**1 L’Internet**

* 1. **Le origini di Internet**

Le origini di internet risalgono al lancio del primo satellite nello spazio: lo **Sputnik** (4/10/1957), a seguito del quale numerosi investimenti vennero riversati da parte dell’America nel settore spaziale, cominciando così la cosiddetta corsa allo spazio.

Nasce quindi **ARPANET:** la prima rete che si basava sull’informazione a pacchetto, creata con lo scopo di avere una rete flessibile.

* Negli anni **’80** nascono i **protocolli** (SMTP, TCP/IP).
* Negli anni **’90** nasce **www** (world wide web, ad opera di Tim Berners-Lee).
* Negli anni **’00** nascono i **social**.
  1. **Cos’è Internet?**

L’internet non è descrivibile sotto una singola definizione, bensì lo possiamo spiegare identificandolo come:

* Un’infrastruttura fisica
* Un’architettura di rete
* Un servizio di comunicazione (utilizzato da applicazioni e protocolli di sistema)

Le sue **componenti fisiche** si dividono in:

* **Host** (terminali), che permettono di inviare e ricevere informazioni.
* **Link** (collegamenti), differiscono per tipo e performance.
* **Router** (nodi), operano sui pacchetti.
* **Altri nodi** (switch, modem…).

**1.3 Le reti**

**1.3.1 Le reti: come si dividono e da cosa sono formate**

Le reti si possono dividere per area coperta in:

* **LAN**, Local Area Network
* **MAN**, Metropolitan Area Network
* **WAN**, Wide Area Network

Una rete è organizzata in:

* **ISP**, ossia un Internet Service Provider (Google, Amazon…).
* **BSP**, un Backbone Service Provider (un collegamento ad alta velocità tra due server).
* **NAP**, ossia Network Access Point (un dispositivo di rete che consente l’accesso wireless).
* **POP**, un Point Of Presence (un punto di accesso alla rete).
* **CN**, il Customer Network.

**1.3.2 Come può essere trasferita l’informazione?**

Attraverso **la commutazione di circuito**, dove le informazioni vengono scambiate attraverso un circuito (es. rete telefonica). Nessun’altra chiamata in parallelo può usare le risorse della chiamata in attivo, ne può avvenire solo una per circuito.

Questa commutazione è stabile, garantisce la qualità del servizio, perché le risorse non possono essere tolte a metà chiamata, ciò fornisce anche uno svantaggio in quanto le risorse rimangono inattive se non utilizzate e quindi non avviene la connessione.

Il secondo svantaggio è quello di dover “attaccare” il circuito attraverso un **protocollo di segnalazione** che permetta alla rete di cercare, trovare e dedicare le risorse necessarie per la chiamata.

La capacità dei canali in ingresso è pari a quella dei canali in uscita.

Attraverso **la commutazione di pacchetto**, dove i dati vengono inviati in rete con messaggi suddivisi in un certo numero di bit.

L’informazione viene suddivisa, in fase di **commutazione di pacchetto**, in diversi pacchetti formati da: un **header**, chiamati anche bit di informazione: bit aggiuntivi che contengono tutte le informazioni necessarie affinché l’Internet sappia instradare il pacchetto stesso, e **l’informazione.**

Ogni pacchetto contiene informazioni **sull’indirizzo di destinazione** e sul prossimo nodo, le risorse sono condivise: pacchetti di informazioni diverse possono percorrere la stessa strada (a differenza della commutazione di circuito).

Tra gli svantaggi troviamo il dover aggiungere dei bit nella suddivisione dei pacchetti, e il rischio di perdere pacchetti o che questi arrivino in modo confusionario.

La commutazione di pacchetto inoltre non garantisce la stabilità della trasmissione (perché dipende dal numero di utenti che usano lo stesso percorso).

Inoltre, il router deve avere una coda di entrata e uscita che cambi il modello o il nodo.

Questo modello di nodo si chiama **packet switch/router:**

* L’arrivo dei pacchetti è **asincrono**.
* La capacità dei collegamenti è **arbitraria**.
* Possono esserci **conflitti** temporali per la trasmissione.
* Serve **memorizzare temporaneamente** (in quella che si chiama coda) all’ingresso per analizzare l’indirizzo di destinazione e all’uscita per gestire i conflitti.

Prima di decidere cosa fare con un pacchetto avvengono uno **store and forward**, ossia il commutatore deve ricevere l’intero pacchetto per poi poterlo trasmettere (**cut-thorugh** avviene invece quando il pacchetto viene ritrasmesso alla completa ricezione dell’header).

Il **ritardo di ogni pacchetto** è una variabile casuale determinata dalla **multiplazione statistica**, ossia l’accodamento dei pacchetti in attesa dell’utilizzo del collegamento.

Esistono due versioni della commutazione di pacchetto: **Datagram** (quella sopra descritta) e la **commutazione a circuito virtuale**.

* Nella commutazione **datagram** la scelta della porta d’uscita viene fatta sulla base del solo indirizzo di destinazione (i pacchetti dello stesso flusso vengono inoltrati indipendentemente).
* Nella **commutazione a circuito virtuale** i nodi identificano i **pacchetti di un flusso informativo** sulla base di un **identificativo di circuito virtuale** (CVI o label), il circuito virtuale viene instaurato in una fase di setup prima della fase dati e dopo la fase setup **i pacchetti  
  seguono tutti lo stesso percorso** di rete perché instradati sulla base dell’identificativo di circuito virtuale.

**1.3.3 Come si misura la capacità di una rete?**

Delle prestazioni di una rete possiamo descrivere la **velocità di trasmissione**, il **tempo di trasmissione**, il **ritardo di propagazione** e i **tempi di attraversamento del canale.**

La **velocità di trasmissione** è la velocità (rate) **R** con la quale l’informazione digitale viene trasmessa su una linea; questa viene misurata in bit/s (bps).

Il **tempo di trasmissione** **T** per trasmettere **L** bits dipende dalla velocità di trasmissione **R**

Il **ritardo di propagazione** è il tempo **τ** che occorre affinché il singolo bit che parte da un router arrivi al

suo prossimo destinatario.

Il **tempo di attraversamento del canale** è il ritardo tra la trasmissione del primo bit e la ricezione

dell’ultimo.

**1.3.4 Come è composto un router?**

Il **router** è formato da:

* Un **packet input**.
* Un **header processing** (diviso in routing table lookup e DMA transaction).
* Un **packet output.**

Per questo in realtà prima che avvenga la trasmissione di un pacchetto dobbiamo calcolare un piccolo ritardo, chiamato **process time**.

**2 Il livello fisico**

**2.1.1 I tipi di segnale**

I segnali possono essere **digitali** (discreti), composti ossia da stringhe di bit o **analogici** (continui), ossia segnali associati a grandezze fisiche.

Mentre un segnale digitale, proveniente da una sorgente discreta viene subito distribuito al **modulatore** che si occupa della sua trasmissione, e rilascia quindi il segnale modulato, un segnale analogico, proveniente da una sorgente continua deve prima passare per un **campionatore e quantizzatore**.

**2.1.2 L’analisi di Fourier**

Grazie a questa possiamo trasformare un segnale a tempo continuo in uno discreto

La trasformata di Fourier generalizza la serie di Fourier anche per segnali non periodici.

* Per rappresentare appieno un segnale non periodico servirebbero infinite armoniche.
* Il risultato è una funzione continua ma con le armoniche più importanti al “centro” delle frequenze.
* Si usa una banda specifica in base al contenuto informativo.

**2.1.3 Campionamento e quantizzazione**

Mentre la rappresentazione umana della realtà è continua, e si rifà al mondo analogico, gli **elaboratori** numerici gestiscono **informazione** **discreta**.

È quindi necessario trasformare i segnali analogici in un loro equivalente digitale.

Per farlo:

* Il segnale analogico a **banda illimitata**, tempo e ampiezza continui passa attraverso un **filtro**.
* Il nuovo segnale analogico a **banda limitata** e **tempo e ampiezza continui** passa attraverso un **quantizzatore**.
* Il segnale analogico ora a **tempo e ampiezza discreti** e banda limitata può quindi essere trasmesso al **codificatore**.

**2.1.3.1 Il Campionamento**

Secondo il **teorema di Nyquist** un segnale del tempo è completamente determinato dai suoi campioni presi a distanza **T** tale che **T**  dove **B** è la banda del segnale, o usando la **frequenza di campionamento**  tale che:

I campioni presi alla frequenza di Nyquist rappresentano il contenuto informativo del segnale,

campioni più frequenti non sono indipendenti, mentre quelli meno frequenti “perdono informazione” (e il segnale non è più ricostruibile esattamente).

Il contenuto informativo viene completamente preservato se si utilizza una frequenza di campionamento maggiore della frequenza di Nyquist.

La **banda** rappresenta il contenuto informativo.

Ogni segnale analogico di banda B può essere ricostruito interamente in base ai suoi campioni presi a frequenza 2B, e ciò avviene con un **filtro** posto al ricevitore che taglia le frequenze oltre 2B.

**2.1.3.2 La quantizzazione**

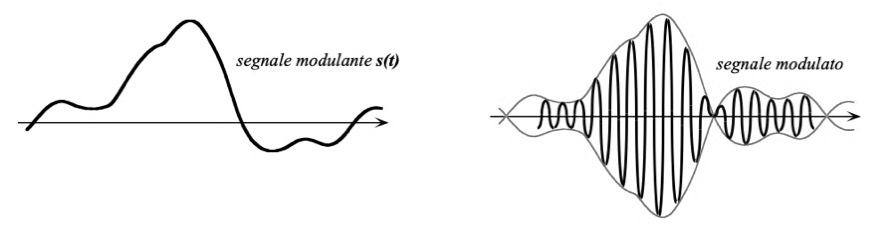
È l’operazione con cui una grandezza continua è trasformata in discreta, nella trasformazione si commette un errore di approssimazione (quantizzazione), più livelli si hanno meno è l’errore di quantizzazione.

**2.1.3.3 La modulazione**

La modulazione di un segnale può avvenire in **banda base** o in **banda passante**.

In **banda base** i segnali usati nella modulazione hanno uno spettro contiguo rispetto all’origine, mentre in **banda traslata** (o passante) i segnali hanno uno spettro traslato su intervalli di frequenze non contigue all’origine.

Per la **modulazione in banda traslata** si usa un’onda elettromagnetica (sinusoide) detta PORTANTE (carrier) ad una determinata frequenza ( per traslare lo spettro del segnale intorno alla frequenza del portante



Il segnale può propagarsi nell’atmosfera o in un **mezzo trasmissivo guidante** (doppino) per mezzo di un supporto (**portante**) che è un’onda **elettromagnetica a opportuna frequenza** (opportuna lunghezza d’onda).

Il segnale e la sua banda (spettro) propagandosi nel mezzo trasmissivo possono subire distorsioni (attraverso dispersione, attenuazione o ritardo), legate alla funzione di trasferimento del mezzo.

Questo è infatti caratterizzato da una **velocità di trasmissione** (o capacità) **R** (in bps o bit/s), che dipende dalla banda del mezzo trasmissivo e dalla potenza ricevuta, e da un **ritardo di propagazione** del segnale .

Per evitare distorsioni il canale deve modificare la banda del segnale il meno possibile.  
**Banda passante per il mezzo > banda occupata dal segnale**.

Affinché gli impulsi siano distinguibili:

E la loro velocità massima:

Inoltre, il mezzo trasmissivo è caratterizzato da una funzione di trasferimento in frequenza **H(f)** che determina le caratteristiche del segnale in uscita a partire da quello in entrata.

In un canale con banda passante pari a **Be**:

Per aumentare la capacità di canale [bit/s] è possibile incrementare l’ordine della modulazione attraverso una **modulazione multilivello**.

Per fare ciò abbiamo bisogno di **N simboli**, di un flusso di bit diviso in gruppi **n = log2N**, e di una convenzione di assegnamento tra un simbolo e n bit.

Durante la modulazione multivello possiamo però avere degli errori in ricezione che possono esser causati da:

* **Rumore termico**.
* Interferenza da altre trasmissioni sullo stesso mezzo.
* Disturbi elettromagnetici.
* Perdite di sincronismo.

**2.1.4 La classificazione dei mezzi trasmissivi**

I mezzi trasmissivi si distinguono in: mezzi trasmissivi **guidati** e **non guidati**.

I mezzi trasmissivi **guidati** sono a loro volta divisi in mezzi elettrici, dove ad ogni bit è associato un particolare valore di tensione o corrente o determinate variazioni di tali grandezze, e in mezzi ottici (fibre ottiche) basati sulla propagazione guidata della luce.

I mezzi trasmissivi **non guidati** sono invece le onde radio, dove il segnale è associato ad un’onda elettromagnetica che si propaga nello spazio che ha la proprietà di riprodurre a distanza una corrente elettrica in un dispositivo ricevente (antenna).

**2.1.5 L’attenuazione**

Se un segnale di ingresso ha una potenza **PIN** e il segnale di uscita ha una potenza **POUT** si definisce **attenuazione** del collegamento A il rapporto:

**2.1.6 I codici correttori**

Per abbassare la probabilità d’errore in un blocco adottando dei **codici correttori d’errore** (Forward Error Correction - **FEC**).

Questi consistono nell’aggiungere dei bit di ridondanza cosicché gli errori che occorrono, se limitati in numero, possono essere corretti.

Se un codice non riesce a correggere un errore può a volte riuscire a rilevarlo (quando il controllo di parità da risultato negativo).

Nella trasmissione a commutazione di pacchetto si possono rilevare gli errori in ricezione e richiedere quindi la ritrasmissione del pacchetto errato (**ARQ** - Authomatic Repeat reQuest).

In ogni caso, nonostante codici e ritrasmissione esiste un limite alla velocità massima di un canale, questa viene detta **capacità di canale**:

**3 Il livello applicativo**

**3.1 I componenti del livello applicativo**

Mentre protocollo è l’insieme di regole, il servizio applicativo è “l’obbiettivo”.

L’**applicazione di rete** è un software che possa essere eseguito su diversi terminali e che possa comunicare tramite la rete.

(es. Firefox).

Creare una nuova applicazione non richiedere di cambiare il software della rete, poiché i nodi della rete non hanno software applicativo.

Inoltre, le applicazioni si trovano solo nei terminali, e possono quindi essere facilmente sviluppate e diffuse.

**Host**: dispositivo dell’utente.

**Processo**: programma software in esecuzione su un host.

**Comunicazione inter-processo** (**IPC**): tecnologie software il cui scopo è di consentire a diversi processi di comunicare scambiandosi informazioni e dati.

**3.2 Gli ingredienti per la comunicazione tra processi remoti**

Per la comunicazione tra processi serve innanzitutto un indirizzo associato a questi (un po’ come conoscere il numero di telefono dell’interlocutore).

Dopodiché bisogna scegliere la “lingua” della conversazione, attraverso il protocollo di scambio (tipi di messaggi scambiati, la loro sintassi, la semantica e le regole su come e quando inviare e ricevere).

Lo scambio di messaggi tra i processi applicativi avviene usando servizi di livelli inferiori attraverso i **SAP** (Service Access Point).

Cose serve quindi per esempio per identificare il browser Chrome in esecuzione sul mio host?

* L’indirizzo dell’host (indirizzo IP da 32 bit)
* L’indirizzo del SAP (numero di porta).

L’indirizzo di un processo in esecuzione (IP + porta) viene chiamato **socket**.

**3.2.1 I requisiti delle applicazioni**

* Affidabilità:  
  alcune applicazioni possono tollerare perdite parziali (es. audio), altre richiedono una completa affidabilità (es. telnet).
* Ritardo:  
  alcune applicazioni richiedono basso ritardo (games).
* Banda:  
  alcune applicazioni richiedono un minimo di velocitò di trasferimento, altre adattano la velocità disponibile

**3.2.2 Quale servizio di trasporto scegliere**

Servizio **TCP**:

* + - Connection-oriented:  
      instaurazione della connessione prima dello scambio di dati.
    - Trasporto affidabile:  
      senza perdita di dati.
    - Controllo di flusso:  
      il trasmettitore regola la velocità in base al ricevitore.
    - Controllo di congestione:  
      per impedire di sovraccaricare la rete.
    - Non fornisce però garanzie di ritardo e di banda.

Servizio **UDP**:

* Trasferimento non affidabile.
* Avviene senza connessione.
* Avviene senza controllo sul traffico.
* Non ha garanzie.
* Il trasferimento è “veloce”.

**3.3 Architetture applicative**

Si dividono in:

* **Client-server:** i dispositivi coinvolti nella comunicazione implementano solo il processo client o solo il processo server, inoltre i dispositivi client-server hanno caratteristiche diverse: i client possono solo eseguire richieste, i server possono solo rispondere a richieste ricevute.
* **Peer-to-peer** (**P2P**)**:** I dispositivi implementano tutti sia il processo client che quello server.
* **Ibrida**.

I **server** hanno**:**

* L’host sempre attivo.
* Un indirizzo IP permanente.
* La possibilità di utilizzo di macchine in *cluster.*
* Possono ricevere richieste da molti client.

I **client**:

* Comunicano con il server.
* Possono essere connessi in modo discontinuo.
* Possono cambiare indirizzo IP.
* Non comunicano con altri client.
* Possono inviare molte richieste allo stesso server.

**3.4 Servizio di comunicazione**

Il servizio di comunicazione è come una nuvola che collega un nodo A e un nodo B, contenenti due entità che vogliono comunicare tra loro.

È un servizio di trasferimento di unità informate.

Le chiamate del servizio si chiamano **primitive di servizio** e richiedono l’informazione.

La trasmissione dell’informazione può avvenire attraverso una **modalità a connessione**, dove viene instaurata quest’ultima, inviati i dati e chiuso il contatto, o attraverso una **modalità senza**, dove il tutto avviene con una sola fase.

**3.4.1 Il protocollo di comunicazione**

È l’insieme di regole con cui vengono scambiate le informazioni, divise in **PDU** (**Packet Data Units**).  
Le PDU sono formate da un header e da un pacchetto di dati e vengono scambiate solo per un colloquio tra identità dello stesso livello.

**3.4.2 La multiplazione**

Consente ad un livello di ottenere informazioni da più livelli superiori.

**3.4.3 Funzione di controllo d’errore**

Se non ci sono errori si invia un piccolo pacchetto (**ACK**), sennò scaduto un timer viene rimandata l’informazione.

Questa funzione è di solito implementata al livello Linea o Fisico

**3.4.4 La Funzione di instradamento (routing)**

Attraverso questa funzione l’interfaccia instradante, dopo aver ricevuto il messaggio da un livello superiore decide, a seguito della lettura dell’header, l’interfaccia d’uscita della PDU.

Esistono varie modalità di instradamento in Internet:

* Router
* LAN switch
* Proxy

La scelta dell’uscita avviene sulla base di informazioni memorizzate in una tabella di instradamento.  
Nei router IP le rotte configurate manualmente sono dette rotte statiche.

**3.5 Il servizio di web browsing**

Le pagine web sono un insieme di oggetti: file HTML, immagini JPEG, audio, e generalmente hanno un file HTML base che chiama gli altri oggetti collegati da un **URL** (**Uniform Resource Locator**).

In un URL esempio: http://www.polimi.it:80/index.html:

* **http** indica il protocollo.
* **www.polimit.it** indica l’indirizzo di rete del server.
* **:80** indica l’indirizzo di porta.
* **Index.html** indica la pagina web richiesta.

**3.5.1 La comunicazione HTTP**

Usa un’architettura client/server, dove ad ogni richiesta (**HTTP request**) corrisponde una risposta (**HTTP responses**).

Nella comunicazione HTTP viene usato un protocollo **stateless**, secondo il quale nessuna richiesta viene memorizzata.

HTTP si appoggia su TCP a livello di trasporto:

* Il client HTTP **inizia una connessione** TCP verso il server.
* Il server HTTP **accetta** connessioni TCP dal client HTTP.
* Client e server HTTP si **scambiano informazioni**.
* La connessione TCP tra client e server viene chiusa.

Le modalità di connessione tra client e server HTTP sono due: la **connessione non persistente**, e la **connessione persistente**.

Nella connessione non persistente le entità client/server aprono connessioni TCP diverse per ogni oggetto scambiato, ma possono però essere aperte più connessioni in contemporanea.

Nella connessione persistente invece, la connessione è singola, rimane aperta, e può essere usata per scambiare più oggetti contemporaneamente. Quest’ultima può inoltre avvenire **with pipelining** o **without pipelining**, ossia con o senza richieste HTTP inviate in serie.

**3.5.2 La stima del tempo di trasferimento in HTTP**

Il **RTT** (**Round Trip Time**) è il tempo impiegato per trasferire un messaggio “piccolo” con una connessione client-server.

In condizioni ideali il controllo del TCP è in grado di **limitare la congestione** di rete e **dividere in modo acquo la capacità** dei link tra i diversi flussi.

Queste condizioni ideali sono diversi RTT per flussi diversi e buffer nei nodi minori rispetto al prodotto banda/ritardo.

**3.5.3 Le richieste HTTP**

I messaggi HTTP sono codificati in ASCII (human readable).

Questi avvengono attraverso:

* Method URL version (**request line**).
* Header field name value (**header lines**).
* Entity body (**body**).

Alcuni esempi di **metodi HTTP** sono:

**GET:** è usato quando il client vuole scaricare un documento dal server.

**HEAD:** è usato quando il client vuole scaricare determinate informazioni.

**POST:** è usato per fornire degli input al server per un particolare oggetto.

Alcuni esempi di **Header HTTP** sono:

**Cache-control:** fornisce informazioni sulla cache.

**Authorization:** mostra i permessi del client.

**If-modified-since:** invia un documento se e solo se questo è modificato.

Le **risposte HTTP** sono invece divise in:

* Version status code phrase (**status line**).
* Header field name value (**header lines**).
* Entity body (**body**).

**3.6 I cookies**

Gli ingredienti dei cookies sono:

* Un **header cookie** nelle risposte HTTP (es. Matteo visita un sito per la prima volta).
* Un **header cookie** nella successiva richiesta HTTP (es. Quando il sito riceva la prima richiesta genera un **ID unico**, e una **entry** nel data base locale che corrisponde all’ID creato).
* Una **lista di cookie** mantenuta sull’host.
* Un **data base** di cookie mantenuto dal sito web.

Esempio:

Il client genera una richiesta usuale (**http request msg**), e il server genera un ID (**cookie**).

Questo ID viene associato al client e l’associazione viene salvata in un **database locale**.

Nella risposta il server include i campi usuali della risposta (**usual http response**) e un’header line  
**set-cookie** che permette di inserire nelle richieste successive l’identificativo del client, e ciò permette la customizzazione delle risposte da parte del server.

I **cookie** hanno una **validità temporale** che può variare.

**3.7 I proxy HTTP: cache di rete**

L’obbiettivo, nell’usare un proxy è quello di rispondere alle richieste HTTP senza coinvolgere il server HTTP, il cliente invita **tutte le richieste** HTTP ad un **proxy HTTP** e, se l’oggetto richiesto è disponibile nella cache del proxy questo risponde alla domanda, altrimenti recupera l’oggetto dal server d’origine e lo restituisce al client.

I proxy sono quindi degli **application gateway**, ossia instradatori di messaggi di livello applicativo e devono essere sia client (verso i server d’origine) che server (verso i client).

Inoltre, il server d’origine vede arrivare tutte le richieste dal proxy, per questo avviene un **mascheramento degli utenti**.

**3.8 HTTP/2**

L’obbiettivo della seconda versione di HTTP era quello di ridurre la latenza (o **loading time**) di pagine web e risolvere alcuni problemi di HTTP/1.1.

Esempio:

Il sito della gazzetta include 209 oggetti.

Attraverso una connessione HTTP/1.0 è consentita una connessione per oggetto (209 connessioni TCP richieste), mentre attraverso HTTP/1.1 avvengono connessioni **TCP persistenti ma “seriali”**, ossia se un oggetto è “lento” blocca anche gli altri (**Head of Line Problem**).

**3.8.1 Le caratteristiche di HTTP/2**

HTTP/2 è in formato binario, ossia trasferisce **frame**.

Consente la **multiplazione**: una connessione TCP per stream multipli, e la **comprensione degli header**.

Comprende un **servizio di server push** e uno di **controllo del flusso** (**flow control**) a livello applicativo.

Si appoggia inoltre su **TLS** (un protocollo che **cifra** gli scambi di informazione tra client e server).

**3.8.2 La comprensione degli header**

L’header delle richieste HTTP può avere dimensione non trascurabile in quanto può contenere molti cookie, header line per autenticazione, specifiche della transazione e altri dati.

Inoltre, l’header di richieste HTTP consecutive (verso lo stesso server) contiene informazioni ridondanti.

HTTP/2 tenta quindi di comprimere i messaggi scambiati attraverso:

* **La codifica di Huffman**, durante la quale si assegnano a stringhe binarie simboli più comuni.
* **L’indexing**, durante il quale si assegna un indice a header line più comuni, permettendo l’invio nei messaggi solo quest’ultimi.
* **La codifica differenziale**, dove l’header di richieste successive riporta per esteso solo la differenza con l’header delle richieste precedenti.

**Immagine che contiene diagramma

Descrizione generata automaticamente**

**3.8.3 La multiplazione**

Permette l’invio di più richieste e risposte in contemporanea.

Immagine che contiene diagramma

Descrizione generata automaticamente

**3.8.4 Il server push**

Attraverso il server push il server può inviare informazioni utili al client senza che il client ne faccia esplicita richiesta.

Immagine che contiene grafico

Descrizione generata automaticamente

**3.9 Il servizio E-mail**

Il servizio di posta elettronica è formato da:

* Un **client** d’utente (o **user agent**), ossia il processo applicativo che l’utente usa per accedere al servizio.
* Un **Mail server**, usato per trasferire messaggi di posta elettronica attraverso il protocollo **SMTP**.
* Il protocollo **SMTP**.
* I **protocolli di accesso** (quali **POP3** o **IMAP**), usati per “scaricare” e-mail dal proprio mail server.

Il server di posta, specifico per ogni utente colloqui con altri server di posta elettronica perché i messaggi devono essere smistati nel server del ricevente.

Il protocollo usato per spostare un messaggio di posta elettronica è **SMTP**, questo serve anche per trasferire mail dal client al main server di competenza.

**3.9.1 Il protocollo SMTP**

Essendo l’SMTP un protocollo applicativo è necessario definire sintassi e connessioni che usa.

Per le **connessioni** viene utilizzato il protocollo **TCP**, e fornisce sia da client che da server.

I mail server contengono una coda di **e-mail in ingresso** (**mailbox**) e una **coda di e-mail in uscita**.

Per il **formato** invece è necessario che le mail abbiano un **header** (contenente informazione e segnalazioni che specificano il destinatario, il mittente e l’oggetto), e un **body** (dove è presente il contenuto dell’e-mail (ASCII).

Per trasferire file non testuali esistono invece delle estensioni chiamate **MIME** (Multipurpose Internet Mail Extensions).

Per il download dei messaggi vengono invece usati altri protocolli:

* Il **POP3**, per scaricare messaggi, formato da una fase di **autorizzazione**, dove sono richiesti **user** e **password**, da una di risposte del server (**+OK** o **-ERR**) e da una fase di transazione (**list** elenca il numero di messaggi, **retr** recupera un messaggio, **dele** cancella il messaggio, **quit**). La connessione usata è di tipo persistente.
* L’**IMAP** per il download dei messaggi e la gestione della mailbox sul mail server. Consente inoltre di **leggere file system** e di **cancellare messaggi**.
* L’**HTTP**.

**3.10 Applicazioni P2P**

Nel file sharing P2P gli utenti utilizzano il software P2P sul proprio dispositivo, si collegano in modo intermittente a internet prendendo indirizzi IP diversi ogni volta e se un utente cerca un file l’applicazione trova altri utenti che lo hanno.

Sta quindi all’utente scegliere da dove scaricarlo. Il file entra quindi in download usando un protocollo HTTP e altri utenti potranno in seguito scaricarlo dallo stesso che ha scaricato il documento in origine.  
L’applicazione P2P è sia client che server.

**3.10.1 P2P a directory centralizzata**

Per la connessione P2P a directory centralizzata viene definito un **centro stella** (**directory**) che contiene gli indirizzi IP degli utenti che partecipano alla comunicazione.

Ogni dispositivo peer interroga il server centrale per uno specifico file, ma questo viene in seguito scaricato direttamente.

Seguono però i seguenti problemi:

* Se il server si rompe il sistema si blocca.
* Il server è un collo di bottiglia per il sistema.
* Chi gestisce il server può essere accusato di infrangere le regole sul copyright.

**3.10.2 P2P completamente distribuita**

In una connessione P2P completamente distribuita non è presente nessun server centrale il protocollo è di pubblico dominio.

Viene usata una **rete overlay**, dove peer si collegano ad un numero (< 10) di altri peer vicini, e la ricerca è distribuita. I vicini nella rete overlay possono essere fisicamente distanti.

Per la **ricerca di un file** i messaggi di richiesta vengono diffusi sulla rete di overlay, i peer inoltrano le richieste fino a una certa distanza e le risposte vengono inviate sul cammino opposto.

Per poter accedere alla **rete di overlay** bisogna conoscere almeno un indirizzo di un dispositivo già presente nella rete.

Il nuovo dispositivo può quindi mandare **messaggi di ping** a tutti i presenti nella rete e ne seguirà una **risposta pong** nel caso di possibile connessione.

**3.10.3 BitTorrent**

In questa particolare connessione P2P completamente condivisa i file sono divisi in **chunk**, che hanno tipicamente una dimensione di **256kB**.

Quest’architettura comprende sia gli host che vogliono partecipare alla rete di overlay che dei **tracker** che tengono traccia di tutti i dispositivi che partecipano ad un **torrent**.

Un **torrent** è un gruppo di peer che si scambiano chunk di uno stesso file.

I **chunk** scaricati seguono un meccanismo di **rarest first** secondo il quale, davanti a più richieste, viene scaricato il chunk mancante.

**3.11** **Il DNS (Domain Name System)**

Il DNS è un processo applicativo che si trova al livello 5.

Il suo utilizzo deriva dalla scarsa adattabilità degli indirizzi IP (32 bit) ad essere usati dagli applicativi, per questo è più comodo usare indirizzi simbolici.

Questo protocollo non è come l’HTTP o l’SMTP, serve infatti per risolvere nomi simbolici: dato un nome si trova l’indirizzo IP.

Sono però necessari:

* Un **database distribuito** costituito da molti **name servers**.
* Un **protocollo applicativo** basato su UDP tra name server e host per risolvere nomi simbolici (tradurre simboli in indirizzi IP).
* Un servizio di **host aliasing**, usato per trasformare indirizzi simbolici corti in più lunghi.
* Un **mail server aliasing**, in grado di accedere al mail server tramite il suo nome simbolico.
* Una **load distribution**.

I nomi simbolici sono organizzati secondo un’**architettura gerarchica**, dove in cima troviamo i **root main server**, contenenti informazioni per risolvere qualsiasi nome simbolico, troviamo poi i **TLD**, usati per risolvere i nomi simbolici appartenenti ad uno stesso dominio.

Esistono vari tipi di DNS:

* **A**, i quali contengono il mappaggio effettivo tra un indirizzo simbolico e IP.
* **NS**, che non contengono un mappaggio diretto e simbolico.
* **CNAME**, usate per definire gli alias.
* **MX**.

Esistono due modalità di risoluzione del nome simbolico: una iterativa e una ricorsiva.

Quella più utilizzata è la **modalità ricorsiva**, durante la quale:

* Avviene un collegamento al name server e avviene una richiesta di **NS**.
* Si contatta la rete del name server e viene **rilanciata la richiesta** inviata dal client.
* La richiesta arriva al **Root NS**, alla quale risponde il Local Name Server con l’indicazione “a chi chiedere”.
* Il Local Name Server contatta il **TLD** che corrisponde con un indirizzo del NS a cui chiedere.
* Si chiude il loop segnalando l’indirizzo **IP** che si voleva ottenere.

Nella modalità ricorsiva invece si crea un problema al root che si trova a gestire tutto.

Durante la comunicazione può comunque avvenire del **catching** (come per i proxy), ossia se il componente interrogato ha l’informazione richiesta in memoria risponde immediatamente senza passare per il percorso completo.

**3.11.1 I Content Distribution Networks (CDNs)**

Nascono dall’esigenza di distribuire in maniera efficiente molti contenuti contemporaneamente a più utenti lontani gli uni dagli altri (hanno la stessa funzione dei proxy).

Vengono usati per **contenuti multimediali** (video) e sono delle **reti di cache**, di server, ciascuno dei quali ospita una replica dei contenuti da condividere in una rete geograficamente distribuita.

Il miglior server è quello **più vicino**, quello con il **percorso più corto** e che **lascia decidere all’utente** (rilascia una lista di server possibili, tra i quali l’utente può scegliere).

**4 Il livello di trasporto**

È il livello tramite il quale interagiscono i processi applicativi (attraverso i socket) e disaccoppia il livello applicativo da ciò che c’è sotto mascherandone la complessità.

Il livello di trasporto ha il compito di i**nstaurare un collegamento logico tra le applicazioni residenti su host remoti** ed è presente **solo negli end system** e non negli altri dispositivi di rete. Ma più applicazioni possono essere attive su un solo host.

Questo livello svolge funzioni di **multiplexing/demultiplexing** per le applicazioni e ciascun collegamento logico tra esse deve passare per questo livello.

I servizi che offre sono:

* La **multiplazione** (sia UDP che TCP), associa ad ogni interfaccia SAP un socket.
* Il **buffering**, che fa riferimento alla capacità di gestire code in ingresso e in uscita del processo quando questo viene associato ad una porta.
* Il **trasporto**, che può essere affidabile (TCP), orientato alla connessione (TCP), non affidabile (UDP) o senza connessione (UDP).

Per poter assicurare il giusto trasferimento dell’informazione avviene il **checksum**, ossia un controllo dell’integrità, è un’informazione ridondante inserita nell’header per controllare l’errore.

Ciò avviene grazie a **16 bit** calcolati dal trasmettitore e inseriti nell’header, il ricevitore ripete lo stesso calcolo sul segmento ricevuto.

Il trasmettitore invia un insieme di bit diviso in blocchi da 16 e il campo checksum indirizzato a 0, tutti i blocchi vengono sommati in complemento a uno e si complementa il risultato.

Il ricevitore, quindi, sommerà tutti i blocchi in complemento a uno e complementerà ulteriormente il risultato, avendo infine tutti 0 (che verranno scartati) eccetto il pacchetto.

Una volta ricevuto il pacchetto può essere inviato un messaggio di **ACK** o **NACK**, a seconda della correttezza o meno di questo.

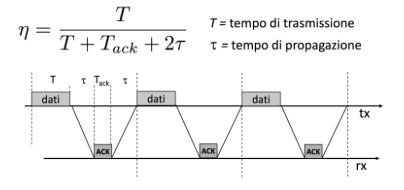
**4.1 Il protocollo Stop and Wait**

In questo protocollo si utilizzano gli ACK e il timeout: per ogni pacchetto il ricevitore si aspetta il riscontro in un certo tempo.

Il trasmettitore trasmette infatti un pacchetto e inizializza un contatore accoppiato con un meccanismo di stima del RTT.

Nel pacchetto viene inserito un sistema di numerazione chiamato **SN** che il ricevitore utilizza per riconoscere i pacchetti che sono stati trasmessi più di una volta.

Quando il trasmettitore riceve un ACK deve poterlo assegnare al pacchetto corretto, per questo anche l’ACK deve avere lo stesso SN del pacchetto corrispondente.

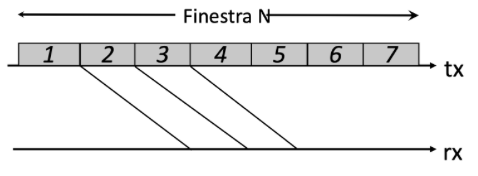


**4.2 Il protocollo Go-back-N**

È una variante dello Stop and Wait nella quale si possono trasmettere fino a N pacchetti senza aver avuto il riscontro.

Se il riscontro del primo pacchetto avviene prima della fine della finestra, la finestra viene fatta scorrere di una posizione (**sliding window**).

Inoltre, se non ci sono errori la trasmissione non si ferma mai.



Altrimenti, si ricomincia a trasmettere la finestra dal primo pacchetto non riscontrato.

Raggiunto l’ultimo pacchetto la trasmissione si blocca in attesa di un nuovo ACK (la ritrasmissione del primo pacchetto avviene a seguito dello scadere di un timeout).

Escludendo il fuori sequenza il riscontro **ACK può essere collettivo**, per esempio se il ricevitore riceve un ACK3 vuol dire che è avvenuta la corretta trasmissione di 3 pacchetti (dal primo al terzo).

La **finestra ottimale** coincide con il RTT, questa aumenta di un pacchetto alla volta e la trasmissione non si interrompe mai.

Inoltre, la finestra può anche essere dimensionata in tempo, ma il dimensionamento si complica se non si conoscono i tempi di propagazione.

Costruire una finestra grande può essere un’arma a doppio taglio in quanto può passare più tempo dalla scoperta di un errore e può avvenire un maggior numero di trasmissioni inutili.

Gli ACK possono anche essere inseriti negli header dei pacchetti che viaggiano in direzione opposta (piggy-backing).

**4.2.1 Le regole del protocollo Go-back-n**

Consideriamo **N** la dimensione della finestra, **Nlast** l’ultimo riscontro ricevuto e **NC** il numero corrente disponibile per pacchetto in trasmissione.

* Ogni nuovo pacchetto viene messo in attesa se **NC Nlast + N**.
* Ad ogni riscontro AN ricevuto si pone **Nlast = AN**.
* I pacchetti della finestra vengono trasmessi senza vincoli di temporizzazione.
* In caso di scadenza di timeout la ritrasmissione deve ripartire dal pacchetto **Nlast**.

**4.3 Il Controllo di flusso**

Il buffer di ricezione è limitato a **W** posizioni, per evitare di sovraccaricare il destinatore con una quantità di dati che non riesca a immettere nel buffer.

Ciò si implementa con il **controllo di** **flusso a finestra mobile**, è infatti possibile usare un meccanismo come quello del Go-Back-N dove la sorgente non può inviare più di **W trame** senza aver **ricevuto il riscontro** (inviato dal ricevitore solo quando i pacchetti vengono letti e **tolti dal buffer**).

Questo metodo funziona ma genera problemi nel caso in cui l’applicazione sia lenta a leggere il buffer.

**4.4 Il protocollo TCP**

In questo protocollo gli indirizzi di porta (sorgente e destinazione) sono formati da 16bit e vengono chiamate **Source Port** e **Destination Port**.

Il **Sequence Number** è invece il numero di frequenza del primo byte nel payload.

**5 Il livello di rete**

Mentre lo strato di trasporto mette in comunicazione due **processi** lo strato di rete **trasferisce i dati tra gli host** che ospitano i due processi comunicanti.

Il livello di rete si divide in un **piano dati** (**data plane**), dove vi troviamo i protocolli usati per trasferire i dati d’utente (IP), e in un **piano di controllo** (**control plane**), dove invece sono memorizzati i protocolli di segnalazione per supportare il trasferimento dati (ICMP, ARP, RARP, OSPF, RIP).

**5.1 Le funzioni fondamentali dello strato di rete**

Queste sono:

* **Indirizzamento**, ossia l’identificazione univoca dell’interfaccia di rete di un host/server.
* **Inoltro/forwarding**, ossia una funzione locale con cui il router trasferisce i pacchetti dall’ingresso all’uscita.
* **Instradamento/routing**, ossia il processo che determina i percorsi dei pacchetti dalla sorgente alla destinazione (il processo globale viene svolto da **algoritmi di routing**). Più pacchetti destinati allo stesso host possono essere trattati in maniera diversa.
* **Connectionless**, ossia la comunicazione è progettata secondo un paradigma packet-oriented.
* **Frammentazione/Deframmentazione**

**5.2 Il data plane a livello di rete**

I segmenti dello strato di trasporto vengono trasferiti dallo strato di rete dall’host sorgente all’host di destinazione.

Nel **lato sorgente** vengono incapsulati in **datagram**, questi vengono inoltrati **hop-by-hop** fino a destinazione e i router **esaminano i campi dell’header** di ciascun datagram IP che li attraversa.

**Lato destinazione**, invece, i segmenti vengono consegnati allo strato di trasporto.

**\*I protocolli del livello di rete sono implementati in ogni host e in ogni router.**

Il **trasferimento** avviene senza connessione, ogni router che riceve un datagramma legge l’header e decide   
come/dove inoltrare il datagramma sulla base di un **indirizzo di destinazione** (presente nell’header) e di una **tabella di instradamento** presente in ogni nodo.

I pacchetti possono percorrere strade diverse tra sorgente e destinazione.

**5.3 L’indirizzo IP (IPv4)**

È un numero binario di **32 bit**, e “a beneficio degli esseri umani” viene scritto nella forma x.y.z.w, dove ciascuno di questi rappresenta 8 bit.

Si usa scriverlo nella forma decimale.

Questo è associato in modo univoco ad un’interfaccia di rete di un host o di un router.

(non vi è associato direttamente perché questi potrebbero avere più interfacce di rete).

L’indirizzo IP deve avere una valenza e un’univocità universali, per questo ogni gestore di rete ha a disposizione un blocco di indirizzi che distribuisce alle interfacce dei singoli apparati.

Gli indirizzi del blocco devono avere i primi n bit identici e questi vengono chiamati **prefisso** (o **identificativo**) **di rete** (network prefix o **NetID**)**.**

L’indirizzo IP è quindi diviso nei primi n bit di NetID che identificano la rete e nei rimanenti bit chiamati **HostID** che sono invece usati per identificare un host specifico nell’ambito della rete.

Ad oggi gli indirizzi IPv4 **sono finiti** e si è passati agli indirizzi IPv6.

**5.4 Le netmask**

La netmask è un numero binario di 32 bit associato ad una rete IP, inizia con n bit a 1, con n pari alla lunghezza del NetID, i restanti 32 – n bit sono a 0.

Questa indica quali bit di un indirizzo IP sono assegnati al NetID e viene indicata anch’essa con una dotted decimal notation.

**5.5 Indirizzi speciali**

Tra questi troviamo **l’indirizzo broadcast diretto:** un indirizzo con il campo HostID di soli 1 che viene trattato dai router di transito come un normale pacchetto (inoltrano al next hop)**.**

È un altro indirizzo speciale **l’indirizzo di broadcast limitato:** un indirizzo di soli 1 che assume il significato di indirizzo broadcast nella stessa rete di chi invia il pacchetto (il pacchetto non può oltrepassare il router.  
  
Quando il campo NetID è posto a zero l’indirizzo indica l’host il cui indirizzo è contenuto nel campo host sulla stessa rete del mittente.

Se anche il campo host è posto a zero l’indirizzo indica il mittente stesso del pacchetto.

Infine, l’indirizzo con il primo ottetto pari a 127 e gli altri campi qualsivoglia indica il loopback sullo stesso host.

**5.6 L’indirizzamento classful**

Gli indirizzi IPv4 erano divisi in 5 classi:

* Classe A: poche reti grandi.
* Classe B: reti medio-grandi.
* Classe C: tante reti piccole.
* Classe D: indirizzi multicast.
* Classe E: riservato a uso futuro.

Nel sistema classful i router deducono la lunghezza del prefisso dai primi bit del primo byte.

**Non occorre** la netmask per ciascuna entry della tabella di routing.

I problemi delle classi sono che le reti di classe A e B sono troppo grandi, quest’ultime sono inoltre troppo poche e quelle di classe C sono invece troppo piccole.

Il problema viene quindi risolto eliminando completamente la distinzione tra classe A, B e C.

**5.7 Il problema della grandezza delle reti IP**

Le reti IP possono essere troppo grandi e ciò scaturisce problemi di performance e di management.

La soluzione viene quindi trovata nell’introdurre una gerarchia nelle reti:

* Suddividere la rete in molte sottoreti (subnet)
* Ciascuna subnet corrisponde ad una rete fisica.

**5.8 L’inoltro dei pacchetti**

IP è una tecnica di internetworing: nel trasferimento di pacchetti tra due host si serve della capacità di inoltro delle reti attraversate.

L’inoltro può essere **diretto** quando la destinazione è nella stessa rete IP, o **indiretto** quando invece non lo è.  
La loro trasmissione nelle reti locali di basa sui pacchetti incapsulati nelle trame di livello 2 e sugli indirizzi di livello 2 (indirizzi MAC) dei dispositivi.

I **router** sono dispositivi di internetworking con interfacce di uscita multiple e anche essi seguono le tecniche di inoltro diretto ed indiretto ma nel primo hanno più di un’interfaccia dove poter effettuare l’invio, e per il secondo si basano su tabelle di routing dove è definita la rotta di instradamento.

In questo caso l’inoltro è **destination based**, ossia basato sul solo indirizzo di destinazione e vi troviamo il **next hop routing** secondo il quale nelle tabelle di routing, per ogni rete di destinazione, è indicato solo il router successivo.

L’inoltro avviene da router a router attraverso le reti IP e questi inoltrano i pacchetti bassandosi esclusivamente sulla parte di NetID dell’indirizzo di destinazione.

Tutti gli host che appartengono alla rete di destinazione sono identificati nelle tabelle di routing da una singola entry (**address aggregation**).

Nelle tabelle di routing deve **sempre essere indicata** **esplicitamente**, per ogni entry, la lunghezza del NetID.  
Inoltre, i protocolli di routing devono supportare l’invio dell’informazione di Netmask insieme agli indirizzi di rete in tutti i messaggi di annuncio di rotte.

Per poter inoltrare un pacchetto bisogna sapere se appartiene alla sottorete di una delle interfacce e per effettuare la verifica si fa un **AND bit a bit** tra indirizzo dell’interfaccia e netmask e tra indirizzo di destinazione e netmask.  
Se i due risultati coincidono allora la sottorete è la stessa e si procede con l’inoltro diretto.

Se i due risultati non dovessero coincidere si procederebbe allora con un inoltro indiretto consultando la tabella di routing; nel caso di un esito positivo per più righe della tabella viene selezionata quella con la netmask che ha il maggior numero di 1 (**longest prefix matching**).

Il **formato della tabella di routing** è il seguente:

Immagine che contiene tavolo

Descrizione generata automaticamente

Dove nella sezione flag possiamo ritrovare:

* U: la route è attiva.
* G: il destinatario è fuori dalla sottorete (default gateway).
* H: il destinatario è un host specifico (non una rete).
* D: rotta dinamica creata dal protocollo di routing o reindirizzamento all’ICMP.
* M: rotta modificata dal protocollo di routing o reindirizzamento all’ICMP.

Il **reference count** non è che il numero di connessioni attive per quella rotta, l’**use** il numero di pacchetti verso il destinatario e l’**interface** il nome dell’interfaccia d’uscita.

Inoltre, il datagram IP viene eliminato se non consegnato entro un preassegnato **time-to-leave**.

**5.8 Il formato del datagramma IP**

Questo ha una grandezza che varia dai 20 ai 65536 byte di cui 20-24 sono di header.

L’header è diviso in:

* **Ver** (4 bit): indica la versione del protocollo.
* **HLEN** (4bit, **Header Lenght**): indica la lunghezza dell’header del pacchetto.
* **Total Lenght** (16 bit).
* **TOS type of service** (8 bit).
* **Protocol**: indica il protocollo di livello superior.
* **TTL** (**Time To Live**): viene settato ad un valore elevato da chi genere il pacchetto e viene decrementato da ogni router attraversato.
* **Identification** (16 bit) e **Frag. Offset** (13 bit): sono i campi usati per la frammentazione.
* **Flags** (2 bit): settato a **M** (more) pari a 0 solo nell’ultimo frammento e **D** (Do not fragment) posto a 1 quando non si vuole che lungo il percorso venga applicata la frammentazione.

**5.9 I protocolli accessori**

Il protocollo IP richiede alcune funzioni di gestione accessorie che vengono svolte da appositi protocolli:

* **ICMP** (Internet Control Message Protocol): trasferisce i messaggi di segnalazione tra router e host.
* **ARP** (Address Resolution Protocol): viene usato per scoprire gli indirizzi di livello due associati agli indirizzi IP in una rete.
* **RARP** (Reverse Address Resolution Protocol): usato per scoprire l’indirizzo IP da parte di un host che si connette ad una rete.
* **DHCP** (Dynamic Host Configuration Protocol): viene usato per assegnare dinamicamente gli indirizzi agli host di una rete.

**5.9.1 ICMP**

Nel pacchetto IP il campo Protocol indica il codice dell’ICMP.

Vi possono essere diversi tipi di errori, come il destination unreachable, il source quenchee il parameter problem.

**Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente**

L’ICMP non corregge gli errori ma si limita a segnalarli e l’evento errore è notificato alla sorgente del pacchetto IP che lo ha causato.

I messaggi di errore contengono **l’header del pacchetto IP** che li ha generati e i suoi **primi 8 byte di dati**.

Messaggi come quelli di **Echo-request** e **Echo-reply** sono usati per verificare la raggiungibilità e lo stato di un host io un router.

In questi casi il campo identifier viene scelto dal mittente della richiesta e nella risposta viene ripetuto lo stesso identifier di quest’ultima.

Il **traceroute** usa messaggi di Echo-request verso la destinazione e il primo di questi ha TTL = 1.

Il secondo messaggio avrà quindi TTL = 2 e così via fino ad arrivare al destinatario, il quale risponderà con un’Echo reply e così il mittente saprà di aver esplorato tutta la via.

Inoltre, un host può mandare un messaggio di **address mask request** al/ai router di una rete per conoscerne la netmask.

**5.9.2 ARP e RARP**

Tabelle di corrispondenza tra indirizzi IP e indirizzi di livello inferiore (fisici) vengono create dinamicamente da ciascun host mediante il protocollo **ARP**.

L’**Address Resolution Protocol** si basa sulla capacità di indirizzamento broadcast della rete locale.  
 E se l’**ARP** è in grado di associare un indirizzo fisico ad un indirizzo IP noto il protocollo **RARP** è invece capace di fare l’inverso.

**5.9.3 DHCP**

Le procedure statiche di assegnamento degli indirizzi sono poco flessibili e per questo può essere comodo non configurare i singoli host con l’indirizzo IP ma usare un server per memorizzare tutte le configurazioni.

Questo perché in molti casi non è necessario avere un’associazione stabile tra i due indirizzi ma si può usare un’associazione dinamica.

Un client che deve configurare il proprio stack IP invia in broadcast un messaggio di **DHCPDISCOVER** contenente il proprio indirizzo fisico e il server risponde con un messaggio di **DHCPOFFER** contenente un proprio identificativo e un indirizzo IP proposto.

Il client può quindi accettare la richiesta attraverso un **DHCPREQUEST** a seguito della quale il server crea l’associazione con l’indirizzo IP e manda un messaggio di **DHCPACK** contenente tutte le informazioni di configurazione necessarie.

Il rilascio dell’indirizzo avviene infine con l’invio di un messaggio di **DHCPRELEASE** da parte del client.

**5.10 Connessione intranet/internet**

Le connessioni intranet possono usare un indirizzamento pubblico (usufruendo di proxy applicativi e di un router semplice) o un indirizzamento privato (attraverso NAT e proxy).

**5.10.1 Connessione con router semplice**

L’intranet usa indirizzi IP pubblici e di fatto **scompare**, dando però scarsa sicurezza.

**5.10.2 Connessione tramite proxy applicativo**

Funziona sia con indirizzamento pubblico che privato e in questo caso Intranet e INTERNET sono scollegate a livello IP: qualunque richiesta viene inviata al proxy che la inoltra con il proprio IP address pubblico (occorre avere un proxy per tutte le applicazioni).

**5.10.3** **Connessione tramite NAT (Network Address Translator)**

I NAT hanno tutte le funzionalità dei router classici e in più sanno gestire anche il mapping di uno spazio di indirizzamento privato in un altro spazio pubblico.

Ciò è reso disponibile su un router/gateway e consente di associare (anche contemporaneamente) un ridotto numero di indirizzi pubblici ai numeri della numerazione privata.

Perché lo scambio di informazioni sia bidirezionale però occorre mantenere l’associazione tra indirizzo privato e pubblico attraverso una tabella di NAT.

**5.11 Routing Unicast**

L’instradamento è alla base della funzionalità di rete implementata dalle entità di livello 3 (OSI) dei nodi.

Consente a due nodi A e B, non collegati direttamente, di comunicare tra loro mediante la collaborazione di altri nodi.

**L’algoritmo di routing** definisce i criteri di scelta del cammino nella rete per i pacchetti e costruisce le **tabelle di routing** che vengono usate dai nodi per effettuare il **forwarding**.

Il tipo di rete (datagram, circuito virtuale) determina il tipo di tabelle da utilizzare e i gradi di libertà della politica di routing.

Quando parliamo di **protocolli di routing** vogliamo indicare due diverse funzionalità:

* Lo scambio fra i router di informazioni di raggiungibilità.
* La costruzione delle tabelle di routing.

Il tipo di inoltro (**forwarding**) utilizzato dalle reti IP condiziona la scelta delle politiche di routing.

Questo è **destination-based** e con **inoltro al nodo successivo** (next-hop routing).

Non è quindi possibile instradare in modo indipendente ogni relazione di traffico (coppia sorgente-destinazione).

Nella tabella viene indicato soltanto il next-hop con cammino minimo.

Per il calcolo di quest’ultimo ad ogni **arco** tra due router è associato un **peso** scelto in base ad una **metrica di costo**.

Il percorso di cammini minimo forma l’**MST** (Minimum Spanning Tree).

Per poter trovare l’MST possiamo ricorrere a due algoritmi diversi:

**5.11.1** **Bellman-ford**

**Bellman-ford** secondo il quale si hanno pesi sia positivi che negativi e non esiste alcun ciclo di lunghezza negativa.

Lo scopo è quello di trovare i cammini minimi tra un nodo e tutti gli altri oppure di trovare i cammini minimi da tutti i nodi ad un nodo.

L’informazione sulla connettività scambiata dai nodi è costituita dal Distance Vector (DV).

Il DV è inviato ai soli nodi adiacenti, periodicamente o a seguito di un cambiamento nella topologia di rete.

Secondo l’algoritmo di **Bellman-ford distribuito** viene incrementata la distanza dalle destinazioni specificate del costo del link in ingresso, e questo viene ripetuto per ogni destinazione specificata nel DV.

Il vantaggio del DV è quello di essere molto **facile da implementare**, vi sono però **problemi di convergenza**, il fatto che questo sia **limitato dal nodo più lento** e che dopo un cambiamento **possono sussistere dei cicli** (loop) per un tempo anche lungo.

Un altro problema che potrebbe sorgere è quello del **counting to infinity**, il quale avviene nel caso in cui uno o più nodi non siano più raggiungibili dalla rete.

Rimedi a questo possono essere quello di assumere l’infinito come valore finito, attraverso l’**hop count limit**.

Si potrebbe anche usufruire dello **split-horizon**, secondo il quale un nodo non va a dichiarare al suo next-stop il peso necessario per arrivare a destinazione.

Una versione dello split-horizon è il **poisonus reverse**, nel quale il nodo include nel messaggio tutte le destinazioni ma pone a distanza infinita quelle raggiungibili tramite i link mancanti.

Un altro rimedio potrebbe consistere nell’utilizzo dei contatori (**hold down**), secondo il quale dopo un tempo Tflush la rotta viene cancellata.

**5.11.2** **Dijkstra**

**Dijkstra**, si applica al generico nodo sorgente, questo è in generale più conveniente.

In questo caso infatti ogni nodo impara a conoscere i nodi, le sue destinazioni adiacenti e le relative distanze per raggiungerle.

Ogni nodo invia a tutti gli altri nodi della rate (**flooding**) queste informazioni mediante dei **Link State Packet** (**LSP**).

Tutti i nodi si costruiscono un database di LSP e una mappa completa della topologia della rete e sulla base di questa informazione vengono calcolati i cammini minimi verso tutte le destinazioni.

Questo algoritmo risulta più flessibile in quanto ogni **nodo ha una mappa completa della rete** e non è necessario inviare l’informazione LSP periodicamente ma solo dopo un cambiamento.

Tutti i **nodi inoltre vengono subito informati dei cambiamenti**.

È necessario però un **protocollo dedicato** a mantenere l’informazione sui vicini (Hello), l’utilizzo del **flooding** e un riscontro dei pacchetti di routing inviati.

Durante il **flooding** ogni pacchetto in arrivo viene ritrasmesso su tutte le uscite eccetto quella da cui è stato ricevuto (per questo occorre prevenire i loop).

Ovviamente non possiamo pensare di applicare algoritmi del genere in tutto l’internet.

Quest’ultimo è infatti suddiviso in **Autonomous Systems** (**AS**), ossia porzioni di rete gestite dalle stesse autorità.

Un router esterno a un **AS** si chiama **Interior Gateway**, mentre uno a bordo dell’**AS** viene nominato **Exterior Gateway**.

Immagine che contiene cerchio, diagramma, linea, design

Descrizione generata automaticamente

All’interno di un **AS** il routing è indipendente dagli altri **AS**: gli interior gateway si scambiano informazioni di routing usando un **Interior Gateway Protocol** (**IGP**).

Mentre gli exterior gateway si scambiano informazioni usando l’**Exterior Gateway Protocol** (**EGP**).

**6 Il livello di linea**

I router possono scambiare pacchetti IP attraversando collegamenti e reti locali di tipo eterogeneo, ma come viaggiano le informazioni tra uno e l’altro?

Che differenza c’è tra i collegamenti diretti e quelli condivisi?

Il **livello di linea** è il primo livello logico presente nella modalità a pacchetto e serve a identificare logicamente i bit o gruppi di bit scambiati col livello fisico e a segnalare o correggere gli errori (può inoltre comprendere multiplazione e accesso multiplo).

Questo livello è implementato nella **scheda di rete** (**NIC – Network Interface Card**), solitamente su un chipset dedicato.

Esistono fondamentalmente tre tipologie di livelli di linea:

* Collegamenti punto-punto (**P2P**).
* Collegamenti **broadcast**.
* Collegamenti **commutati** (variante del P2P con altri elementi di rete).

**6.1 I collegamenti P2P**

La prima funzione del livello logico è quella di individuare il significato dei bit scambiati con il livello fisico, per questo vengono raggruppati in una struttura dati definita dal protocollo e chiamata “**trama**” (**PDU-2**).  
Occorre quindi un mezzo per identificare la posizione delle varie trame all’interno del flusso e per questo vengono usati **delimitatori di trama** (particolari stringhe di bit) o **segnalazioni** passate dal livello fisico.

**6.1.1 Esempio HDLC**

Un esempio di flag ritrovabile all’inizio e alla fine di una trama è:

**Immagine che contiene Carattere, testo, numero, logo

Descrizione generata automaticamente**

Un possibile problema potrebbe essere quello di ritrovare la stessa flag nella sequenza di dati, ma si può facilmente ovviare a ciò semplicemente attraverso il **bit stuffing**: si inserisce uno 0 dopo aver osservato cinque 1 consecutivi.

**6.2 Il controllo d’errore**

Già trattato nel livello di trasporto a differenza di questo, il quale scopo era quello di recuperare i segmenti persi, nel livello di linea l’obiettivo è il recupero degli errori di livello fisico

**6.3 La multiplazione**

Nei collegamenti punto-punto i protocolli di linea possono essere istanziati su più canali fisici.

Quest’operazione viene definita **multiplazione fisica**.

**6.3.1 La** **multiplazione fisica**

La multiplazione a questo livello consiste nel suddividere la capacità di un canale a velocità costante in sottocanali.

Si distingue per la caratteristica fisica attraverso la quale i di versi segnali vengono separati in:

* + Divisione di spazio.
  + Divisione di frequenza (**FDM**).
  + Divisione di tempo (**TDM**).
  + Divisione di codice (**CDM**).
  + Divisione di lunghezza d’onda (**WDM**).

**6.3.2 FDM (Frequency Division Multiplexing)**

Il mezzo trasmissivo è caratterizzato da una banda di frequenza utilizzabili.

La banda complessiva può essere divisa in sotto-bande cui associare un canale.

**6.3.3** **TDM**

I bit di N flussi vengono raccolti in code e trasmessi sul flusso di uscita a gruppi di K

**Immagine che contiene diagramma, schermata, linea, Piano

Descrizione generata automaticamente**

La durata della trama deve eguagliare l’intervallo di tempo in cui sul singolo canale in entrata arrivano i bit in numero pari a quelli trasmessi nella trama.

**6.4 Collegamenti broadcast**

Negli anni 70, quando la funzione di rete non implementava ancora capacità computazionali e i nodi dell’ARPANET gestivano collegamenti dell’ordine di poche decine di kbps, esistevano già reti locali con velocità dell’ordine dei Mbps.

Per questo si usava il **canale broadcast** senza funzione di rete (commutazione/switching), attraverso il quale tutti ricevevano le trame ma solo i destinatari la prelevavano e la inoltravano ai livelli superiori.

In questo canale però trasmissioni contemporanee (o quasi) possono provocare “collisioni”.

**L’accesso multiplo** è la funzione che consente di regolare l’accesso al canale ed evitare le collisioni, questa può essere implementata sia a livello fisico dividendo staticamente le risorse tra le stazioni, sia a livello del protocollo di linea gestendo l’accesso pacchetto per pacchetto.

**6.4.1 Il duplexing**

È la modalità con la quale si ricavano i due sensi di trasmissione da un unico mezzo trasmissivo.

Può essere visto come un caso particolare di accesso multiplo.

In alcuni casi particolari (**FULL DUPLEX**) si può inoltre trasmettere e ricevere contemporaneamente.

**6.4.2 Accesso multiplo casuale**

Al livello di pacchetto l’accesso multiplo è gestito con dei meccanismi che regolano l’istante di trasmissione dei singoli pacchetti.

Il coordinamento può essere gestito da un’entità centrale ma molto più spesso è gestito in modo distribuito dalle singole stazioni.

In questi casi il livello di linea è diviso in due sottolivelli:

* **MAC** (**Medium Access Control**).
* **LLC** (**Logical Link Control**).

Un esempio di trasmissione casuale è quello che avveniva con il protocollo **ALOHA** dove i pacchetti venivano inviati in ordine e dopo un ritardo casuali.

Questo è però un protocollo “maleducato” che non verifica che qualcuno stia usando il canale prima di trasmettere, e per questo è poco efficiente con più canali.

Una collisione avviene dal momento in cui un altro pacchetto viene mandato prima della chiusura di una finestra equivalente a **2**.

**Immagine che contiene linea, diagramma, schermata, Diagramma

Descrizione generata automaticamente**

L’efficienza del meccanismo dipende dal rapporto:

Se **a > 1** il primo bit arriva quando la trasmissione è già finita e ascoltare il canale non serve a niente.

Inoltre, anche le stazioni possono ascoltare durante la comunicazione così da effettuare una **collisione**, attraverso il **collision detect**.

Ciò elimina la necessità di un riscontro ACK dal ricevitore.

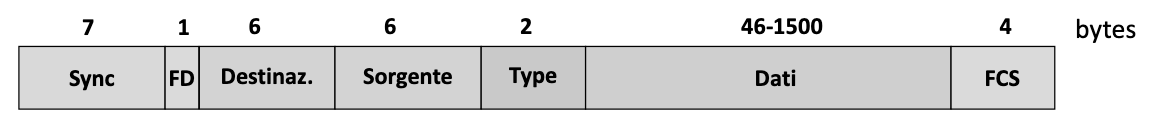
**6.5 Collegamenti locali**

Collegamenti locali avvengono attraverso cavi come l’ethernet.

I protocolli usati per collegamenti di rete locale (IEEE) sono il **Logical Link Control** (**LLC**) per il Multiplexing, il **MAC** per il filtraggio e l’accesso multiplo e il **PHY**.

Per rete locale (ethernet) viene invece usato quest’ultimo per multiplexing, filtraggio e accesso multiplo, e il **PHY**.

**6.5.1 Le trame dell’ethernet**



L’ethernet si divide in:

* **Sync**: ossia il preambolo di sincronizzazione di livello fisico.
* **FD**: il delimitatore di inizio trama.
* **Indirizzi**: sono indirizzi di 48 bit definiti dal costruttore nella scheda di rete (NIC) che indicano destinazione e sorgente.
* **Type**: serve per la multiplazione di più livelli superiori.
* **Dati**: Campo dati per PDU proveniente dal livello superiore.
* **FCS**: Frame Check Sequence per il controllo d’errore.

Gli **indirizzi** di rete locale sono detti **indirizzi MAC** o **indirizzi fisici**, servono per la funzione di filtraggio:

* I **primi 3 byte** identificano il costruttore.
* I **secondi** **3** identificano la scheda.

I 48 bit sono di solito indicati con notazione esadecimale e l’indirizzo con 48 bit a “1” è l’indirizzo broadcast (tutte le stazioni ricevono e processano la trama).

**6.6 Collegamenti commutati**

Finora abbiamo visto con il livello di linea broadcast soltanto le LAN, ma con l’ethernet sorge una possibilità in più: le **LAN commutate**, implementabili attraverso **hub** o **switch**.

Mentre nell’**hub** avviene una ripetizione bit a bit nello **switch** avviene lo store & forward.

**6.6.1 Lo switch**

Le funzioni dello switch sono:

* **Filtering**: se una trama ricevuta da LAN 1 è indirizzata ad una stazione di LAN 2 viene scartata.
* **Relay**: se una trama ricevuta da LAN 1 è indirizzata ad una stazione di LAN 2 viene trasmessa su LAN 2.

Per stabilire se filtrare/instradare una trama si consulta una tabella di instradamento locale chiamata **forwarding data base** (**FDB**).

A ciascuna porta di uno switch è collegato un **dominio** che può contenere una rete broadcast tradizionale o anche solo una stazione (LAN completamente commutata).

Tramite l’**inoltro** lo switch assicura che le trame provenienti da ciascun dominio siano inoltrate al dominio di destinazione.

Tramite la **funzione broadcast** invece le trame con destinazione indirizzo broadcast sono inoltrate su tutti i domini tranne quello di origine.

Lo switch segmenta domini di accesso multiplo, anche detti **domini di collisione**, mantenendo la rete un unico dominio broadcast.

La segmentazione consente di aumentare l’efficienza.

Per quanto riguarda la tabella di switching in essa lo switch è completamente trasparente e questa viene compilata automaticamente.

Si chiama **transparent bridging** ed avviene in più fasi:

* **Inizializzazione**: la tabella di switch viene creata ed è vuota.
* **All’arrivo di ogni pacchetto**: l’indirizzo sorgente viene inserito in tabella.
* **Inoltro**: se l’indirizzo si trova in tabella avviene l’inoltro alla porta corrispondente, altrimenti avviene un inoltro broadcast.

Per far si che ciò funzioni è però necessaria una rete ad albero; in caso contrario viene usato un algoritmo di **spanning tree** che permette di ricavare a partire da una topologia fisica magliata una logica ad albero.

Le **Virtual LANs** (**VLAN**) consentono di creare LAN logicamente separate su un’unica LAN fisica (commutata).

Pur essendo sulla stessa rete fisica però le stazioni differenti non possono comunicare direttamente a livello 2 ma la comunicazione deve avvenire a livello 3 attraverso un router.

Sui link che connettono switch e router le trame di differenti VLAN devono essere differenziate da etichette (tag).

A questo scopo è usato il protocollo **802.1q** (**LAN tagging**).

Dal punto di vista dei dispositivi di livello 3 (router) le VLAN equivalgono a LAN fisiche separate.

**6.7 La WiFi**

La tecnologia WiFi è standardizzata dal gruppo di lavoro IEEE 802.11 e rappresenta la versione Wireless di Ethernet usata per reti locali (WLAN).

Esistono molte versioni di livello fisico che operano a velocità e bande di frequenze diverse.

**6.7.1 La struttura di rete**

Una rete è formata da **Basic Service Set** (**BSS**) con stazioni mobili in mutua comunicazione a 2 modalità: **Access Point** (**AP**) o **rete ad-hoc**, e da **Extended Service Set** (**ESS**) composti da diversi BSS messi in comunicazione tramite un **Distribution System** (**DS**).

Per far si che la rete funzioni abbiamo bisogno di un nuovo protocollo: il **CSMA/CA** (**Collision Avoidance**), dove il mittente invia esplicita richiesta di autorizzazione a trasmettere (**RTS**), il destinatario risponde con esplicita autorizzazione (**CTS**) e l’**ACK** viene richiesto per ogni trama ricevuta correttamente.

Immagine che contiene cerchio, design

Descrizione generata automaticamente

in questo caso se A e C volessero trasmettere a B non si sentirebbero

e senza l’uso di CSMA/CA avverrebbe una colisione

Nel caso di Collision Avoidance **con Backoff** se il canale è occupato si ascolta fino a che questo non sia libero e poi si aspetta un invtervallo di tempo pari a **DIFS + Backoff** (= numero di slot scelto casualmente).

Senza backoff

Immagine che contiene schermata, linea, testo, diagramma

Descrizione generata automaticamente

Con backoff

Immagine che contiene testo, linea, schermata, Carattere

Descrizione generata automaticamente

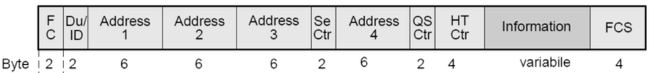
**6.7.2 Wireless LAN**

Oltre che in modo distribuito (**DCF**) l’accesso al mezo può essere gestito in modo centralizzato (**PCF**) e in questo caso una stazione agisce da **Point Coordinator** (**PC**).

Il **PC** controlla centralmente l’accesso al mezzo trasmissivo dando alle varie stazioni l’autorizzazione esplicita a trasmettere.

In questo caso si invia una trama speciale dopo un tempo **PIFS** (**=PCF interframe space**) tale che   
**SIFS < PIFS < DIFS** (ciò garantisce che non possa iniziare la finestra di contesa per trame “normali”, che richiedono un intervallo **DIFS**, ma possono essere inviati prima **CTS** o **ACK** che richiedono invece un tempo **SIFS**.

Il **formato della trama** è diviso in:



* **Duration/ID**, ossia l’occupazione prevista del canale espressa in , oppure riporta un ID della stazione che sta inviando una trama in risposta ad un’interrogazione.
* **Address 1, 2, 3, 4**, sono rispettivamente: indirizzo di stazione mobile sorgente, di stazione mobile destinazione, di ricevitore e trasmettitore.
* **Sequence control**, formato dal numero di frammento nella trama (primi 4 bit) e dal numero di trama (ultimi 12 bit).
* **QoS/HTcontrol**, ossia le informazioni per gestire la qualità del servizio.
* **FCS**, usato per il controllo di errore.
* **Information**, 0-2312 byte (payload).
* **Frame control** (**FC**), dove si trovano la versione del protocollo, il tipo, il sottotipo, il TDS e FDS, more frag, retry, power management, more data, protected frame e l’+HTC/Order.

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

**6.8 WPAN (Wireless Personal Area Network)**

Le WPAN sono caratterizzate da un raggio piccolo, come la potenza e il costo. Vengono usate per comunicazione di dispositivi in un “Personal Operating Space”, lo standard usato è l’IEEE 802.15.

**6.8.1 Bluetooth**

Usufruente della tecnologia radio ha un costo basso ed un raggio che va dai 10 ai 20 metri, una complessità scarsa e una banda di 2.4 GHz.

L’architettura web più semplice definita in bluetooth è la Piconet: un network ad hoc composto da 2 o più dispositivi dove uno di questi agisce da “**master**” e gli altri da “**slave**”, la comunicazione può avvenire soltanto tra master e slave e non direttamente tra quest’ultimi.

In una piconet possono esserci un massimo di **7** slaves, gli altri possono essere in stand-by on parked (fino a 256).

Immagine che contiene testo, schermata, cerchio, orologio

Descrizione generata automaticamente

Al livello fisico troviamo **Frquency Hopping** (**FH**), dove la **FH sequence** è pseudo-randomica e determinata dal **master** che regola i canali; gli altri canali schiavi seguono la sequenza **fk** definita dal master.

Il pacchetto è formato da:

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

* **Access code**, usato per la sincronizzazione e la piconet identification.
* **Header**, usato per Link Control (LC) che in clude anche il retransmission scheme.
* **Payload**

**6.9 ZigBee**

Ha un costo molto basso (soli 2 dollari per l’hardware), un raggio limitato a 10 metri, una bassa latenza e una grande efficienza nell’uso di energia.

Questi pregi sono però seguiti da una grandi problemi di incompatibilità/interoperabilità.

**6.10 Tipi di device**

I device usati nelle PAN sono:

* **PAN** **Coordinator**, i quali sono responsabili di una PAN e controllano le associazioni/dissociazioni.
* **Full Function Device** (**FFD**), possono mandare segnali, comunicare con altri FFD, instradare frame, agire da coordinatori PAN.

Hanno bisogno di un alimentatore.

* **Reduced Function Device** (**RFD**) i quali non possono instradare frame, ne comunicare con altri RFD, cosa invce possibile con gli FFD.

**7 Protocolli di routing in IP**

**7.1 RIP (Routing Information Protocol)**

Questo protocollo si affida ad UDP come protocollo di trasporto utilizzando la “well known port”, basato su Distance Vector ed una metrica di costo ad hop count con massima distanza uguale a 15 e aggiornamenti sul DV inviati ogni 30 secondi.

È di facile implementazione ma ha un’elevata complessità, possono verificarsi loop ed è quindi possibile un   
count-to-infinity.

Le **request** sono inviate da un router che si è appena connesso oppure ha alcune entry in scadenza.

Con queste si possono richiedere ai propri vicini entry specifiche oppure tutte le entry in loro possesso, e le **response** possono essere **solicited**, cioè inviate in risposta ad una request, oppure **unsolicited**, inviate periodicamente (ogni 30 s).

Altre caratteristiche sono:

* **Triggered update**: appena camiba il costo di una rotta la nuova tabella di routing è inviata immediataente.
* **Split horizon**
* **Poison reverse**.

I limiti sono composti dal fatto che l’hop count è una metrica di costo troppo semplice, inoltre converge lentamente (router a distanza N hanno bisogno di N passei dell’algoritmo = 30\*N secondi) ed è inoltre limitato a reti di piccole dimensioni (poiché DV = 16 vuol dire irraggiungibile).

**7.2 OSPF (Open Shortest Path First)**

Adotta lo shortest path con Dijkstra e ad ogni nodo conosce l’intera toppologia del routing domain, i router inviano **Link State Packe** (**LSP**) a tutti gli altri nodi nel routing domain.

Questi vengono inoltrati attraverso il flooding (in broadcast), periodicamente o in seguito ad eventi di modifica dei costi.

È caratterizzato da una **metrica di costo generica**, configurata dall’amministratore di rete, dal **load balancing dinamico**, ottenuto specificando rotte multiple e supporta **routing gerarchico**.

**7.3 BGP (Border Gateway Protocol)**

Consente a router di diversi AS di scambiarsi informazioni di routing.

Usa un algoritmo di tipo **path vector** ed è indipendente dai protocolli IGP adottati dentro gli AS.

Esistono due tipologie di BGP:

* **eBGP** (**external BGP**), usato da diversi AS border routers per scambiare tra loro informazioni di routing.
* **iBGP** (**internal BGP**), usato da un AS border router per propagare l’informazione di routing dentro l’AS.

**7.3.1 Il path vector**

Simile al distance vector nelle info scambiate tra i nodi non è però indicata una distanza dalla destinazione, bensì l’intero percorso verso di essa.

Agli AS è assegnato un **Autonomous System Number** (**ASN**) globale da IANA (come per indirizzi IP).

In realtà il messaggio di path vector che si scambiano due router EG vicini non contiene solo il percorso ma anche degli attributi che si distinguono in **attributi obbligatori** e **attributi facoltativi**.

Tra quelli **obbligatori** troviamo:

* **Origin**, indica il protocollo IGP da cui proviene l’informazione.
* **AS\_PATH**, indica la sequenza di AS attraversati.
* **NEXT\_HOP**, indica il prossimo router.

Per quanto riguarda lo scambio dei messaggi ogni router BGP invia il proprio path vector ai router BGP vicini (peers), e l’informazione del path vector è trasmessa su connessioni TCP.

I tipi di messaggio sono:

* **OPEN**: apre la connessione TCP e gestisce l’autenticazione reciproca dei router.
* **UPDATE**: annuncia una nuova rotta (o ne annulla una vecchia).
* **KEEPALIVE**: mantiene attiva la connessione in caso di assenza di UPDATE (usato anche come ACK ai messaggi OPEN).
* **NOTIFICATION**: notifica errori in messaggi precedenti (usato anche per chiudere la connessione).

Un router BGP che riceve un path vector da un peer può decidere o meno di aggiungere alla propria tabella di routing la destinazione specificata o inoltrare il path vector ai suoi vicini.

Ciò viene fatto in base alla **politica di routing** implementata.