato del messaggi di richiesta e di risposta di HTTP.

Messaggio di richiesta:

- Scritto in ASCII in modo che possa essere letto dall'utente
- Riga di richiesta:
    - Metodo → GET, POST, HEAD, PUT, DELETE
    - URL (composto dal nome del server che ospita l'oggetto ed il suo percorso)
    - Versione HTTP
- Righe di intestazione:
    - Host in cui risiede l'oggetto
    - Campo che indica se la connessione è persistente o meno
    - User-agent (browser che inoltra la richiesta; Chrome)
    - Lingua in cui si vuole la pagina

Messaggio di risposta:

- 1 riga di stato:
    - Versione del protocollo HTTP
    - Codice di stato (200 ok, 400 bad request, 404 not found)
    - Messaggio di stato (mi dice se il server ha trovato l'oggetto richiesto e lo sta inviando o meno)
- 6 righe di intestazione:
    - Se il server chiude la connessione TCP o no dopo l'invio del messaggio
    - Data e ora in cui viene generala la risposta
    - Che tipo di server ha generato la risposta (Apache)
    - Data in cui l'oggetto è stato modificato l'ultima volta
    - N° di byte dell'oggetto da inviare
    - Tipo di oggetto
- Corpo dell'oggetto: tutti i dati che compongono l'oggetto da inviare

I COOKIE

Dal momento che i server HTTP sono stateless, si può volere che i server web possano autenticare gli utenti. Questo viene fatto attraverso i **cookie**, che sonsentono ai siti di tenere traccia degli utenti che li visitano. I cookie vengono incorporati nelle intestazioni delle richieste e delle risposte e sono salvati sia nel pc dell'utente che nei database dei siti. In questo modo, un sito web può "imparare" molto sull'utente.

CACHING WEB / SERVER PROXY

È un'entità di rete che soddisfa richieste HTTP per conto di un server web d'origine. Ha una propria memoria su disco in cui conserve le copie di oggetti recentemente richiesti, dato che ogni richiesta di oggetti da parte del browser viene inizialmente indirizzata alla cache. La richiesta di un oggetto da parte di un browser si compone dei seguenti passaggi:

1. Il browser stabilisce una connessione TCP con il proxy e invia la richiesta
2. Il proxy controlla se ha l'oggetto richiesto in memoria; in caso positivo lo invia al client
3. Se il proxy non ha l'oggetto richiesto, apre una connessione TCP con il server dove risiede tale oggetto e se lo fa mandare
4. Quando il proxy riceve l'oggetto lo inoltra al client che lo aveva richiesto

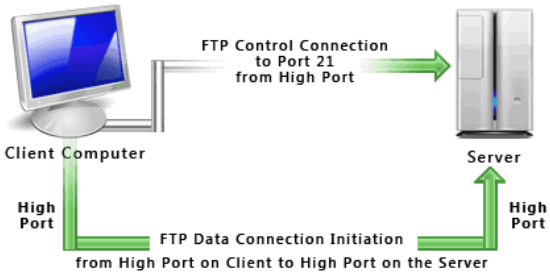
Il **proxy** è sia server che client e serve a ridurre i tempi di risposta alle richieste del client e il traffico di rete.

GET CONDIZIONALE

Il caching riduce fortemente i tempi di risposta, ma il problema che sorge ora è che la copia dell'oggetto preseente in cache potrebbe essere scaduta e l'oggetto, nel frattempo, potrebbe essere stato modificato sul server HTTP. Il **GET condizionale** è un meccanismo che permette alla cache di controllare se i suoi oggetti sono aggiornati o no. Questo viene gestito tramite il metodo GET e una riga di intestazione If-modified-since: in questo modo si comunica al server HTTP di inviare l'oggetto solo se è stato modificato rispetto alla data specificata in quel campo dell'intestazione.

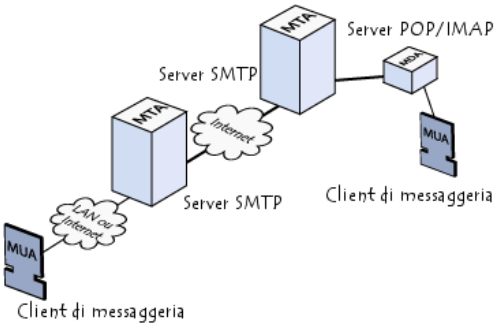
TRASFERIMENTO DI FILE FTP (File Transfer Protocol)

L'utente utilizza un host (locale) per trasferire un file da/verso un host remoto. Servono username e una password per poter scambiare il file. **FTP**, come HTTP, utilizza TCP come protocollo di trasporto; al contrario di HTTP, FTP utilizza due connessioni parallele dette controllo e dati. Per questo si dice che tale protocollo invia le proprie informazioni di controllo "fuori banda" (dato che utilizza una connessione separata), diversamente da HTTP che utilizza una sola connessione TCP e si dice perciò "in banda".



- **Controllo**: connessione con la porta 21 del server, dove il client invia username e password per l'identificazione e i vari comandi. Questa connessione rimane aperta per l'intera durata della sessione utente
- **Dati**: connessione con la porta 20 del client; viene chiusa ogni volta che finisce il trasferimento di un file (connessione non persistente).

POSTA ELETTRONICA IN INTERNET



Come il servizio di posta ordinario è un mezzo di comunicazione asincrono e ha tre componenti principali: gli **agenti utente**, i **server** e **SMTP** (Simple Mail Transfer Protocol). Ciascun destinatario ha una propria casella di posta collocata in un server di posta SMTP.

SMTP

Questo protocollo presenta un lato client, in esecuzione sul server del mittente, ed un lato server, in esecuzione sul server del destinatario. Quando un server invia agisce come client SMTP, mentre quando riceve agisce da server **SMTP**.

- Protocollo molto vecchio (1982) e per questo ha delle carattteristiche "arcaiche" (per esempio tratta il corpo dei messaggi di posta come ASCII a 7 bit).
- Non utilizza server di posta intermedi (di solito). Connette direttamente i due server tramite una connessione TCP
- Il client apre con il server una connessione TCP sulla porta 25. Se il serve è inattivo riprova più tardi, mentre se è attivo effettuano l'handshaking e poi si inviano le informazioni (email del mittente, email del destinatario)
- Utilizza TCP per la trasmissione affidabile dei dati
- Il server invia risposte al client dopo ogni comando (HELO, MAIL FROM …)
- Utilizza connessioni persistenti (se il server d'invio deve inviare molti messaggi, le invia al server di ricezione sulla stessa connessione TCP senza chiudere la connessione ad ogni messaggio)

Mentre HTTP trasferisce file da un server web ad un client web, SMTP trasferisce file (email) da un server di posta ad un altro, utilizzando entrambi connessioni persistenti.
HTTP → pull protocol (qualcuno carica informazioni e gli utenti a cui servono le scaricano)
SMTP → push protocol (il server di posta d'invio spedisce file al server di posta di ricezione)

MIME (Multipurpose Internet Main Extension)

È un protocollo è un'estensione di SMTP: il corpo dei messaggi di posta elettronica è preceduto da delle righe di intestazione che contiene delle informazioni di servizio.

- L'intestazione deve contenere i campo "From" e "To"
- Non sono più limitato ad un messaggio scritto in ASCII a 7 bit (in SMTP) ma lo estendo per poter poter inviare immagini o testi in lingue che non siano l'inglese
- Aggiunti altri due campi:
    - Content-type → tipo di azione da fare sul messaggio (se il corpo contiene un JPEG, l'agente utente dirige il corpo del messaggio ad una routine di decompressione delle immagini per visualizzarla)
    - Content-transfer-encoding → il tipo di codifica ASCII utilizzata nel messaggio per visualizzarlo correttamente

Messaggio ricevuto:

- Vengono inserite dal server di ricezione SMTP delle riche di intestazione con le informazioni del server che ha spedito e di quello che ha ricevuto l'email
- Se il messaggio è stato inoltrato, vengono aggiunti tutti i passi intermedi effetuati

L'agente che riceve i messaggi non dialoga direttamente con i server di posta. Per poter inviare una email, questa deve prima essere depositata sul server del mittente che la invia al server di posta del destinatario. L'agente del destinatario può utilizzare vari protocolli per dialogare con il proprio server di posta, in modo da controllare e leggere le email ricevute, dato che SMTP è solamente un protocollo di push (e non di pull). I protocolli per poterlo fare sono **POP3** e **IMAP**.

POP3 (Post Office Protocol – versione 3)

È un protocollo molto semplice e di agevole lettura, ma al contempo ha delle funzionalità piuttosto limitate. Entra in azione quando il client apre una connessione TCP verso il server di posta sulla porta 110 e poi esegue le seguenti operazioni: autenticazione (controlla usermane e password per autenticare l'utente), transazione (si recuperano i messaggi) e aggiornamento (comando "quit", che chiude la connessione POP3).
Il protocollo può:

- Scaricare e mantenere le mail
- Scaricare e cancellare le mail (non è più visibile da altri terminali)

Infine, **POP3** tiene traccia di alcune informazioni di stato ma non le trasporta mai.

IMAP (Interner Mail Access Protocol)

Con qusto protocollo, al contrario di POP3, l'utente può mantenere una gerarchia delle cartelle contenenti le mail sul server remoto. Per questo **IMAP** è più complesso, ma anche completo, di POP3.

- Si possono organizzare le mail in cartelle
- I server IMAP conservano informazioni di stato sull'utente da una sessione all'altra
- Ci sono dei comandi che permettono all'utente di scaricare le componenti dei messaggi

Oggigiorno la posta si basa molto sul web (vedi Hotmail, Google). In questo modo mi connetto alla mia casella di posta tramite un browser. In questo modo i messaggi vengono inviati dal browser utilizzando HTTP (e non POP3/IMAP). Altri, però, continuano ad utilizzare SMTP per mandare e ricevere messaggi.

DNS (Domani Name System)

Gli host di internet possono essere identificati in vario modo, come le persone. Gli **hostname** (www.cnn.com) vanno bene per noi uomini, ma non sono agevoli per l'elaborazione da parte dei server. Per questo motivo si utilizzano gli **indirizzi IP** [un indirizzo IP consta di 4 byte ed ha una struttura gerarchica ed è costituito da una stringa in cui un punto separa uno dei byte, espressi con un numero decimale compreso tra 0 e 255]. Il **DNS** perciò ha il compito di tradurre gli hostname nei loro rispettivi indirizzi IP.

Il DNS è composta da una struttura gerarchica composta da databases ed da un protocollo a livello di applicazione che consente agli host di interrogare il database. Esso utilizza UDP sulla porta 53 e viene utilizzato a sua volta da altri protocolli come HTTP, SMTP e FTP.

Il procedimento che segue il DNS è il seguente:

- La macchina utente fa girare il lato client dell'applicazione DNS
- Viene estratto l'hostname dall'URL
- Il client DNS invia una query con l'hostname al server DNS
- Il server riceve la query e invia la risposta contenente l'indirizzo IP associato all'hostname
- Una volta ricevuta la risposta, il client può aprire una connessione TCP con quell'indirizzo IP

L'utilizzo del DNS introduce un ritardo aggiuntivo, ma l'indirizzo IP che si cerca si trova spesso nella cache di un server DNS vicino.

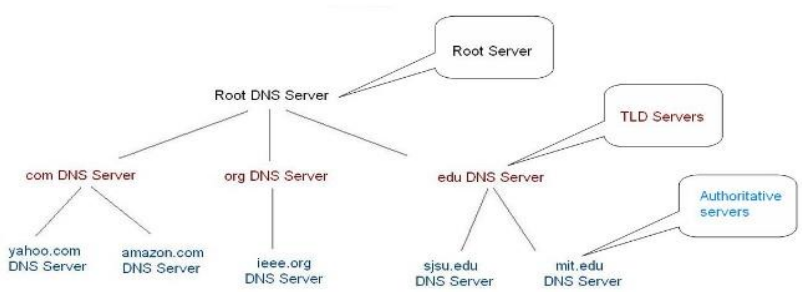
Gli altri servizi gestiti dal DNS sono:

- Host aliasing: un host dal nome complicato può avere uno o più sinonimi (**alias**). Questo alias reindirizzerà all'**hostname canonico**, da cui poi calcola l'indirizzo IP
- Mail server aliasing: permette al server di posta di una società e al server web di avere hostname idendici (alias) (entrambi possono essere chiamati enterprise.com)
- Distribuzione locale: i siti con molto traffico vengono replicati su più server e ciascuno di questi gira su macchine diverse con IP differenti. Nel caso di server web duplicati, va associato a ogni hostname canonico un insieme di indirizzi IP. Il server DNS li invia con ordine sempre differente, in modo da distribuire il carico dei server.

Da notare è che il DNS non è un unico server, ma è progettato in maniera **distribuita**. Nessun server DNS, infatti, ha le corrispondenze per tutti gli host in Internet, dato che sono distruibuiti tra tutti i vari server DNS.

Ci sono tre livelli di server DNS:

1. Server radice: sono 13 in tutto il mondo. Ogni root server, in realtà, è un clouster di server duplicati
2. Top-level domani server (TLD):
    - Si occupano di domini di alto livello (.com, .org, .net …)
    - Si occupano di domini locali di alto livello (.it, .fr, .uk …)
3. Server di competenza: sono server installati da chi possiede degli host internet pubblicamente accessibili e devono fornire record DNS che mappino i nomi di tali host in indirizzi IP.



Un server DNS locale, inoltre, ha la funzione di proxy: se un client inoltra una richiesta, il proxy inoltra la query in una gerarchia di server DNS.

I server DNS utilizzano in modo intensivo il **caching** per diminuire il ritardo e per ridurre il numero di

messaggi DNS. Si salva anche le informazioni anche se non è competente per il dominio in cui risiede l'hostname (queste informazioni vengono cancellate dopo un certo periodo di tempo, in genere 2 giorni).

I server DNS memorizzano i cosidderri **record di risorsa** (RR), tra cui quello che forniscono la corrispondenza hostname-IP. Ogni messaggio di risposta trasporta uno o più record. Un record contiene i seguenti campi:

(Name, Value, Type, TTL)

TTL è il "time to live", ossia il tempo di vita residuo del record. I campi Name e Value variano al variare di Type:

1. Type=A: utilizzatoquando si conosce l'associazione nome-indirizzo. Value è l'indirizzo IP associato al nome
2. Type=NS: utilizzato quando si gira la query ad un server DNS di un altro dominio. Name è un dominio e Value è l'hostname del server DNS di competenza
3. Type=CNAME: utilizzato se ad un hostname sono associati più indirizzi IP. Name è il nome canonico dell'host per il sinonimo Name
4. Type=MX: utilizzato se bisogna risolvere un hostname di un server di posta elettronica. Value è il nome canonico di un server di posta che ha il sinonimo Name

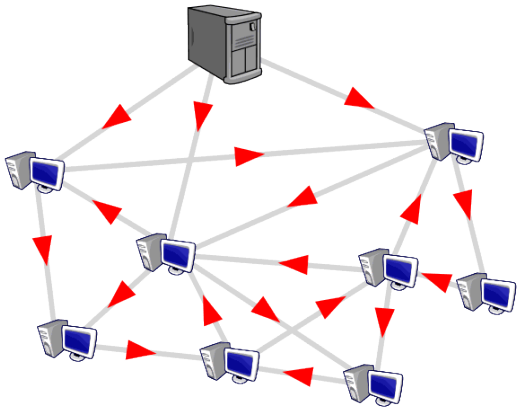
Grazie al caching dei server DNS, un server non di competenza potrebbe comunque avere l'associazione nome-indirizzo che ci serve e quindi invia un record di tipo A.

APPLICAZIONI PEER TO PEER (P2P)

Mentre nelle applicazioni web c'è un'architettura client-server con un server perennemente attivo, l'architettura P2P questa dipendenza è minima o del tutto nulla. I vari peer sono connessi in modo intermittente. Il grande vantaggio del peer to peer è la sua scalabilità.
Esamineremo tre differenti applicazioni P2P: la discribuzione di file, l'organizzazione e la ricerca di informazioni in una comunità di peer e la telefonia su intenret.

DISTRIBUZIONE DI FILE

Ogni peer può re-distribuire agli altri qualsi porzione del file che ha ricevuto, aiutando in questo modo il server nel processo di distribuzione (vedi Bittorrent).



Il **tempo di distribuzione** è il tempo richiesto perché tutti gli n peer ottengano una copia del file.
Il tempo perché il server S invii a tutti una copia del file F è

$$T \geq \max\{\frac{NF}{u_s}, \frac{F}{d_{min}}\}$$

dove N è il numero di peer, F è la dimensione del file, u_s la velocità di upload e d_{min} è la capacità di download minimia tra tutti i peer.
Ma nel P2P ogni macchina ha la capacità di upload, perciò

$$u_{tot} = u_1 + u_2 + \cdots + u_n$$

Il tempo di trasferimento minimo del file F diventa allora

$$T = \frac{NF}{u_{tot}}$$

Perciò il tempo minimo diventa

$$T \geq \max\{\frac{NF}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_{tot}}\}$$

Notiamo che con l'architettura client-server il tempo di distribuzione aumenta linearmente con l'aumentare dei peer, mentre con l'architettura P2P è sempre minore di quello ottenuto con c-s.

LIVELLO DI TRASPORTO

È il livello che permette la comunicazione logica tra i processi applicativi di host diversi, convertendo i messaggi che riceve da un processo applicativo mittente in **segmenti a livello di trasporto**. Poi passa il segmento al livello di rete che lo incapsula all'interno di un segmento a livello di rete (datagramma) e lo invia a destinazione. Grazie ai protocolli di questo livello, tutto procede come se gli host che eseguono i processi fossero direttamente connessi. I protocolli di questo livello sono implementati nei sistemi terminali, non nei router che lavorano solamente a livello di rete e perciò agiscono solamente sui campi a livello di rete del datagramma.
Mentre il livello di trasporto fornisce una comunicazione logica tra i processi che girano su host diversi, il livello di rete fornisce una connessione logica tra gli host. Determinati servizi possono essere garantiti dal livello di trasporto anche se il livello di rete non fornisce gli stessi servizi (vedi trasferimento dati affidabile di TCP).
I due protocolli che implementano il livello di trasporto sono **UDP** (servizio non affidabile e non connesso) e **TCP** (servizio affidabile, orientato alla connessione e servizi aggiuntivi).
Altre cose importanti:

- Segmento: pacchetto del livello di trasporto
- Internet è un servizio "best effort", ovvero fa del suo meglio per consegnare i segmenti tra host comunicanti, senza alcuna garanzia
- Passaggio da consegna da host-to-host a consegna da process-to-process è detto **de/multiplexing**

MULTIPLEXING E DEMULTIPLEXING

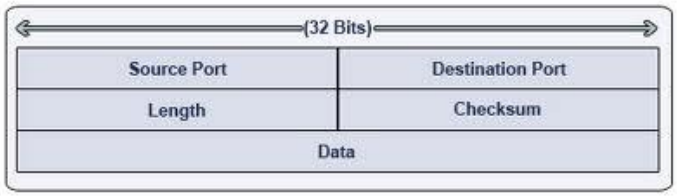
Il livello di trasporto ha il compito di trasportare i segmenti ricevuti dal livello di rete al processo applicativo in esecuzione sull'host. Il livello di rete scambia i segmenti tra host mentre quello di trasporto scambia i segmenti tra i processi all'interno dell'host.
Questo viene fatto inoltrando i dati alle **socket**, cioè alle porte attraverso le quali i dati fluiscono dalla rete al processo e viceversa. Perciò trasportare i dati dei segmenti del livello di trasporto verso la giusta socket viene detto **demultiplexing**. Il processo contrario, ovvero quello di radunare le porzioni di dato presso l'host di origine a partire dalle diverse socket, viene detto **multiplexing**.
Ciò però richiede che (1) le socket abbiano degli identificatori unici e che (2) ciascun segmento indichi sia la porta del mittente che quella del destinatario.
I numeri di porta sono a 16 bit e vanno perciò da 0 a 65535; quelli da 0 a 1024 sono detti **numeri di porta noti** e sono già utilizzati dal sistema e dai protocolli applicativi.
C'è da dire che, come nel caso del server web, non esiste sempre una corrispondenza 1:1 tra socket e porta. I server distinguono i client tramite gli indirizzi IP e il numero di porta di origine (entrambi, altrimenti non ci sarebbe univocità assoluta).

TRASPORTO SENZA CONNESSIONE: UDP (User Datagram Protocol)

Quantomeno, il livello di trasporto deve fornire un servizio di de/multiprexing per trasportare i dati dal livello di rete al processo destinatario corretto a livello di applicazione (e viceversa). UDP fa proprio il minimo indispensabile: oltre a de/multiplexing, offre anche una leggera forma di controllo sugli errori (checksum).
In **UDP** non esiste l'handshaking tra l'unità di avvio e quella di ricezione: per questo si dice **senza connessione**.
Il DNS è un tipico esempio di protocollo applicativo che utilizza UDP (evita i ritardi per stabilire una connessione TCP).
UDP "spara" dati a raffica senza alcun preliminare con chi riceve; se non ottiene alcuna risposta, riprova per un tot di volte finché non informa l'applicazione che non ha ottenuto alcuna risposta.
Ci sono dei motivi per cui un'applicazione potrebbe scegliere di utilizzare UDP rispetto a TCP, anche se offre meno servizi:

- C'è un controllo minore sui dati, i quali vengono impacchettati e spediti immediatamente senza fare alcun tipo di controllo (utile per app in tempo reale)
- Non introduce alcun ritardo, poiché non c'è handshaking per stabilire una connessione
- Non conserve lo stato della connessione
- Ha un'intestazione più corta: 8 bit (rispetto ai 20 bit di TCP)

Però, il fatto che non faccia nessun controllo può creare dei problemi. Per esempio con lo streaming video con un bit-rate alto, UDP intaserebbe subito il router e, di conseguenza, si verificherebbero pesanti perdite di pacchetti, facendo calare vistosamente il bit-rate.



L'intestazione di un segmento UDP presenta 4 campi: il numero di porta sorgente e destinazione, la lunghezza in byte del segmento (intestazione compresa) e il checksum (utilizzato dal destinatario per controllare l'integrità del pacchetto).
Il checksum viene calcolato sommando gli altri tre campi e poi facendo il complemento bit a bit. In ricezione vengono sommati tutti i campi e la somma dei due numeri calcolati deve essere 1111111111111111. Oltre a questo, UDP non fa nulla per evitare gli errori.

TRASPORTO CON CONNESSIONE: TCP (Trasmission Control Protocol)

TCP é un protocollo **orientato alla connessione** perché i processi devono effettuare un handshake prima che inizi effettivamente la comunicazione. Offre inoltre un servizio full-duplex, cioè il flusso di dati può esserci contemporaneamente in entrambe le direzioni, ed è di tipi punto-punto, dal momento che si collega un singolo mittente con un singolo destinatario.
Il metodo per instaurare una connessione TCP è detto **handshake a tre vie**, dal momento che servono tre segmenti per iniziarla: il client invia per primo un segmento speciale, il server risponde con un secondo segmento speciale ed infine il client risponde con un terzo. Una volta instaurata la connessione, i due processi possono scambiarsi i dati.
La quantità massima di dati che possono essere messi in un segmento è detta maximum segment size (**MSS**).
Una connessione TCP è costituita da buffer, variabili e connessione socket verso un processo in un host e da un altro insieme di buffer, variabili e connessione socket in un altro host.
Viene stimato il **RTT** (Round Trip Time; tempo che intercorre tra l'istante dell'invio del segmento e quello di ricezione dell'ack del segmento; in altre parole il tempo di andata e di ritorno) per impostare il timeout di ritrasmissione del segmento (**RTO**).
I servizi che offre questo protocollo sono:

- ACK cumulativi: indica l'avvenuta e corretta ricezione di più segmenti, non un ack per ogni singolo segmento
- Trasporto dati affidabile
- Controllo del flusso
- Controllo di congestione della rete

Quando si instaura una connessione, TCP fornisce una specie di circuito virtuale che è:

- Bidirezionale (full-duplex)
- Controlla gli errori e la sequenza dei segmenti (devono arrivare nell'ordine in cui sono stati mandati)
- Se ci cono segmenti non conformi allo scadere dell'RTO, questi vengono ritrasmessi
- Stima RTT in base a RTO
- Calcola il checksum per controllare eventuali errori
- La velocità di trasmissione viene regolata variando la dimensione della finestra di trasmissione (negli ack comunico lo spazio disponibile nella finestra)

Il comportamento degli **ACK**:

- Lo invio se il segmento è nella sequenza corretta
- Se un segmento è duplicato lo scarto e invia un ack dell'ultimo segmento ricevuto
- Se un segmento ha il campo checksum errato non lo invia
- Se un segmento non è nella sequenza corretta lo memorizza, ma invia un ack rispetto all'ultimo elemento ricevuto nella sequenza corretta

La velocità di trasmissione, in assenza di errori, è

$$\frac{Finestra\ di\ trasmissione}{RTT}$$

Quindi, per regolare la velocità, posso agire sul RTT (ritardando l'invio degli ack, ma posso provocare delle ritrasmissioni inutili) o sulle dimensioni della finestra di trasmissione.
La velocità di trasmissione di un mittente che utilizza TCP è regolata da:

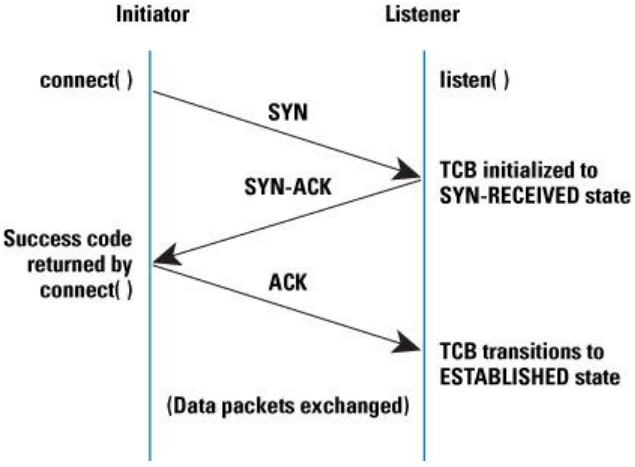
- **Controllo del flusso**: evita che un host che invia velocemente intasi un host che riceve lentamente
    - Il ricevitore può imporre la dimensione massima della finestra indicandola negli ack
- **Controllo della congestione**: evita che un host saturi la rete trasmettendo sempre alla velocità massima, anche se chi riceve non riesce a tenere il passo
    - Il trasmettitore regola la dimensione massima della finestra in base alle perdite riscontrate durante la trasmissione

IL SEGMENTO TCP

<table>
<tr><td colspan="3">Source Port (16)</td><td>Destination Port (16)</td></tr>
<tr><td colspan="4">Sequence Number (32)</td></tr>
<tr><td colspan="4">Acknowledgement Number (32)</td></tr>
<tr><td>Data offset</td><td>Reserved (6)</td><td>Flags (6)</td><td>Window (16)</td></tr>
<tr><td colspan="3">Checksum (16)</td><td>Urgent (16)</td></tr>
<tr><td colspan="4">Options and Padding</td></tr>
<tr><td colspan="4">Data (Varies)</td></tr>
</table>

- Il numero di porta sorgente e destinatario indicano le app che stanno inviando/ricevendo dati; con i relativi IP identificano univocamente i processi
- Il numero di sequenza e il numero di riscontro servono per implementare il trasferimento dati affidabile; il numero di riscontro è il numero di sequenza +1 ed indica l'ultimo byte ricevuto correttamente
- Il campo lunghezza dell'intestazione (data offset) indica la lunghezza dell'header in byte (variabile, ma di solito è 20 byte)
- Campo riservato/non trattato
- Il campo opzioni è facoltativo e di lunghezza variabile
- Il campo flag contiene 6 opzioni che possono essere abilitate o meno (RST, SYN, FIN, PSH, URG e ACK)
- Il campo window indica il numero di ack consecutivi che il ricevitore è disposto ad accettare
- Il campo urgent è un puntatore a dei dati urgenti contenuti nel campo data
- Il campo dati contiene i dati veri e propri che devono essere inviati

Dal momento che TCP è un protocollo orientato alla connessione, le due entità che devono mettersi in comunicazione devono fare l'handshaking a tre vie. Ecco come funziona:



SYN → nel segmento chi inizia la connessione specifica il numero di porta a cui vuole accedere
SYN+ACK → l'host che ascolta risponde con un segmento con SYN e ACK a 1 nei flag
ACK → chi ha iniziato la connessione risponde a chi ascolta con un ACK e inizia in questo momento la connessione vera e propria e la trasmissione dei dati

Tra i parametri che i due host si scambiano all'inizio può esserci il valore (in byte) del MSS, ovvero la dimensione massima dei segmenti che è in grado di ricevere (default = 536 byte).

Quando invece si vuole terminare la connessione, si manda un segmento con il flag FIN a 1; ma visto che la connessione è bidirezionale, questo deve avere un riscontro da parte dell'altro host e anche la chiusura (come l'apertura) deve avvenire da entrambe le parti. Se ciò avviene da una sola parte, l'altra può continuare ad inviare dati, poiché la connessione è chiusa solamente da un lato.

GESTIONE DEL TIMEOUT DI RITRASMISSIONE (RTO)

In TCP c'è un meccanismo di timeout e di ritrasmissione per ripristinare i segmenti perduti. La questione più importante è la durata di tale intervallo: ovviamente RTO > RTT, altrimenti avrei sempre delle ritrasmissioni. Come viene perciò calcolato e gestito l'RTO?

- RTO indica il tempo entro il quale il mittente si aspetta di ricevere un ack di ritorno
- Non è statico ma dipende da:
    - Distanza tra sorgente e destinatario
    - Condizioni della rete
    - Disponibilità della destinazione
- Viene calcolato durante l'instaurazione della connessione o durante la trasmissione dei dati
- Si basa sul valore di RTT:
    - SRTT = (1-a) SRTT + a RTT
        - Smoothed RTT, ovvero la stima del valore medio di RTT
        - 0 <= a <=1, tipicamente a = 0.125
    - RTTVAR = (1-b) RTTVAR + b * |SRTT – RTT|
        - RTTVAR è la stima della varianza di RTT
        - Tipicamente b = 0.25
    - RTO = SRTT + 4 RTTVAR
- Se c'è ritrasmissione probabilmente vuol dire che c'è congestione → RTO' = 2*RTO → dopo la trasmissione del segmento viene riportato il valore originale di RTO
- Il valore minimo è 200 ms, il valore massimo >= 60 s

CONTROLLO DI FLUSSO

Se l'applicazione è relativamente lenta nella lettura dei dati, può accadere che il mittente mandi in overflow i buffer di ricezione inviando dati troppo rapidamente. TCP offre questo servizio per evitare queste situazioni. Il controllo di flusso viene eseguito facendo mantenere al mittente una variabile, chiamata **finestra di ricezione** (reciever window), che fornisce al mittente un'indicazione dello spazio libero nel buffer del destinatario. La finestra di ricezione viene perciò impostata alla quantità di spazio disponibile nel buffer.
Questa tecnica è detta di **sliding window** (finestra scorrevole): invio n segmenti e man mano che arrivano i relativi ack spedisco altri segmenti.
La dimensione della finestra è troppo piccola perché sottovaluto la banda. Ecco che entra in gioco il campo RCWND (reciever window) negli ack per indicare al trasmettitore il numero di byte che il ricevitore può ospitare nel suo buffer, senza mandarlo in overflow.
Il paramentro RCWND viene impostato **dinamicamente** in funzione dei segmenti già consegnati.

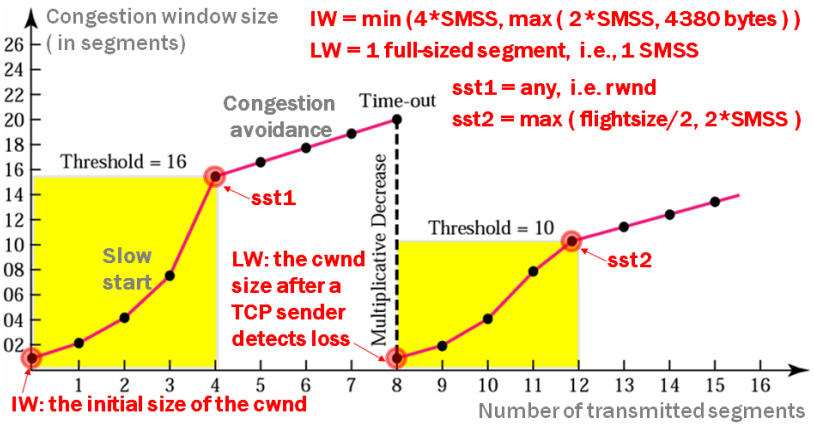
Ma se un segmento inviato in una sliding window si perde, come si comporta TCP? Può comportarsi in due modi:

- **Go-back-n**: ritrasmette tutti i segmenti della finestra; è di facile implementazione
- **Selective repeat**: ritrasmette solo i segmenti di cui non ha ricevuto un ack; è di più difficile implementazione ma è evidentemente più efficiente, visto che non sovraccarica la rete inviando pacchetti che erano stati correttamente ricevuti

CONTROLLO DELLA CONGESTIONE

L'idea di base è la riduzione da parte del mittente della propria frequenza di invio (riducendo la dimensione di CWND, la **finestra di congestione**) a seguito di una perdita di un pacchetto. Il mittente TCP incrementa in modo additivo la propria frequenza di invio quando capisce che il percorso tra i due host è libero dal traffico, mentre la decrementa in modo moltiplicativo quando rileva che il percorso è congestionato.
La dimensione di CWND (congestion window) cambia in 4 passaggi:

1- Inizialmente CWND = 1 segmento (ovvero MSS byte)
SSTHRESH = RCWND o RCWND/2
2- CWND cresce con l'algoritmo **slow start** fino a che CWND = SSTHRESH
3- Quando CWND = SSTHRESH la dimensione di CWND viene regolata secondo l'algoritmo **congestion avoidance**
4- CWND cresce finché CWND = RCWND



In questa immagine si capisce meglio come il controllo della congestione da parte di TCP utilizzi i due algoritmi che modificano la dimensione della finestra di congestione.

Quando si utilizza l'algoritmo slow start, per ciascun riscontro la CWND aumenta la sua dimensione di un segmento (in questo modo aumenta esponenzialmente nel grafico); con la congestion avoidance, invece, per ciascun ack la CWND aumenta di 1/CWND (aumenta in modo lineare).
Se si trovano degli errori nei pacchetti o avvengono delle perdite:

1) La trasmissione si interrompe
2) Si attende lo scadere del RTO
3) Si pone SSTRESH = CWND/2 e CWND = 1
4) Si ritrasmette il pacchetto seguendo il **selective repeat**
5) Si riparte con slow start finché CWND = SSTHRESH

Le due cause principali che possono portare alla perdita di un segmento sono degli errori di trasferimento (influiscono sul singolo segmento) oppure la congestione della rete (provoca la perdita di più pacchetti).
Ma se si perde un solo segmento, in questo modo si ha una reazione "eccessiva" con la chiusura della finestra di congestione. Perciò si usano, in coppia, due algoritmi:

- **Fast retrasmit**: il segmento viene ritrasmesso subito, prima dello scadere del timeout. Il mittente si può accorgere della perdita prima che si verifichi l'evento di timeout grazie agli ack duplicati, che riscontrano un segmento rià ricevuto dal mittente. In genere, per 3 ack duplicati consecutivi si assume che il segmento è andato perduto (TCP Tahoe).
- **Fast recovery**: la CWND non viene chiusa del tutto, ma viene ridotta al valore di SSTHRESH (invece di metterla al valore iniziale). In questo modo si elimina la fase di slow start dopo un triplice ack duplicato (TCP Reno).

LIVELLO DI RETE

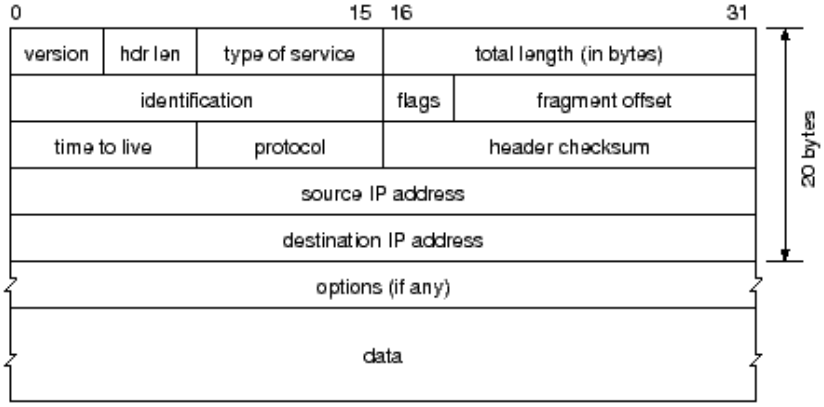
I protocolli di rete, al contrario di quelli di trasporto, sono implementati in tutti i router, dal momento che il loro ruolo è quello di inoltrare i **datagrammi** (pacchetto a livello di rete) dai loro collegamenti d'ingresso a quelli di uscita. Le principali funzioni del livello di rete sono l'**instradamento** (routing) e l'**inoltro** (forwarding). Per instradamento si intende trovare il percorso che i pacchetti devono seguire per arrivare dalla sorgente alla destinazione (algoritmi di instradamento), mentre con inoltro si intende il processo che deve fare il router quando riceve un pacchetto, trasferendolo sull'appropriato collegamento d'uscita.
Instradamento → processo di rete che determina i percorsi dei pacchetti dalla sorgente alla destinazione
Inoltro → azione locale con cui il router trasferisce i pacchetti dall'interfaccia di ingresso a quella di uscita
La relazione tra i due è importantissima, dal momento che è l'algoritmo di instradamento che determina i valori da inserire nelle tabelle di inoltro dei router.

IP (Internet Protocol)

Ogni datagramma IP è composto da:

- Intestazione: contiene le informazioni per la consegna; ha una lunghezza che varia da 20 byte a 60 byte
- Dati (payload): la sua dimensione è determinata dal livello di trasporto, ma ho un minimo di 64 kbyte

In modo più specifico è così composto:



- Version: versione del protocollo (IPv4), necessaria per la corretta interpretazione del datagramma
- Hdr len: lunghezza dell'header (in genere 20 byte) e perciò indica quando iniziano i dati
- Type of service: tipo di servizio che il datagramma richiede, distinguendo i diversi tipi di datagrammi
- Total lenght: lunghezza totale del datagramma IP (intestazione + dati) misurata in byte
- Identification, flags, fragment offset: questi campi hanno a che fare con la frammentazione del datagramma
- Time to live: indica per quanto tempo il datagramma può "vivere" nella rete, evitando che possa circolarvi all'interno per sempre
- Protocol: indica il protocollo di trasporto con il quale i datagrammi vanno trattati
- Header checksum: il checksum per gli errori nei datagrammi ricevuti
- Source IP address
- Destination IP address
- Options: servono per estendere l'intestazione IP; si utilizzano sporadicamente
- Data: il carico utile da inviare, che contiene segmenti a livello di trasporto da consegnare alla destinazione

FRAMMENTAZIONE DEI DATAGRAMMI

La massima quantità di dati che un frame a livello di rete può trasportare è detta **MTU** (maximum trasmission unit) ed è un limite rigido alla lunghezza dei datagrammi IP.
Se un singolo datagramma è più grande dell'MTU, come in genere succede, questo va diviso in parti più piccole per essere inviato. I **frammenti** dovranno poi essere **riassemblati** prima di raggiungere il livello di trasporto alla destinazione, dal momento che TCP/UDP si aspettano di ricevere segmenti completi dal livello di rete.

Dal momento che IP fornisce un **servizio non affidabile**, alcuni frammenti potrebbero non arrivare mai a destinazione. Per questo, affinché l'host ricevente sia certo di aver ricevuto tutti i frammenti e sia in grado di riassemblarli correttamente, l'ultimo ha un bit posto a 0, mente in tutti gli altri è posto a 1.
Dal momento che IP non è affidabile, si devono fare alcuni passaggi quando si frammenta un datagramma:

- Quando il router frammenta il datagramma, li contrassegna con gli indirizzi di origine e di destinazione con l'identificatore numerico del datagramma originale
- Quando la destinazione riceve una serie di frammenti dallo stesso host guarda gli identificatori per vedere se fanno parte dello stesso datagramma
- L'ultimo frammento ha un bit posto a 0 invece che a 1 per comunicare che si è arrivati all'ultimo frammento
- Se qualche frammento non arriva a destinazione, il datagramma viene scartato e non viene inoltrato al livello di trasporto (TCP o UDP che sia)

INDIRIZZAMENTO

Il confine tra host e collegamento fisico viene detto **interfaccia**. L'indirizzo IP è associato ad un'interfaccia, invece che al router o all'host. Difatti un router può presentare più interfacce, ognuna per ogni suo collegamento.
Gli IP sono lunghi 32 bit (4 byte) e perciò, in totale, ci sono 2^{32} indirizzi IP disponibili (circa 4 miliardi). Essi sono scritti con la cosiddetta **notazione decimale puntata**: ciascun byte viene rappresentato in decimale ed è separato con un punto dagli altri byte dell'indirizzo.
L'indirizzo IP è **globalmente univoco**: può, al più, avere una parte comune che si trova in una **sottorete**, ovvero una rete che interconnette più interfacce di host e una sola interfaccia del router.
In una sottorete IP assegna a tutta la sottorete l'indirizzo a.b.c.d/e, dove /e viene detto **maschera di subnet** ed indica il numero di indirizzi che può ospitare la sottorete.
Una /24, ad esempio, indica che i 24 bit più a sinistra dell'indirizzo (detto anche prefisso) definiscono l'indirizzo della sottorete, mentre gli 8 bit restanti (32 bit – 24 bit) indicano che la sottorete ha a disposizione 2^8 indirizzi.
C'è un indirizzo fondamentale, detto di **broadcast**. Se un datagramma ha come destinazione 255.255.255.255, il messaggio verrà inviato e consegnato a tutti gli indirizzi all'interno della stessa sottorete.

DHCP (Dynamic Host Configuration Protocol)

DHCP consente ad un host di ottenere automaticamente un indirizzo IP in modo automatico, senza utilizzare altre informazioni quali subnet mask, gateway di default, indirizzo DNS locale e senza configurarlo manualmente.
DHCP può utilizzare indirizzi IP **persistenti**, in modo che all'host che entra in rete venga assegnato sempre lo stesso indirizzo, oppure IP **temporanei**, assegnando indirizzi diversi ad ogni accesso alla rete da parte dell'host.
Il protocollo si articola in quattro punti:

1) Individuazione del server DHCP: l'host manda un segmento UDP dalla porta 67; l'indirizzo di destinazione è quello di broadcast
2) Offerte del server DHCP: quando il server riceve il messaggio risponde al client con un messaggio di offerta DHCP, inviato in broadcast; l'indirizzo IP offerto è valido per un tempo limitato
3) Richiesta DHCP: il client sceglie tra le offerte ricevute e risponde con un messaggio di richiesta DHCP che contiene i parametri di configurazione
4) Conferma DHCP: Risponde alla richiesta DHCP con un ACK DHCP che contiene i parametri di configurazione

Se il client vuole continuare ad utilizzare l'indirizzo IP che gli è stato assegnato precedentemente, DHCP offre la possibilità di rinnovare la connessione con lo stesso indirizzo IP.

NAT (Netword Address Translation) → è il processo di modificazione degli indirizzi IP dei pacchetti in transito su di un router durante la comunicazione tra host

ICMP (Internet Control Message Protocol)

Viene utilizzato da host e router per scambiarsi informazioni a livello di rete; è usato in genere per notificare gli errori. I pacchetti ICMP sono spesso considerati parte di IP e per questo vengono trasportati all'interno di datagrammi IP (come carico utile IP, esattamente come segmenti TCP e UDP).
I messaggi ICMP hanno un campo **tipo** e uno codice, i quali contengono rispettivamente l'intestazione ed i primi 8 byte del datagramma IP che ha generato il messaggio (così il mittente può determinare quale sia il datagramma che ha causato l'eventuale errore).
I due tipi di messaggi sono quelli:

- Per avvisare di **errori**: TTL esaurito oppure destinazione irraggiungibile
- Per portare **informazioni**: richieste/risposte echo

Da ricordare che, se ICMP stesso genera un errore, questo non viene segnalato in alcun modo.

INSTRADAMENTO

Instradare (routing) significa trovare il percorso ottimale attraverso i router della rete, ossia il percorso più a **costo** minimo per inviare il datagramma da un host A ad un host B. Da notare che non è detto che il cammino più breve (quello con un numero minore di hop) sia quello a costo minimo.
Ci sono due tipi di algoritmi di instradamento:

- **Globali**: hanno una conoscenza globale della rete e conoscono i costi di tutti i collegamenti. L'algoritmo riceve in ingresso tutti i collegamenti tra tutti i nodi della rete ed i relativi costi. Questi algoritmi sono anche chiamati **algoritmi a stato di collegamento (link-state algorithm)**, dal momento che conoscono lo stato dell'intera rete.
- **Decentralizzati**: il percorso di costo minimo viene costruito in modo distribuito ed iterativo; nessun nodo possiede informazioni sui costi dell'intera rete. Inizialmente i nodi conoscono solamente i costi dei collegamenti ad esso adiacenti e poi, attraverso un processo iterativo che comprende anche lo scambio di informazioni tra i nodi adiacenti, un nodo calcola gradualmente il percorso a costo minimo verso la destinazione. Gli algoritmi di questo tipo sono detti **algoritmi a vettore di distanza (distance vector algorithm)**, poiché ogni nodo elabora un vettore di stima dei costi (distanze) verso tutti i nodi della rete.

Un altro criterio per distinguere gli algoritmi d'instradamento è se essi sono:

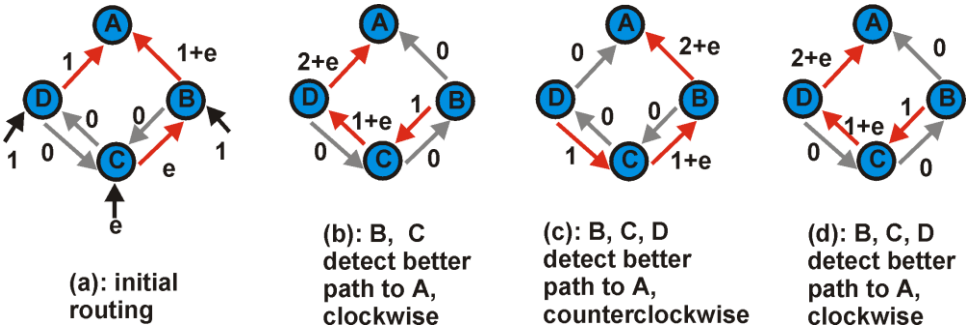
- **Statici**: i cammini cambiano molto raramente, spesso come risultato di un intervento umano (per esempio la modifica manuale di una tabella di inoltro di un router)
- **Dinamici**: determinano gli instradamenti al variare del volume di traffico (load-sensitive/load-insensitive) e della topologia della rete. Negli **algoritmi sensibili al carico** della rete, i costi dei collegamenti variano dinamicamente per riflettere il livello di congestione su di essi. Gli attuali algoritmi di instradamento sono però insensibili al carico.

ALGORITMO A STATO DI COLLEGAMENTO (Link-state)

In questo tipo di algoritmo, la topologia di rete e tutti i costi dei collegamenti sono noti e disponibili all'imput dell'algoritmo. Attraverso un algoritmo **link-state broadcast** vengono ottenuti tutti i costi, in modo da poterli utilizzare.
L'algoritmo LS principale è l'**algoritmo di Dijkstra** ed esso calcola il cammino a costo minimo da un nodo d'origine a tutti gli altri nodi in modo iterativo. Ha le seguenti proprietà:

- Dopo la N-esima iterazione, i cammini a costo minimo sono noti a N nodi di destinazione
- Tra i percorsi a costo minimo verso tutti i nodi destinazione, questi N cammini hanno gli N costi più bassi

Quando l'algoritmo LS termina, abbiamo per ciascun nodo il suo predecessore lungo il cammino a costo minimo dal nodo di origine. Per ciascun predecessore, abbiamo il rispettivo predecessore e così via. In questo modo riusciamo a costruire l'intero percorso dall'origine a tutte le destinazioni.
La **complessità** computazionale di questo algoritmo (ovvero quanti calcoli bisogna fare nel caso peggiore per calcolare tutti i cammini a costo minimo dall'origine a tutte le destinazioni possibili) è di **ordine quadratico**.



(a): initial routing

(b): B, C detect better path to A, clockwise

(c): B, C, D detect better path to A, counterclockwise

(d): B, C, D detect better path to A, clockwise

Questa immagine spiega meglio come funzioni un algoritmo link-state in un caso con 4 router.

ALGORITMO CON VETTORE DISTANZA (Distance Vector)

L'algoritmo DV ha le seguenti caratteristiche:

- **Distribuito**: ciascun nodo riceve parte dell'informazione da uno o più dei suoi vicini a cui è direttamente connesso
- **Iterativo**: questo processo si ripete fino a quando non avviene ulteriore scambio di informazioni tra i vicini
- **Asincrono**: non richiede che tutti i nodi operino al passo con gli altri

Prima di parlare dell'algoritmo, però, è necessario conoscere la formula di Bellman-Ford che mette in relazione i costi ed i percorsi a costo minimo:

$$d_x(y) = min_v \{c(x,v) + d_v(v)\}$$

dove $d_x(y)$ è il percorso a costo minimo dal nodo x al nodo y e min_v riguarda tutti i vicini di x. È abbastanza intuitiva: infatti se, dopo aver viaggiato da x a y, considero il percorso di costo minimo da v a y, il costo del cammino sarà $c(x,v) + d_v(y)$. Questa formula in pratica fornisce le righe della tabella di inoltro del nodo x.
Con l'algoritmo DV, ciascun nodo x mantiene i seguenti dati d'instradamento:

- Per ciascun vicino c, il costo $c(x,v)$ da x a v
- Il vettore distanza del nodo x
- I vettori distanza dei suoi vicini

In questo modo un nodo invia una copia del proprio vettore distanza a ciascuno dei suoi vicini e in seguito il vicino utilizza la formula di Bellman-Ford per aggiornare il proprio vettore distanza.
Perciò se il vettore distanza del nodo x è cambiato per via di un passaggio di aggiornamento di questo tipo, il nodo x manderà il proprio vettore distanza aggiornato a tutti i suoi vicini, i quali a loro volta aggiornano il loro. In questo modo le informazioni ottenute dal nodo sono i costi dei collegamenti verso i vicini direttamente connessi con e quelle ricevute da questi vicini.
Se non si verifica alcun cambiamento di costo nei collegamenti, l'algoritmo rimane in uno stato di quiescenza finché non cambia qualche costo.
La **complessità computazionale** dell'algoritmo DV è **variabile** (loop, counting to infinity).

I problemi che possono nascere con questo algoritmo al cambiamento del costo di un collegamento sono i seguenti:

- **Istradamento ciclico (loop)**: può accadere quando cambia il costo di un collegamento con uno più grande (per esempio da 1 a 40). Un loop in un instradamento assomiglia ad un buco nero, poiché un pacchetto destinato a x arriva a y o a z e questo rimbalzerà tra questi due nodi per sempre (a meno che non vengano cambiate le tabelle di inoltro ovviamente). Il ciclo proseguirà per un N iterazioni, finché N diventa maggiore della somma dei costi degli altri due collegamenti.
- **Counting to infinity**: conseguenza di un instradamento ciclico, quando si interrompe un collegamento. In questo modo A, adiacente a quel collegamento, vede che non può raggiungere B attraverso quel collegamento; ma C è ancora convinto che quel collegamento sia ancora funzionante (dal momento che non è a lui adiacente) e perciò indirizza A a B attraverso C. Questa situazione può propagarsi in tutta la rete, generando così stime di distanze sempre maggiori.

Per cercare di risolvere il problema del conteggio all'infinito, viene introdotto un numero massimo di hop pari a 15, lo **split horizon**, che consiste nell'omettere nelle tabelle di inoltro dei router i passaggi che ho "imparato" attraverso i vicini e l'**inversione avvelenata (poison reverse)**. L'idea di quest'ultima tecnica è semplice: quando un collegamento diventa non più utilizzabile per qualsiasi motivo, tutti i router nella rete

vengono informati di ciò e impostano nella loro tabella di inoltro la distanza come **infinito**. Questo fa sembrare i nodi sul collegamento non utilizzabile infinitamente distanti ed in questo modo nessun router indirizzerà pacchetti su quel collegamento.
La soluzione, purtroppo, non è definitiva dal momento che i cicli che non riguardano semplicemente due nodi adiacenti non verranno rilevati con questa tecnica.

Distance - Vector	Link - State
View network topology from neighbors perspective	Gets common view of entire network topology
Adds distance vectors from router to router	Calculates the shortest path to other routers
Frequent, periodic updates: Slow convergence	Event-triggered updates: Faster convergence
Passes copies of routing tables to neighbor routers	Passes link-state routing updates to other routers

Se i collegamenti cambiano costi, la regola generale è che le buone notizie viaggiano velocemente (ovvero quando il costo di un collegamento di abbassa), mentre quelle "cattive" viaggiano lentamente all'interno della rete.
Ecco un'immagine che mette a condronto le principali caratteristiche degli algoritmi LS e DV.
Presentiamo ora i principali protocolli di utilizzo reale per l'instradamento che utilizza Internet.

ESEMPIO DI DISTANCE VECTOR: RIP (Routing Information Protocol)

È stato uni dei primi protocolli Internet d'instradamento utilizzato generalmente nelle reti aziendali. È un protocollo DV che lavora in modo molto simile al protocollo DV idealizzato. **RIP** utilizza un conteggio degli hop come metrica per calcolare i costi; in altre parole, tutti i collegamenti hanno costo unitario. Il termine hop viene utilizzato per indicare il numero di sottoreti (passaggi) attraversate lungo il percorso minimo dal router d'origine a quello di destinazione.
I limiti di questo protocollo sono il **costo massimo** di un percorso è posto a **15 hop** (ciò confina RIP all'interno di reti con diametro inferiore a 15 hop; 16 equivale a infinito) e la convergenza lenta.
In questo protocollo, i router adiacenti si scambiano gli aggiornamenti d'instradamento circa ogni 30 secondi utilizzando un messaggio di risposta RIP; se un router non viene aggiornato per più di 3 minuti viene messo a distanza 16 (infinito) e risulta perciò irraggiungibile e dopo ulteriori 2 minuti questo viene eliminato dalle tabelle di instradamento.

ESEMPIO DI LINK STATE: OSPF (Open Shortest Path First)

Questo protocollo viene solitamente utilizzato negli ISP di livello superiore e usa anch'esso l'algoritmo di Dijkstra per determinare un albero delle rotte minime verso tutte le sottoreti. Al contrario del RIP, i costi dei collegamenti vengono impostati dall'amministratore e non impone nessuna politica di scelta dei costi. **OSPF** utilizza tre procedure per svolgere le proprie funzioni:

- **Hello** Protocol: messaggio che ha il compito di controllare che il collegamento con il router vicino sia operativo (il collegamento deve essere operativo in entrambe le direzioni)
- **Exchange** Protocol: serve per sincronizzare i database link state tra due router che hanno appena verificato la loro operatività (il messaggio link state update viene distribuito in flooding)
- **Flooding** Protocol: ovvero inonda di informazioni di stato del collegamento per aggiornare il database link state di tutta la rete a seguito del cambiamento dello stato di un link

Ogni qualvolta si verifica un cambiamento nei collegamenti (ad esempio cambia un costo), il router manda informazioni di instradamento via broadcast a tutti gli altri router della rete. In più, esso invia periodicamente ogni 30 secondi un aggiornamento dello stato della rete a tutti gli altri router, anche se nulla è cambiato: ecco perché è considerato "flooding".
Tra i vantaggi dell'utilizzo di OSPF ci sono la sicurezza (gli scambi tra i router possono essere autenticati), il supporto sia per l'instradamento unicast che multicast e il fatto che se ci sono due percorsi a con lo stesso costo, OSPF consente di utilizzarli senza doverne scegliere uno per trasportare tutto il traffico.

LIVELLO DI COLLEGAMENTO

L'obiettivo di questo livello è quello di fornire al livello di rete di due macchine adiacenti un **canale di comunicazione** il più affidabile possibile, offrendo dei servizi agli strati superiori. Il canale fisico a disposizione non è ideale: infatti ha degli errori di trasmissione, deve gestire la velocità di trasmissione dei dati e il ritardo di propagazione non è nullo.
Per trasportare i datagrammi lungo un canale di comunicazione, i protocolli di collegamento definiscono il formato dei pacchetti scambiati tra i due nodi; le unità dati del livello di collegamento sono chiamate **frame**.
Tra i protocolli di collegamento ricordiamo **Ethernet**, **802.11** wireless LAN (il wi-fi) ed il token ring.
I servizi che offrono sono il framing, l'accesso al collegamento, la consegna affidabile, il controllo di flusso, la rilevazione degli errori, la correzione degli errori e half/full duplex.

- Framing: i datagrammi di livello di rete vengono incapsulati all'interno di un frame, la cui struttura è specificata dal protocollo
- Accesso al collegamento: il protocollo che controlla l'accesso al mezzo (MAC) specifica le regole con cui immettere i frame nel collegamento
- Consegna affidabile: viene garantita la consegna senza errori di ciascun frame
- Controllo di flusso: tecniche per fare in modo che il trasmettitore non saturi il nodo ricevente
- Rilevazione degli errori: un nodo può ricevere erroneamente dei bit; gli errori sono causati dall'attenuazione del segnale o dal rumore elettromagnetico
- Correzione degli errori: il nodo ricevente determina il punto preciso del frame in cui si è verificato l'errore per poi correggerlo
- Half/full duplex: nella trasmissione full-duplex gli estremi del collegamento possono trasmettere contemporaneamente mentre non possono farlo in una connessione half-duplex.

Il livello di collegamento è implementato in un adattatore, noto anche come scheda di interfaccia di rete (NIC).

TECNICHE DI RILEVAZIONE E CORREZIONE DEGLI ERRORI

E' in genere un servizio fornito dai protocolli a livello di collegamento. Dal nodo trasmettitore vengono aggiunti dei bit detti **EDC** (error detection correction) ai dati da inviare. Questi vengono inviati all'interno di un frame al ricevente che legge una certa sequenza di dati e un certo EDC che possono essere diversi da quelli originali; se i dati coincidono, basandosi anche sull'EDC che gli è arrivato, allora non vi è presenza di errori nel frame.
Noi chiediamo se il nodo rileva l'errore, non se si è verificato: è possibile infatti che avvengano degli **errori** ma **non** vengono **rilevati** dal nodo ricevente, con tutte le conseguenze del caso.
Per poter rilevare gli errori nei dati trasmessi si utilizzano:

- **Controllo di parità**: è la tecnica più semplice ed è quella che utilizza un bit di parità. In pratica si aggiunge un bit alla fine dei dati da inviare in modo da rendere pari (o dispari) il numero totale dei bit a 1 nei bit trasmessi a seconda se si utilizza uno schema di parità pari o dispari. Il ricevente deve perciò solamente contare il numero di bit a 1 ricevuti e confrontarlo con lo schema di parità scelto. Un caso di errore non rilevato, ad esempio, sarebbe quello di un numero pari di errori all'interno dei dati, scegliendo uno schema di parità pari (caso che potrebbe tranquillamente accadere semplicemente con due errori durante la trasmissione)
- Somma di controllo (**checksum**): con questa tecnica i bit che compongono i dati vengono trattati come una sequenza di numeri interi, che vengono poi sommati. La somma che si ottiene viene utilizzata per controllare all'arrivo che questa somma e il complemento a 1 del checksum calcolata sui dati arrivati diano una somma pari a 1. Se non è così vuol dire che c'è stato un errore da qualche parte
- **Controllo a ridondanza ciclica** (CRC): è un metodo per calcolare il checksum nel quale i dati d'uscita sono ottenuti elaborando i dati in ingresso, i quali vengono fatti scorrere ciclicamente in una rete logica piuttosto semplice. Il CRC prevede la generazione di una stringa di bit di controllo che viene normalmente trasmessa assieme ai dati; il calcoli si basa sull'aritmetica modulare. Ogni codice CRC è definito dal suo polinomio generatore. Questa tecnica è molto utile per la rilevazione di errori casuali nella trasmissione dei dati.

IL SOTTO-LIVELLO MAC (Medium Access Control)

PROTOCOLLI DI ACCESSO MULTIPLO

Questi protocolli si utilizzano per coordinare l'accesso di più nodi trasmittenti e riceventi su un canale broadcast condiviso, ovvero il problema dell'accesso multiplo. È il problema analogo a quando ci si trova in una stanza in cui molte persone parlandosi ed ascoltandosi a vicenda: bisogna decidere chi e quando debba parlare, ovvero trasmettere nel nostro caso. I protocolli di accesso multiplo gestiscono appunto questa situazioni nelle reti, regolando le trasmissioni dei nodi sul canale condiviso. Dal momento che tutti i nodi sono in grado di trasmettere frame, il problema principale è quello della collisione, che si genera quando due o più nodi trasmettono contemporaneamente sullo stesso canale da loro condiviso. A causa della collisione nessuno dei nodi riceventi riuscirà a interpretare i frame che risultano "ingarbugliati" tra loro.
Possiamo così classificare questi protocolli: protocolli a suddivisione del canale, ad accesso casuale e a rotazione.

PROTOCOLLI A SUDDIVISIONE DEL CANALE

Il multiplexing può avvenire a divisione di tempo (TDM) e a divisione di frequenza (FDM): queste sono due tecniche che possono essere utilizzate per suddividere il canale.

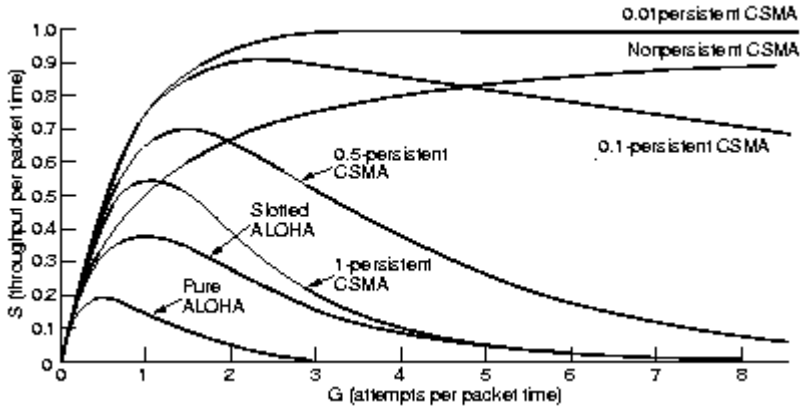
- **TDM** (Time Division Multiplexing): suddivide il tempo in intervalli di tempo e poi suddivide ciascun intervallo in slot (la dimensione degli slot viene in genere gestita in modo da consentire la trasmissione di un singolo pacchetto). TDM riesce ad evitare le collisioni ed è perfettamente imparziale nella suddivisione del tempo dedicato alla trasmissione di un nodo. Uno svantaggio degno di nota è che anche se un nodo non deve trasmettere nulla, ogni nodo deve aspettare il proprio turno in sequenza e non superare il limite medio di R/N bps.
- **FDM** (Frequency Division Multiplexing): mentre TDM suddivide il tempo in intervalli di tempo e vengono assegnati ad ogni nodo, FDM divide il canale in frequenze differenti e assegna a ciascuna frequenza un nodo. Quindi, se ho un canale a R bps e ho N nodi creo N canali a R/N bps l'uno. Però la lunghezza di banda è limitata a R/N, anche quando un nodo non deve trasmettere nulla (un po' come succede anche con TDM con gli intervalli)
- **CDMA** (Code Division Multiple Access): questo protocollo gestisce i nodi assegnando ad ognuno un codice ed in questo modo ogni nodo utilizza questo codice univoco per codificare i dati inviati. In questo modo, se i codici CDMA sono scelti con accuratezza, è consentito ai nodi di trasmettere simultaneamente ed ai rispettivi destinatari di ricevere correttamente i bit di dati codificati nonostante le interferenze che si creano con le trasmissioni in contemporanea di tutti i nodi. CDMA viene utilizzato particolarmente nelle reti wireless.

PROTOCOLLI AD ACCESSO CASUALE

Questa classe di protocolli concerne le situazioni in cui un nodo trasmette sempre alla massima velocità consentita dal canale. Quando si verifica una collisione, i nodi coinvolti ritrasmettono ripetutamente i loro frame fino a quando questi raggiungono la destinazione senza collisione. La ritrasmissione non è immediata, ma il nodo attende per un tempo casuale stabilito indipendentemente.
I protocolli di questo tipo sono molti, ma i più utilizzati sono:

- **Slotted Aloha**: richiede che tutti i nodi sincronizzino le loro trasmissioni a partire dall'inizio di uno slot. Se non si verifica una collisione, l'operazione ha avuto successo e quindi non occorre effettuare alcuna ritrasmissione. Al contrario, se avviene una collisione, il nodo la rileva prima del termine dello slot e ritrasmette con probabilità p il frame durante gli slot successivi fino a quando la ritrasmissione non avviene con successo.
- **Pure Aloha**: appena arriva un frame il nodo lo trasmette immediatamente e integralmente nel canale broadcast. Se un frame va in collisione, il nodo lo ritrasmette immediatamente con probabilità p, altrimenti attenderà il tempo di trasmissione del frame. Trascorso questo tempo, il nodo potrà ritrasmettere il frame con probabilità p.
- Per chiarire meglio: entrambi i protocolli aloha corrispondono ad un maleducato che continua a parlare senza preoccuparsi de fatto che altri stiano parlando tra loro. Ciò vuol dire che un nodo non presta attenzione al fatto che se un altro nodo sta trasmettendo né smette di trasmettere se lo sta già facendo un altro.
- **CSMA** (Carrier Sensitive Multiple Access): è un protocollo più "educato" in quanto segue due regole

principali: ascolta prima di parlare e se qualcun altro comincia a parlare, lui smette. La prima è detta rilevazione della portante: un nodo, prima di trasmettere, si mette in ascolto del canale e controlla che altri non stiano già trasmettendo a loro volta. Se è libero inizia a trasmettere, altrimenti aspetta un intervallo di tempo casuale e ripete questo passaggio. La seconda è detta rilevazione della collisione (**CSMA/CD**, **Collision Detection**): il nodo che sta trasmettendo resta comunque in ascolto del canale in cui sta inviando i dati e se rileva qualcun altro che inizia a trasmettere, arresta l'invio dei i propri dati. Determina in seguito quando potrà ricominciare a trasmettere.
Un'altra variante è il **CSMA/CA** (Collision Avoidance), nel quale si utilizzano delle tecniche per diminuire la probabilità di collisione, utilizzato soprattutto nelle reti wireless.
CSMA viene detto **p-persistente** (con una probabilità p che varia da 0 a 1) perché la probabilità che il nodo mittente sia pronto ad inviare è p, dal momento che è in continuo ascolto del canale per vedere se è libero o meno. CSMA 1-persistente invia solamente se il canale è completamente libero, mentre 0-persistente sarebbe quello ideale, ma andrebbe all'infinito.



In questa immagine vediamo il confronto sull'efficienza degli algoritmi appena presentati.

PROTOCOLLI A ROTAZIONE

Anche di questi protocolli ce ne sono a decine, ma i più importanti ed usati sono:

- **Polling protocol** (a sondaggio): uno dei nodi, designato come principale, sonda "a turno" gli altri. Il nodo principale manda un messaggio al nodo 1 dicendogli che può iniziare a trasmettere fino ad un massimo di tot frame. Dopo che 1 ha smesso di trasmettere, il nodo principale avvisa il nodo 2 che può iniziare la sua trasmissione di un tot di frame. Tutto questo continua con il nodo principale che sonda tutti gli altri nodi in modo ciclico. Questo protocollo elimina le collisioni e anche gli slot vuoti in cui nessun nodo trasmette e quindi ha un'efficienza più elevata. Introduce però un ritardo di polling, ovvero il ritardo che si impiega per notificare ad un certo nodo che può iniziare a trasmettere. L'inconveniente più serio è se il nodo principale si guasta: allora l'intero canale diventa inattivo.
- **Token-passing protocol** (a passaggio del testimone): qui non esiste un nodo principale (viene così eliminato un eventuale enorme problema), ma un frame, un messaggio detto token che circola fra i nodi della rete seguendo un ordine prefissato. In questo modo si informa un dato nodo che può iniziare a trasmettere. Questo protocollo è decentralizzato ed altamente efficiente, ma presenta il problema che il guasto di un nodo può mettere fuori uso l'intero canale, dal momento che questo nodo non è più in grado di far girare il messaggio di token.

INDIRIZZI A LIVELLO DI COLLEGAMENTO

Ogni nodo di una LAN ha il proprio indirizzo di collegamento. Il protocollo di risoluzione degli indirizzi (**ARP**) fornisce il meccanismo per trasformare gli indirizzi IP in indirizzi a livello di collegamento.
Gli **indirizzi MAC** identificano più correttamente le schede di rete di ogni PC; esso è lungo 6 byte (2^{48} indirizzi MAC disponibili) e viene generalmente espresso in notazione esadecimale. Come per gli indirizzi IP, anche gli indirizzi MAC sono **univoci**: non esistono due adattatori con lo stesso indirizzo. La sua struttura, al contrario di quella dell'indirizzo IP, è piatta (non gerarchica) e non cambia mai con il tempo (è analogo al codice fiscale di una persona).

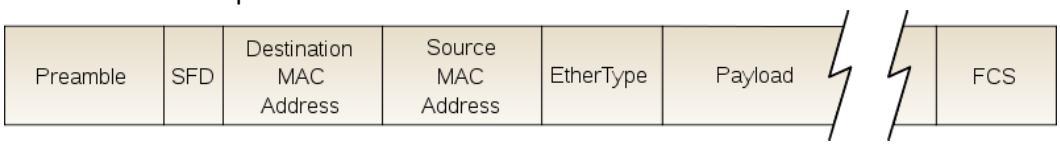
ARP (Address Resolution Protocol)

Dato che esistono sia indirizzi IP (livello di rete) che indirizzi MAC (livello di collegamento), serve un protocollo che trasformi i primi nei rispettivi secondi. In pratica, il nodo trasmettitore, per determinare l'indirizzo MAC del nodo destinazione con un certo indirizzo IP, deve utilizzare ARP. Possiamo dire che, in un certo senso, è simile al DNS (che traduce i nomi degli host nei rispettivi indirizzi IP). ARP, però, risolve solamente gli indirizzi

IP per i nodi che sono nella stessa sottorete.
Nella RAM dei nodi vi è la **tabella ARP** che contiene la corrispondenza tra indirizzi IP e MAC e anche un **tempo di vita** (TTL) che indica quando bisognerà eliminare una certa voce dalla tabella. Da notare è che la tabella non contiene necessariamente una voce per ciascun nodo della sottorete, dal momento che alcuni nodi possono essere stati cancellati o mai inseriti.
Il nodo trasmettitore, per conoscere l'indirizzo MAC del destinatario, invia un **pacchetto ARP** con molti campi al suo interno, compresi quelli degli indirizzi IP e MAC del mittente e del destinatario; la sua funzione è quella di interrogare tutti i nodi della sottorete riguardo all'indirizzo MAC corrispondente all'indirizzo IP da risolvere.
I pacchetti ARP di richiesta e di risposta hanno lo stesso formato con i campi da "completare". Il pacchetto di richiesta viene perciò inviato in broadcast, mentre il messaggio di risposta viene inviato in un frame standard solamente al nodo che ha inviato il messaggio di richiesta.
ARP è considerato un protocollo **plug-and-play** poiché la tabella ARP di un nodo si costruisce automaticamente e non deve essere configurata da nessuno.

ETHERNET

È lo standard IEEE 802.3 e viene utilizzato sin dagli anni '70 per le reti LAN cablate. La sua vasta diffusione è dovuta al costo basso di questi cavi e anche alla loro relativa comodità d'uso. Con il passare del tempo, ovviamente, sono state proposte versioni sempre più veloci.
Con questa tecnologia, gli host ed i router sono direttamente connessi ad un **hub** con cavi di doppino intrecciato. Un hub è un dispositivo a livello fisico che agisce sui singoli bit, rigenerandoli identici come sono entrati amplificando la potenza del segnale e lo trasmette su tutte le sue interfacce (chiunque sia connesso ad una delle interfacce dell'hub riceve una copia identica del dato). Gli **switch** lavorano sempre a livello 2, sono dei dispositivi che filtrano ed inoltrano pacchetti a livello di collegamento all'interno delle reti locali attraverso gli indirizzi MAC.
La trama Ethernet è così composta:



- Preambolo: serve per "svegliare" gli adattatori del ricevente e a sincronizzarlo con quelli del mittente
- SDF (start frame delimiter): determina la fine del preambolo e l'inizio del frame ethernet; è protetto tramire la codifica Manchester
- Indirizzo MAC di destinazione e di sorgente
- Ether type: determina il tipo di protocollo di rete utilizzato durante la trasmissione oppure la lunghezza del campo dati in Ethernet
- Payload: da 46 byte a 1500 byte e contiene i dati reali da trasmettere; è di lunghezza variabile in base all'MTU della tipologia di ethernet che si utilizza
- FCS (frame check sequence): il controllo di ridondanza ciclica (CRC) che permette di rilevare se ci sono stati errori di trasmissione; il ricevente calcola il CRC con un algoritmo e confronta che sia uguale a quello contenuto in questo campo (che è stato calcolato dal trasmettitore)

Ethernet fornisce un servizio **senza connessione** e **non affidabile**: non utilizza nessun handshake preventivo con il destinatario e il frame ricevuto viene solamente sottoposto al controllo CRC, ma non invia nessun riscontro del risultato di tale controllo. Infatti, se il controllo non va a bon fine, il frame viene semplicemente scartato senza avvisare nessuno.

CSMA/CD IN ETHERNET

Quando i nodi di una LAN ethernet sono connessi ad un hub si ha una LAN broadcast: quando uno di questi trasmette un frame, tutti nodi lo ricevono. Per questo serve un protocollo ad accesso multiplo. Ethernet utilizza **CSMA/CD** che:

- Non utilizza slot temporali
- Effettua la rilevazione (ascolto) della portante
- Rileva le collisioni
- Prima di ritrasmettere, l'adattatore resta in attesa per un tot di tempo stabilito arbitrariamente

La sua efficienza si approssima al 100% ed è utilizzato senza alcuna forma di coordinazione con gli altri adattatori. Il funzionamento del protocollo in un caso tipico è il seguente:

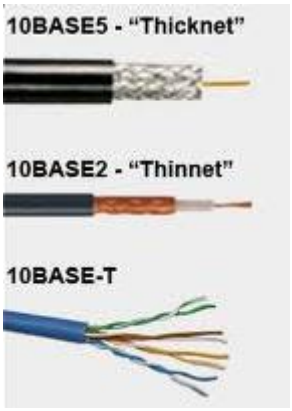
- L'adattatore prende il datagramma dal livello di rete, lo incapsula in un frame ethernet e lo mette in un bufferr
- Quando "sente" che il canale è inattivo inizia la trasmissione del frame; se invece è occupato, resta in attesa fino a quando non rileva più il segnale
- Durante la trasmissione controlla che altri segnali provenienti da altri adattatori non disturbino la trasmissione
- Se invece rileva segnali provenienti da altri adattatori, interrompe immediatamente la trasmissione e trasmette un segnale di disturbo (serve ad avvisare della collisione tutti gli adattatori in fase di invio in quel momento)
- Dopo aver annullato la trasmissione e trasmesso il segnale di disturbo, l'adattatore entra in una fase di **attesa esponenziale (expoential backoff).** L'elgoritmo di backoff esponenziale non fa altro che calcolare l'intervallo che l'adattatore deve aspettare prima di cominciare a trasmettere di nuovo.

Collision domain → è quella porzione di rete ethernet in cui, se due stazioni trasmettono simultaneamente, avviene una collisione delle trame. Il diametro del collision domain è la distanza massima tra ogni possibile coppia di stazioni (es: il collision domain di un ethernet a 10 Mbps è di 2.8 km)

Ethernet, quando c'è un solo nodo che deve inviare dati, può trasmettere alla massima velocità consentita (10/100/1000 Mbps); se invece ci sono più nodi che vogliono trasmettere in contemporanea, la velocità diminuisce di molto.

TIPOLOGIE ETHERNET

Ethernet compare in molte forme differenti con svariate denominazioni. Con il termine **"base"** si intende la banda base di ethernet, ovvero 10/100 Mbps (**Fast Ethernet**) e 1/10 Gbps (**Gigabit Ethernet**); la parte finale dell'acronimo rimanda al mezzo fisico utilizzato. La cosa che è rimasta immutata tra tutte le versioni di ethernet è il formato della trama. Ecco le diverse tipologie:



A bus o ad albero di bus:

- 10 BASE 5: cavo coassiale spesso, lunghezza massima cavo coassiale pari a 500 m
- 10 BASE 2: cavo coassiale sottile, lunghezza massima cavo coassiale pari a 2800 m

A stella:

- 10 BASE T: doppino UTP (unshielded twisted pair), lunghezza massima cavo di 100 m, connettori RJ45
- 10 BASE FB: fibra ottica monomediale
- 10 BASE FP: fibra ottica monomediale

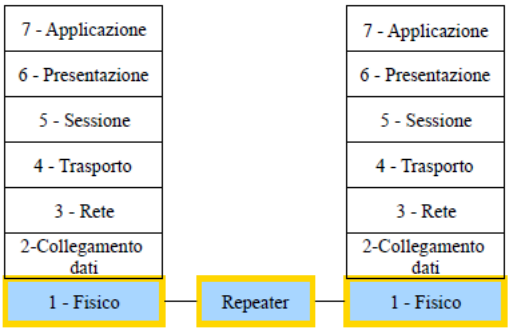
PPP (Point-to-point Protocol)

È un protocollo a livello di collegamento che funziona appunto per **canali punto-punto** ed è senza dubbio il protocollo di questo livello più utilizzato. Esso collega direttamente i due nodi agli estremi del collegamento e le sue caratteristiche principali sono:

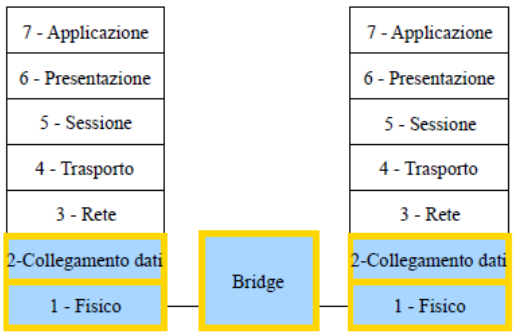
- Framing dei pacchetti
- Trasparenza
- Rilevazione degli errori
- Supporta vari protocolli di livello superiore (rete)
- Autenticazione del "chiamante"
- Negoziazione degli indirizzi IP di rete
- **Semplicità** (non implementa infatti la correzione degli errori, il controllo di flusso e il collegamento multi-punto)

LAN ESTESE

Per necessità si è dovuto scegliere dei mezzi condivisi sia per necessità (wireless) che per motivi economici. Una tipica LAN altro non è che una serie di stazioni (PC) connessi ad un segmento di caso (bus); questo segmento non può essere troppo lungo, altrimenti il segnale si attenua troppo. Perciò è sorto il problema di come **estendere le LAN**. Ciò viene fatto, in ordine di complessità, attraverso repeater o hub, bridge e switch.



Repeater e Hub: replica le trame in arrivo da un segmento ad un altro, amplificando il segnale. I repeater possono connettere più di due segmenti ed in questo caso si parla di hub. Il dominio di collusione coincide con quello di broadcast ed il problema è proprio questo, un dominio di collisione molto esteso



Bridge: al contrario dei repeater/hub, il bridge aumenta le prestazioni della rete. Esso collega due segmenti (ha SOLO due porte) di rete ed è un dispositivo dotato di backward learning (intelligenza in un certo senso): infatti seleziona se ripetere una trama o meno in base ad una tabella che esso mantiene dove c'è sctitto quale stazioni fanno parte di ciascun segmento. Il vantaggio è che spezza il dominio di collisione.

Switch (di livello 2): non è altro che un bridge multiporta; spesso ogni porta è collegata ad un'unica stazione invece che ad un segmento di rete.

Le più diffuse sono ormai le **Wireless LAN**, nelle quali il canale trasmissivo non è un cavo coassiale o una fibra ottica, bensì l'aria che ci circonda. Le sue componenti principali sono il Basic Service Set (BSS), che consiste in un certo di numero di stazioni con lo stesso protocollo MAC che condividono lo stesso mezzo trasmissivo, e l'**access point**, che funge da ultimo hop ed ha le stesse funzioni di un brigde, solamente che non utilizza dei cavi ma delle frequenze d'onda (di solito serve un'identificazione per utilizzarlo, al contrario di un bridge ethernet).
Le WLAN sono lo standard **802.11** (a/b/g/n, in base alla frequenza nella quale si trasmette e hanno velocità differenti [vedi figura sotto]).

	802.11b	802.11g	802.11a
Max rate (Mbps)	11	54	54
Modulation Type	CCK	CCK, OFDM	OFDM
Data Rates	1, 2, 5.5, 11	1, 2, 5.5. 11, 6, 9, 12, 18, 24, 36, 48, 54	6, 9, 12, 18, 24, 36, 48, 54
Frequency	2.4-2.497GHz	2.4-2.497GHz	~5GHz

I problemi a cui si deve badare maggiormente in questo tipo di reti locali sono l'attenuazione del segnale, l'interferenza da parte di altre sorgenti (il rumore, non necessariamente altri router wifi) e il cosiddetto "effetto multipath", ovvero la propagazione su più cammini del segnale radio.
Il protocollo a livello di collegamento ad hoc per le WLAN è il **CSMA/CA**, nel quale si tenta di mantenere la probabilità di collisione al di sotto di un certo valore, dal momento che il mezzo di trasmissione è unico per tutti (l'aria), nonostante ci sia la possibilità di utilizzare diverse frequenze.

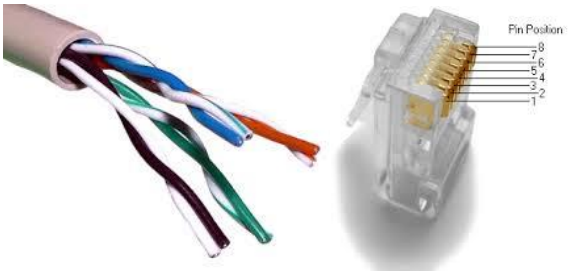
IL LIVELLO FISICO

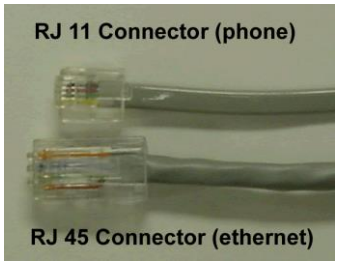
È il livello più basso per il trasporto delle informazioni. La maggior parte dei mezzi trasmissivi odierni si basa sulle seguenti categorie di supporti: **elettrici** (doppino, cavo coassiale), **ottici** (fibra ottica, raggi laser) e **radio** (non richiedono un supporto cablato, ponti radio, satelliti, reti cellulari).
Il mezzo trasmissivo ottimale è caratterizzato da una bassa, deve avere una buona resistenza alla trazione e una buona flessibilità; in altre parole non deve rompersi facilmente.
Le caratteristiche principali dei mezzi trasmissivi elettrici sono:

- Impedenza (in funzione della frequenza): mette assieme resistenza e distorsione del segnale
- Velocità di propagazione del segnale
- **Attenuazione**: aumenta linearmente con la distanza (su cavo) e con il quadrato della distanza (wireless)
- **Diafonia**: misura il disturbo indotto da un cavo vicino e crea rumore sulla linea, disturbando la comunicazione; cresce con la distanza fino a stabilizzarsi

IL DOPPINO

È detto anche **coppia** (pair) ed è il mezzo trasmissivo classico della telefonia. È composto da due fili di rame ritorti (twisted) e si utilizza una trasmissione differenziale per ridurre ulteriormente le interferenze elettromagnetiche. I vantaggi del **doppino** sono i costi ridotti e l'installazione molto semplice.

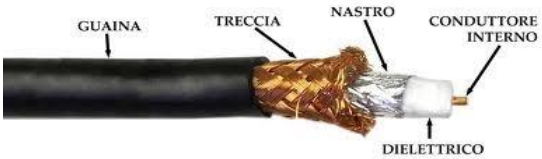




Con i doppini si utilizza il connettore **RJ-45** (figura in centro) a 8 pin ed è utilizzato per il cablaggio di reti locali con lo standard Ethernet 802.3 (la linea telefonica utilizza il connettore RJ-11).
Esistono due tipi di doppino: quella schermata e quella non schermata, detta anche UTP (unshielded twisted pair). UTP è suddiviso in categorie, in base alla sua qualità e alla velocità massima con la quale riesce a trasportare i dati.

IL CAVO COASSIALE

È il cavo che utilizziamo per collegare la TV all'antenna. Questo è un mezzo trasmissivo composto da un **connettore centrale** e una o più **calze** di schermature: maggiore è la schermatura dai disturbi esterni e minori saranno le interferenze. Da notare che non avviene alcuna irradiazione di segnale dal cavo verso l'eterno dello stesso.
Il cavo coassiale, però, ha dei costi più elevati di quelli del doppino e presenta delle maggiori difficoltà di installazione, nonostante abbia delle velocità trasmissive abbastanza alte.
Anche qui ci sono due tipologie: il cavo oscilloscopio (RG-58, utilizzato per cablaggi ethernet) e il cavo TV (RG-59).

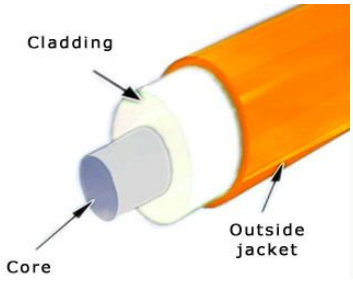


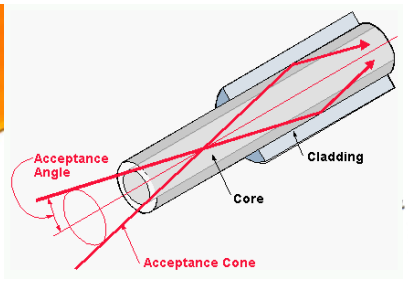
Il cavo coassiale viene molto utilizzato negli USA per la trasmissione della televisione nazionale e si utilizza lo stesso cavo anche per internet.
Nella figura a lato vediamo come è fatto uno di questi cavi.

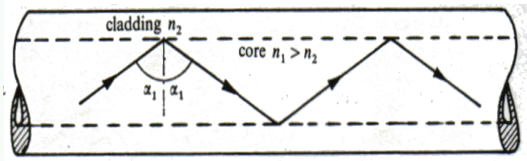
LA FIBRA OTTICA

La fibra ottica è una guida d'onda che porta luce e confina questo segnale luminoso all'interno di un filo di vetro. È composta da due parti, il **core** ed il **cladding**, con due indici di rifrazione diversi; per la **legge di Snell**, il raggio luminoso (che può essere generato da un LED o da un laser) introdotto nella fibra entro un angolo di accettazione rimane confinato nel core grazie alla riflessione totale dell'energia portata dal raggio

luminoso. Per questo motivo, la potenza del segnale in ingresso viene trasferita senza perdite di potenza all'uscita.







In queste immagini vediamo la composizione di una singola fibra ottica e una spiegazione visiva della legge si Snell.

I vantaggi della fibra ottica sono numerosi: è immune ai disturbi elettromagnetici e alla diafonia, ha un'alta capacitò trasmissiva, ha una bassa attenuazione del segnale (non è un mezzo trasmissivo ideale; perde un tot di db/km di potenza del segnale) ed ha dei costi contenuti. Al contrario, però, è adatta solamente a collegamenti punto-punto, i connettori delle fibre sono difficili da collegare tra loro e ha un ridotto raggio di curvatura (per via del filo di vetro che la compone).

IL CANALE TRASMISSIVO RADIO

La propagazione del segnale nell'aria che ci circonda viene disturbata dagli ostacoli naturali. Può avvenire infatti la riflessione del segnale su cammini multipli, ma anche il **fading** (significa letteralmente "sparire"; è una variazione veloce dell'ampiezza del segnale dovuta all'auto-interferenza con la riflessione del segnale stesso; interferenza costruttiva/distruttiva) e lo **shadowing** (come il fading, solo con una variazione lenta del segnale).
Il principale problema è l'**interferenza** causata da altri segnali e l'**attenuazione** dello stesso, che avviene con il quadrato della distanza in condizioni ottime.
Si suddivide in due parti:

- Rete di **accesso**: comprende gli apparati ed i mezzi trasmissivi che collegano l'utente con il nodo di accesso (la centrale telefonica) del gestore dei servizi di telecomunicazione; il cosiddetto "ultimo miglio" per arrivare alle nostre abitazioni
- Rete di **backbone**: comprende gli apparati e i mezzi trasmissivi appartenenti ad uno o più gestori di telecomunicazione e destinati al solo transito di fonia/dati tra i nodi di accesso

ACCESSO POTS (Plain Old Telephone Service): IL MODEM

Il **modem** (MOdulatore e DEMulatore) si utilizza per effettuare trasmissioni seriali si una rete telefonica pubblica e trasforma il segnale da **digitale** ad **analogico** e viceversa. In pratica rende il segnale idoneo per essere trasmesso sulla rete telefonica pubblica su banda fonica.

DTE → apparato utente (PC)
DCE → apparato di rete (modem)

Il bitrate massimo (capacità del canale trasmissivo) per i modem POTS è dato dalla **formula di Shannon**:

$$C = B\ log_2(1 + S/N\ )$$

dove C è la capacità del canale (bit/s), B è la banda (Hz), S il segnale (J) e N il rumore (J). Capiamo da questa formula che il canale attenua e aggiunge rumore e nient'altro.

Per la linea telefonica tradizionale il bitrate è di 34860 bit/s. I modem a 56 kbit/s non violano assolutamente questa formula, bensì sopprimono il filtro fonico in download, consentendo l'uso di una banda più larga.

ACCESSO DSL (Digital Subscriber Line)

Il DSL è una famiglia di tecnologie che fornisce un servizio dati ad alta velocità sulla rete di accesso, chiamate anche xDSL. La più diffusa è l'**ADSL** (Asymmetric DSL) e ha una velocità maggiore sia in downstream che in upstream rispetto al semplice DSL. Le velocità massime teoriche per l'ADSL sono riportate nella tabella seguente:

	ADSL	**ADSL2**	**ADSL2+**
Downstream	6 Mb/s	8 Mb/s	24 Mb/s
Upstream	1.5 Mb/s	3.5 Mb/s	3.5 Mb/s

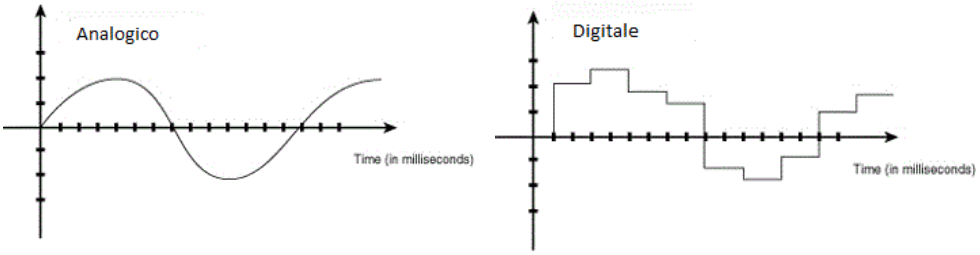
Gli apparati utente utilizzati dall'ADSL sono:

- Filtro **splitter**: serve per evitare che l'energia utilizzata per trasmettere i dati crei rumore nella linea della voce. Di fatto separa il segnale vocale dai dati
- **Modem**: serve a (de)modulare il segnale alle frequenze opportune, dando due frequenze diverse a downstream e upstream

CODIFICHE DI LINEA

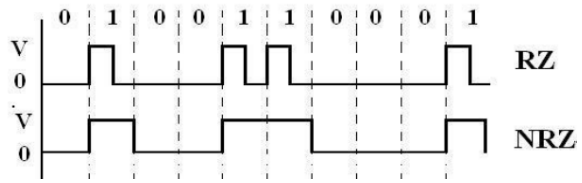
Le tecniche di rappresentazione dell'informazione sono principalmente due:

- **Analogiche**: varia con continuità in ampiezza; potenza del continuo (sinusoide)
- **Digitali** (numeriche): è una sequenza di punti discreti su delle ampiezze prestabilite; non ha la potenza del continuo dell'analogico; è un campionamento nel tempo e quantizzato nell'ampiezza (onde quadre)



I tipi di codifica principali sono i seguenti:

- **Unipolari**
  - Semplici e "primitive"
  - Ha due soli livelli di tensione, uno per 0 e uno per 1 (solitamente tensione nulla per 0 e tensione positiva per 1)
  - Problemi: perdita di sincronismo se trasmetto lunghe sequenze dello stesso simbolo; in mezzi ottici, lunghe sequenze di 1 (luce) può portare al sovraccarico del LED/laser di trasmissione
- **Polari**
  - Utilizzano due livelli di tensione con polarità diverse (si riduce quasi del tutto la componente continua)
  - Sottotipi: **NRZ** (non-return-to-zero; non c'è transizione su tensione nulla nel passaggio tra due bit consecutivi), **RZ** (return-to-zero; transizione su tensione nulla tra due bit consecutivi) e **Bifase** (ogni bit viene rappresentato da due livelli di tensione con polarità inversa; codifica Manchester)
  - Sono le più utilizzate perché sono migliori nel recuperare il sincronismo



- **Bipolari**
  - Si usa la tensione nulla per lo 0 e due polarità opposte per l'1, usate in alternativa
  - Permettono l'uso di tre simboli: -1, 0, +1

- **nBmB**
  - I simboli di *n* bit sono rappresentati da simboli di *m* bit, con *n* < *m* (es: 4B5B, 8B10B)
  - Mappa gruppi di *n* bit in gruppi di *m* bit
  - Molto popolari perché richiedono meno banda delle codifiche polari, forniscono dei caratteri speciali per la delimitazione dei pacchetti e forniscono più transizioni possibili, limitando la componente continua

TECNICHE DI DE-MODULAZIONE

La **modulazione** è l'operazione di mappatura dei bit su simboli analogici da trasmettere sul mezzo fisico.
Le codifiche di linea sono selle semplici modulazioni in **banda base** (rappresentazione vicino all'origine), utilizzate soprattutto nei collegamenti cablati punto-punto a bassa velocità, nella quale le frequenze vengono lasciate inalterate.
Le fibre ottiche, invece, utilizzano delle modulazioni in **banda traslata** (rappresentazione traslata dall'origine), che consentono la multiplazione in frequenza di diversi canali. In questo modo, il segnale viene modulato opportunamente per far sì che la banda occupata sia quella prevista. Tale operazione può essere fatta solamente in banda traslata.
Queste tecniche utilizzano diverse grandezze fisiche per supportare l'informazione: la **frequenza**, l'**ampiezza** e la **fase**.

TRASFORMATA DI FOURIER

Un **segnale trasmissivo** s(t) è una sequenza di simboli x_i nel tempo, ovvero $s(t) = \sum_i x_i\ (t - iT)$, dove T è l'intervallo di segnalazione. Ogni segnale è una funzione del tempo che può avere diverse forme d'onda. Un segnale s(t) nel tempo ha un equivalente S(f) nel dominio della frequenza, chiamato **spettro** del segnale.
Con **banda** di un segnale è l'intervallo di frequenze che vanno a formare il segnale (l'ampiezza dello spettro). La banda di un canale è legata alla banda dei segnali di cui consente la trasmissione.
La **trasformata di Fourier** serve per rappresentare quei segnali per i quali non sussiste una struttura periodica ed è un operatore funzionale che, applicato ad un segnale definito nel dominio del tempo, ne individua un altro nel dominio della variabile frequenza. La trasformata mette in relazione il segnale s(t) con il suo spettro S(f).