Appunti di Reti di Elaboratori

Giovanni Varricchione

12/07/2017

Indice

Note sugli Appunti

Questi appunti coprono l'intero programma del corso di Reti di Elaboratori tenuto dalla Prof.ssa Maselli nell'anno accademico 2016/2017.

Gli appunti sono basati sulle slide fornite dalla professoressa (reperibili al seguente link; si noti che sono relative all'a.a. 2016/2017) e sul libro di testo **Behrouz A. Forouzan e Firouz Mosharraf, "Reti di Calcolatori Un approccio top down", versione italiana, Mc Graw Hill, marzo 2013** .

L'introduzione riguarda concetti generali delle reti, non solo di elaboratori; successivamente viene introdotto lo stack TCP/IP, seguito poi da un approfondimento (in ordine) per ognuno dei seguenti livelli:

1. Applicazione;
2. Trasporto;
3. Rete;
4. Collegamento.

Verranno infine introdotti alcuni concetti sulle LAN Wireless, sulla tecnologia Bluetooth e sul CDMA.

Note sui Formati delle Intestazioni

I formati degli Header dei protocolli saranno divisi in righe, **ogni riga** equivale a **32 bit** e la **dimensione di ogni campo** è data dalla **divisione in parti uguali dei bit rimanenti sulla riga** (ad esempio, se su una riga ci sono 5 campi, di cui uno ha dimensione pari a 16 bit, allora tutti gli altri 4 campi avranno come dimensione $\frac{32-16}{4}$, quindi 4 bit), a meno che non sia specificato diversamente.

Introduzione

In questa sezione verranno introdotti:

- Dispositivi delle Reti;
- Classificazione delle Reti;
- Tipi di Commutazione;
- Internet;
- Capacità e Analisi delle Prestazioni delle Reti;

Una rete è fondamentalmente composta da vari dispositivi che si scambiano informazioni tra di loro; andiamo a vedere nel particolare quali possono essere.

Dispositivi delle Reti

In generale, in una rete i dispositivi vengono divisi in due categorie:

1. Terminali:
    - Host (Utente);
    - Server;
2. Dispositivi di Interconnessione:
    - Router (collega le reti);
    - Switch (collega terminali locali);
    - Modem (codifica i dati).

I dispositivi di ogni rete devono essere in qualche modo collegati fra loro, questo compito è delegato ai Collegamenti (o Link) che possono essere di due tipi:

1. Cablati (se c'è un mezzo fisico che collega i dispositivi)
    - Doppino Intrecciato;
    - Cavo Coassiale;
    - Fibra Ottica;
2. Wireless (il segnale è trasmesso sullo spettro elettromagnetico, di conseguenza è suscettibile ad interferenze).

Classificazione delle Reti

Le Reti sono classificate in base alla loro dimensione, nel seguente elenco vengono ordinate in ordine di dimensione crescente:

- *Vicinity*: PAN (Personal Area Network);
- *Building*: LAN (Local Area Network);
- *City*: MAN (Metropolitan Area Network);
- *Country*: WAN (Wide Area Network);
- *Planet*: The Internet (rete di reti);

Di queste 5, ci interessano particolarmente le strutture delle reti LAN e WAN; si noti che oggi reti LAN e WAN sono spesso collegate fra di loro, utilizzando i Router per instradare i pacchetti fra se stesse. Questa struttura è chiamata "internet" (che si differenzia dall'Internet, essendo una rete di reti più piccola).

Reti LAN

Una rete privata, essa è identificata da un indirizzo nel momento in cui viene connessa ad altre LAN e/o WAN e si dividono in:

- *a Cavo Condiviso*, in cui ogni pacchetto di un terminale è inviato a tutti gli altri terminali ("Broadcast");
- *con Switch*, in cui un dispositivo chiamato "Switch di Interconnessione" è collegato a tutti i terminali e instrada i pacchetti dal mittente al destinatario.

Reti WAN

Una rete che può servire città, regioni e/o nazioni; essa generalmente è gestita da un ISP (Internet Service Provider). Le reti WAN si differenziano in base al tipo di collegamento che c'è tra i mezzi di comunicazione:

- *Punto-Punto*, se collega due mezzi di comunicazione tramite un mezzo dedicato (sia esso cablato o wireless);
- *a Commutazione*, se collega più switch di diverse reti (questa viene usata nelle Dorsali di Internet).

Tipi di Commutazione

Per ogni rete è cruciale individuare il metodo attraverso cui viene determinato il percorso dei pacchetti fra due terminali; per questo motivo le reti vengono divise in due famiglie:

1. *a Commutazione di Circuito*, in cui viene sempre garantito un collegamento dedicato fra i due terminali per tutta la durata della comunicazione. Per garantire ciò, le risorse di rete vengono divise e quindi allocate ai singoli collegamenti (*Frequency* o *Time* Division Multiplexing);

2. *a Commutazione di Pacchetto*, in cui i pacchetti di un singolo messaggio possono prendere strade qualsiasi, il tutto per garantire una maggiore efficienza delle rete.

Internet

Internet consiste di migliaia di reti interconnesse fra di loro.

Per poter accedere ad Internet, un utente deve essere fisicamente collegato ad un ISP; tale collegamento è garantito da una parte di rete che viene detta *Rete di Accesso*.

Metodi di Accesso

Il collegamento ad Internet può avvenire in tre modi diversi:

1. *via Rete Telefonica*, quando si crea una WAN punto-punto attraverso la rete telefonica:

    - *Dial-Up*, in cui il modem è installato sulla linea telefonica, impossibilità di utilizzare la rete telefonica e navigare su Internet contemporaneamente;
    - *DSL*, che divide la rete in tre fasce in modo da permettere l'utilizzo della linea telefonica e di internet contemporaneamente (l'accesso ad Internet è diviso in Upstream e Downstream);

2. *Ethernet*, in cui lo Switch della LAN è collegato ad un router privato connesso a sua volta ad un router della dorsale;

3. *Wireless*, diviso in:

    - *Wi-Fi*, in cui un Access Point è collegato ad una rete Ethernet Cablata ed ha un raggio di qualche decina di metri;
    - *Cellulare*, in cui l'Access Point appartiene ad una compagnia telefonica (è detto *Base Station*) ed ha un raggio di km.

Struttura

Gli ISP sono divisi in vari livelli, classificati sempre in base alla loro grandezza:

- ISP di livello 1 (sono connessi fra di loro e operano a livello nazionale);
- ISP di livello 2 (sono connessi a quelli di livello 1 e operano a livello distrettuale);
- ISP di livello 3 (sono connessi a quelli di livello due e sono le più vicine ai terminali).

Capacità e Analisi delle Prestazioni delle Reti

Queste due caratteristiche di ogni rete sono calcolate in base a 4 elementi:

1. *Ampiezza di Banda* (o *Bandwidth*);
2. *Throughput*;
3. *Latenza* (o *Ritardo*);
4. *Perdita dei Pacchetti*.

Bandwidth

La Bandwidth misura due grandezze:

1. la quantità di hertz che rappresenta la larghezza dell'intervallo di frequenze utilizzato dal mezzo;
2. il *bit rate*, ovvero la quantità di bit/secondo che un link supporta al massimo.

Si noti che le due grandezze sono strettamente correlate.

Throughput

Il Throughput indica la quantità di bit/secondo garantita *effettivamente* dal link: difatti il bit rate ne rappresenta soltanto un potenziale, in quanto il throughput sarà sempre minore.

In un generico percorso con n link, il throughput è determinato dalla seguente formula:

$$\min\{T_1, T_2, \ldots, T_n\}$$

Dove T_i è il throughput dell'*i-esimo* link.

Ritardo e Perdita dei Pacchetti

Innanzitutto, per Ritardo (o Latenza), si intende il tempo che impiega un pacchetto ad arrivare a destinazione da quando parte il primo bit. Nella Commutazione di Pacchetto, i singoli pacchetti vengono accodati nei buffer dei router, a questo punto possono verificarsi tre eventi:

1. Il buffer è vuoto ed il nuovo pacchetto viene quindi spedito subito;
2. Nel buffer ci sono altri pacchetti in attesa di essere inviati, l'ultimo pacchetto viene quindi messo in coda e deve attendere l'invio di quelli già presenti prima di poter essere inviato (Ritardo);
3. Il buffer è pieno, e quindi il nuovo pacchetto è scartato (Perdita).

Di seguito vengono esposte alcune cause dei ritardi:

- *Ritardo di Elaborazione del Nodo*, che consiste nel ritardo generato dal controllo degli errori;
- *Ritardo di Accodamento*, che equivale all'attesa nella coda del router, si può stimare con la formula $\frac{La}{R}$, dove:
  - L è la lunghezza del pacchetto in bit;
  - a è il tasso medio di arrivo dei pacchetti al secondo;
  - R è il rate bit/s del link;
- *Ritardo di Trasmissione*, che equivale al tempo necessario per immettere tutti i bit del pacchetto sul link; è uguale al rapporto $\frac{L}{R}$;
- *Ritardo di Propagazione*, che corrisponde al tempo che un bit impiega per percorrere il link; è dato dal rapporto $\frac{d}{s}$, dove:
  - d è la lunghezza del link;
  - s è la velocità di propagazione.

Stack TCP/IP

Prima di introdurre lo Stack Protocollare TCP/IP, occorre definire cosa è un **protocollo**: per protocollo si intende un insieme di convenzioni che dettano le regole per la comunicazione fra una o più entità; esso va rispettato sia dal mittente che dal destinatario.

Ogni protocollo, seguendo il principio del *Divide et Impera*, viene spesso diviso in più *layers* che possono essere immaginati come i "livelli interni" del singolo protocollo: ogni layer costituirà poi a sua volta un singolo protocollo.

Lo ***Stack TCP/IP*** è una *gerarchia di protocolli* che viene utilizzata in Internet; essendo una gerarchia, ogni livello fornisce servizi al livello superiore, mentre ne riceve da quello inferiore. I livelli dello stack sono 5:

Stack TCP/IP	
Applicazione	Software
Trasporto	Software
Rete	Software e Hardware
Collegamento	Software e Hardware
Fisico	Hardware

Come infarinatura, i livelli dello stack possono essere descritti nel seguente modo:

1. *Applicazione*: costituito dalle applicazioni, ovvero i software che offrono servizi all'utente (e.g. HTTP, SMTP, FTP, DNS), i pacchetti sono detti *messaggi*;

2. *Trasporto*: trasferimento dei pacchetti tra due terminali, i protocolli principali sono il TCP (affidabile, in cui i pacchetti sono detti *segmenti*) e l'UDP (inaffidabile, in cui i pacchetti sono detti *datagrammi utente*);

3. *Rete*: instradamento dei pacchetti da mittente a destinatario, il protocollo principale è l'IP, in cui i pacchetti sono detti *datagrammi*;

4. *Collegamento*: instradamento dei pacchetti da un nodo al successivo nel percorso da mittente a destinatario, i protocolli principali sono l'Ethernet, il Wi-Fi ed il PPP, in cui i pacchetti sono detti *frame*;

5. *Fisico*: trasferimento dei bit a livello fisico.

Quando un pacchetto viaggia da un terminale ad un altro, esso attraversa tutto (o in parte, come si vedrà in seguito) lo stack TCP/IP. Una particolarità dello stack è che fra due livelli uguali dei terminali c'è una connessione diretta ma non fisica che viene detta *"connessione logica"*.

Durante la discesa e la salita ai pacchetti viene aggiunto o tolto ciò che si definisce *"header di livello"* che contiene informazioni relative a quel livello e sono necessarie per il corretto funzionamento della trasmissione; nella discesa si verifica l'*"Incapsulamento"*, mentre nella salita si verifica il *"Decapsulamento"*.

Altre due funzionalità garantite dai protocolli sono il *Multiplexing* ed il *Demultiplexing* che consentono di ricevere o inviare pacchetti da o a più protocolli.

Livello Applicazione

Il livello Applicazione è il livello dello Stack TCP/IP che ha l'obiettivo di **fornire servizi all'utente** tramite i suoi protocolli e riceve servizi dal livello Trasporto.

I protocolli del livello Applicazione sono divisi in due categorie:

1. *Standard*, ovvero quei protocolli che sono certificati e documentati dalle autorità di Internet;
2. *Non Standard*, ovvero quei protocolli di utenti privati che non necessitano autorizzazioni (un esempio potrebbe essere un protocollo di comunicazione di un'azienda privata);

Una caratteristica che contraddistingue i protocolli del Livello Applicazione (sia quelli standard che non) è la loro *architettura*, che può essere una delle seguenti tre:

- *Client-Server*, in cui:
  1. il **Client** è colui che chiede il servizio ed è in funzione solo quando necessita di tale servizio;
  2. il **Server** è colui che offre il servizio ed è sempre in funzione, in attesa degli eventuali Client;
- *Peer-to-Peer*, abbreviata in **P2P**, in cui non esiste gerarchia fra gli utenti e tutti quanti mettono a disposizione le loro risorse per offrire servizi, ma anche per poterli ricevere da altri (ritorneremo su quest'architettura in seguito);
- *Ibrida*, ovvero un'architettura che ha caratteristiche tipiche sia del Client-Server che del P2P (esempi sono le architetture P2P con struttura *Centralizzata*).

3.1.0.1 Numeri di Porta infine, un concetto molto importante per i protocolli del livello Applicazione è il *numero di porta*, che serve ad identificare il processo, e quindi il protocollo, nel terminale. Il numero di porta è un intero lungo 16 bit (può assumere quindi valori compresi fra 0 e 65535), tuttavia vi sono tre fasce che sono dedicate a determinati tipi di protocolli:

1. **da 0 a 1023**, ovvero le *well known ports* che sono le porte dei protocolli Standard;
2. **da 1024 a 49151**, ovvero le porte *registrabili* per i protocolli Non Standard che possono quindi essere usate da protocolli definiti da aziende o privati;
3. **da 49152 a 65535**, ovvero le porte *effimere* che sono usati dai Client per mettersi in contatto con i Server.

HTTP

HTTP (*HyperText Transfer Protocol*) è un protocollo che **permette agli utenti di fare richieste alle pagine Web**; poiché si affida al protocollo TCP del livello Trasporto, garantisce che non vi siano errori o dati persi nelle risposte alle richieste, essendo il TCP stesso un protocollo affidabile.

Utilizza la **porta 80**.

In base alla versione del protocollo, la connessione Client-Server può essere di due tipi diversi:

1. *Persistente* per le versioni **dalla 1.1 in poi**, una connessione che o viene chiusa esplicitamente dal Client o viene chiusa da un timeout;
2. *Non Persistente* per le versioni **precedenti alla 1.1**, una connessione che viene chiusa ad ogni risposta del server, quindi per ogni richiesta bisogna crearne una.

3.1.1.1 Formato dei Messaggi **NOTA**: d'ora in poi verranno utilizzate le seguenti abbreviazioni per ci caratteri specificati nei formati:

- *spazio*: sp;
- *carriage return*: cr;
- *line field*: lf.

Il formato varia nel caso in cui il messaggio sia una richiesta o una risposta:

- ***Richiesta***:
  1. *Riga di Richiesta*:
     Metodo sp **URL** sp **Versione** cr lf
  2. *Righe di Intestazione*, possono essere più di una e contengono le informazioni aggiuntive sulle richieste:
     Nome Intestazione : sp **Valore** cr lf
  3. *Riga Vuota*, serve a dividere le Righe di Intestazione dal Corpo e consiste in un cr ed un lf;
  4. *Corpo*, contiene un numero di righe variabili (dipende dal tipo di richiesta).
- ***Risposta***:
  1. *Riga di Stato*:
     Versione sp **Codice di Stato** sp **Frase di Stato** cr lf
  2. *Righe di Intestazione*, come nella Richiesta, possono essere più di una:
     Nome Intestazione : sp **Valore** cr lf
  3. *Riga Vuota*, vedi Richiesta;
  4. *Corpo*, vedi Richiesta.

Metodi i metodi sono delle parole chiave che servono ad indicare il servizio richiesto dal Client:

- *GET*, richiesta documento;
- *HEAD*, richiesta informazioni documento;
- *PUT*, upload documento;
- *POST*, upload dati;
- *TRACE*, usato per debugging dal lato server;
- *CONNECT*, usato dai server proxy;
- *DELETE*, cancella pagina web;
- *OPTIONS*, richiesta informazioni pagina web.

Codici e Frasi di Stato usati nella Riga di Stato della Risposta, sintetizzano l'esito della Richiesta:

- 100-199: risposta alla richiesta nel corpo;
- 200-299: richiesta eseguita con successo;
- 300-399: Client redirezionato ad un URL diverso;
- 400-499: errore lato Client;
- 500-599: errore lato Server.

3.1.1.2 Richieste Condizionali le richieste possono anche essere accompagnate da condizioni che si devono verificare affinché il Server invii una risposta al Client; questo tipo di richieste è molto usato dai Server Proxy.

3.1.1.3 Caching per ovviare al carico di lavoro che si impone sul Server, spesso si usano dei Server detti *"Server Proxy"* che agiscono da intermediari fra il Client ed il Server ma anche da "cache" per il Server (memorizzano temporaneamente le ultime risposte del Server). Si noti come il Server Proxy agisca sia da client che da server.

3.1.1.4 Cookie sono dei file che mantengono varie informazioni sui Client: ad ogni Client che invia una richiesta ad un Server ne viene assegnato uno, che è identificato dal suo numero; il Server mantiene quindi una tabella dei cookie in cui mantiene gli identificativi dei Client che hanno inviato una richiesta recentemente.

Per aggiornare le informazioni che si trovano nel cookie, ogni volta che il Client invia una richiesta al Server nell'intestazione del messaggio mette il suo identificativo, mentre quando il Server assegna il cookie al Client l'identificativo si trova nell'intestazione della risposta.

FTP

FTP (*File Transfer Protocol*) è un protocollo che viene usato per il **trasferimento di file da un host ad un altro**; nonostante ciò sia già possibile con il protocollo HTTP, l'FTP è consigliabile essendo dedicato proprio a operazioni con file. Per garantire il servizio, l'FTP utilizza il protocollo TCP a livello Trasporto.

Una particolarità dell'FTP è che la sua struttura differisce fra Client e Server:

- nel **Server** vi sono solo due componenti: il *Processo di Controllo* ed il *Processo di Trasferimento Dati*;
- nel **Client** invece le componenti sono tre: oltre a quelle del Server esiste anche l'*Interfaccia Utente*.

3.1.2.1 Connessioni Client-Server fra Client e Server nell'FTP si stabiliscono due connessioni diverse, che connettono due componenti in comune:

1. *Connessione Controllo*, si stabilisce fra i due **Processi di Controllo** sulla **porta 21**. Questa connessione è **persistente** e rimane attiva per tutta la durata della connessione FTP;
2. *Connessione Dati*, si stabilisce fra i due **Processi di Trasferimento Dati** sulla **porta 20** e presuppone l'esistenza di una Connessione Controllo già attiva fra i due host. Questa connessione è **non persistente**, ed è dedicata al trasferimento di un singolo file, dopo il quale la connessione viene chiusa.

3.1.2.2 Procedimento Apertura Connessione Dati una Connessione Dati è stabilita con la seguente sequenza di operazioni:

1. il Client effettua un'apertura *passiva* (ovvero rimane in attesa della risposta del Server), con il comando PORT e usando una porta effimera;
2. il Server, ricevuta la richiesta, effettua l'apertura *attiva* sulla porta 20;
3. prima di inviare il file, il Client deve specificare le caratteristiche del file che sta trasferendo al Server.

3.1.2.3 Formato Messaggi di seguito i formati dei due tipi di messaggi utilizzati nell'FTP:

- *Comando*: **Comando (keyword)** [argomenti] cr lf
- *Risposta*: **Codice (3 cifre)** descrizione cr lf

Si noti che nel formato del Comando gli argomenti sono opzionali, come indicato dalla presenza delle parentesi quadre, mentre nella Risposta la descrizione deve essere sempre presente.

Posta Elettronica: SMTP, POP3, IMAP

Essendo la posta elettronica una comunicazione **asincrona** e **unidirezionale** (ad ogni mail non corrisponde necessariamente una risposta, e se esiste una risposta essa può essere inviata anche non immediatamente), quindi bisogna prestare particolare attenzione a tutte le operazioni che la riguardano; per questo motivo sono stati creati svariati programmi e protocolli per agevolare la **creazione**, l'**invio** e la **lettura** di email.

3.1.3.1 Creazione la Creazione di email è gestita da un programma che è detto **UA** (*User Agent*).

Ogni email ha lo stesso formato composto da due parti: l'**Intestazione**, in cui è specificato l'indirizzo del mittente, del destinatario e l'oggetto, ed il **Corpo**, che contiene il messaggio. Gli indirizzi email sono caratterizzati infine da due parti (divise da una **@**):

Parte Locale @ **Nome di Dominio**

Dove abbiamo che:

- il *Nome di Dominio* è il Server di Posta su cui si trova la Mailbox dell'utente;
- la *Parte Locale* invece è il nome della Mailbox dell'utente.

Va notato che l'utilizzo di un UA **non** è necessario per poter creare email, in quanto è possibile utilizzare anche un semplice Browser se si sta utilizzando una **Webmail**.

3.1.3.2 Trasferimento: SMTP il Trasferimento delle email è un processo che coinvolge due diverse Mailbox: la prima è quella del mittente, mentre la seconda è quella del destinatario.

L'**MTA** (*Mail Transfer Agent*), attraverso l'utilizzo del protocollo **SMTP** (*Simple Mail Transfer Protocol*), permette di eseguire le seguenti operazioni:

1. Trasferire l'email dall'UA del mittente alla sua Mailbox (in questo caso, l'host del mittente agisce da Client, mentre la Mailbox da Server);
2. Trasferire l'email dalla Mailbox del mittente a quella del destinatario (in questo caso, la Mailbox del mittente è il Client, mentre la Mailbox del destinatario, è il Server).

SMTP **SMTP** (*Simple Mail Transfer Protocol*) è un protocollo dedicato al **trasporto di email**, caratterizzato da una connessione **persistente**, utilizza la **porta 25**.

Le fasi della consegna sono divise in tre parti:

1. *Apertura della Connessione*: il Server SMTP inizia la Connessione quando il Client si connette con una connessione TCP sulla porta 25;
2. *Trasferimento del Messaggio*: viene inviato un singolo messaggio a uno o più destinatari;
3. *Chiusura della Connessione*: infine il Client chiude autonomamente la connessione.

3.1.3.3 Lettura: POP3 e IMAP4 la Lettura consiste in una *pull* dal Server di Posta del destinatario, che può essere eseguita o da un UA oppure con una Webmail.

A seconda del metodo utilizzato, vengono usati protocolli diversi:

- nel caso dell'UA, bisogna usare o il **POP3** oppure l'**IMAP4** (in seguito verranno analizzate le differenze);
- nel caso della Webmail, bisogna usare l'**HTTP**.

POP3 **POP3** (*Post Office Protocol*) è un protocollo che compensa le poche operazioni disponibili con una struttura molto semplificata: sostanzialmente, consente soltanto di leggere le email ricevute e poi cancellarle o mantenerle salvate sul Server di posta (due modalità differenti).

Utilizza il protocollo TCP e la **porta 101**.

IMAP4 **IMAP4** (*Internet Message Access Protocol*) è un protocollo molto più strutturato rispetto al POP3 che consente però molte più operazioni, come l'accesso simultaneo a più Server di posta, il mantenimento di informazioni sull'utente da un accesso all'altro, la ricerca di email, ecc...

Utilizza il protocollo TCP e la **porta 143**.

3.1.3.4 MIME **MIME** (*Multipurpose Internet Mail Extensions*) è un protocollo che definisce nuove **caratteristiche per il formato delle email** (già definito dall'SMTP) e soprattutto permette agli UA di **trasformare dati non ASCII in dati ASCII** (specificando chiaramente la codifica da cui vanno tradotti), oltre a consentire l'invio di allegati o file non testuali.

DNS

Il **DNS** (*Domain Name System*) è un sistema e protocollo che è dedicato all'**associazione e alla traduzione dei nomi di siti web e degli indirizzi IP**: poiché gli indirizzi IP sono difficili da memorizzare, gli vengono assegnati dei nomi (più semplici da ricordare) in modo da permettere agli utenti di accedere alle pagine web.

In base alla dimensione della richiesta può utilizzare o il protocollo UDP (per richieste più piccole di 512 byte) oppure il protocollo TCP (per richieste oltre i 512 byte); utilizza la **porta 23**.

3.1.4.1 Struttura Gerarchica il sistema prevede una struttura **gerarchica** e **decentralizzata**, in modo da garantire efficienza nelle ricerche e anche la cosiddetta *fault tolerance*, ovvero la resistenza ai guasti dei Server principali (che, oltre ad essere svariati, sono a loro volta replicati su Server di backup).

Questa struttura gerarchica può essere vista come un albero in cui ci sono vari tipi di nodi:

- *Root*;
- *Top Level Domains* (o **TLD**, sono i domini come .it, .com, .edu, ecc...);
- *Authoritative*.

Importante inoltre è anche il concetto di "**Zona**", ovvero tutto il sottoalbero di cui un Server è responsabile nella struttura; tale responsabilità può essere anche delegata a Server che sono radici in questi sottoalberi.

3.1.4.2 Tipi di Server come già visto, nella gerarchia vi sono vari tipi di nodi; i Server sono infatti differenziati in base al loro ruolo nella struttura:

- *Server Root*: sono i Server che hanno come zona tutto l'albero DNS. Sono 13 e sono distribuiti in tutto il mondo, oltre ad avere varie repliche;
- *Server Primari e Secondari*: sono i Server che hanno la responsabilità sulla loro zona, possono essere detti anche "Authoritative" per tutti gli indirizzi IP di cui sono responsabili. La differenza fra i Primari e i Secondari è che i Primari possono modificare le zone di cui sono responsabili, mentre i Secondari agiscono da repliche per i Primari, senza poter modificare le zone (si noti che un Server può essere Primario per una zona ma Secondario per un'altra).

3.1.4.3 Domini i domini del DNS possono essere di diversi tipi:

- *Generici*: suddividono gli host in base ai loro scopi (.com, .org, ecc...);
- *Nazionali*: individuano con un'etichetta di due caratteri la Nazione dell'host (.it, .us, .fr, ecc...);

3.1.4.4 Risoluzione delle Richieste la risoluzione delle richieste è effettuata da un programma (presente in ogni host) chiamato *Resolver*. Il Resolver contatta il Server DNS locale del Client che può agire in due modalità diverse, in base al modo in cui vengono effettuate le richieste di traduzione:

- *Ricorsiva*: se la richiesta viaggia lungo l'albero DNS partendo dalla radice fino ad arrivare all'Authoritative Server per quel nome;
- *Iterativa*: se la richiesta viene effettuata sempre dal Server Locale, ogni richiesta quindi o torna l'indirizzo IP oppure l'indirizzo di un Server che si pensa possa avere la traduzione richiesta.

3.1.4.5 Caching in un sistema come il DNS ritorna molto utile il Caching dei dati, in modo da evitare sovraccarichi agli Authoritative Server di indirizzi molto richiesti. Sia per motivi di memoria, sia per motivi di validità, ognuno dei record contiene al suo interno un **TTL** (o *Time To Live*, espresso in secondi) che indica per quanto tempo il dato è ancora ritenuto valido.

Il generico Record di Risorsa è fatto nel seguente modo:

< **Nome del Dominio** ; **Tipo** ; **Classe** ; **TTL** ; **Valore** >

Dove, abbiamo:

- il **Nome del Dominio** è il nome di cui si richiede la traduzione;
- il **Tipo** indica come vanno interpretati il Nome del Dominio ed il Valore;
- la **Classe** indica la tipologia di rete;
- il **Valore** indica l'indirizzo IP associato al Nome del Dominio.

Il Tipo di ogni record caratterizza il Nome del Dominio ed il Valore a cui è associato, dando informazioni sul dato:

- *A*, associa un nome ad un indirizzo IPv4 a 32 bit:

  $$\text{Hostname} \rightarrow \text{IP Address}$$

- *CNAME*, associa un Alias (nome alternativo di un host) al Canonical Name:

  $$\text{Alias} \rightarrow \text{Canonical Name}$$

- *NS*, associa un dominio al suo Server di Competenza:

  $$\text{Domain Name} \rightarrow \text{Name Server}$$

- *MX*, associa un Alias al Mail Server Canonical Name:

  $$\text{Alias} \rightarrow \text{Mail Server Canonical Name}$$

Architettura P2P

Un protocollo con architettura P2P, a differenza di quella Client-Server, pone sullo stesso livello tutti gli host (detti *peer*) che lo utilizzano; ogni peer condivide con la rete tutte le risorse che vuole, in modo da poterle chiedere ad altri peer collegati (che a loro volta hanno reso disponibili le loro risorse).

Proprio per questa caratteristica, una grande difficoltà delle reti P2P è tener traccia della disponibilità di determinate risorse poiché i peer non sono Server, e quindi non sono sempre disponibili ad offrire i loro servizi.

3.1.5.1 Strutture Reti P2P

Centralizzate le reti P2P Centralizzate possono essere viste come una sorta di reti *ibride* fra una rete P2P ed una Client-Server, poichè contiene delle directory che contengono la lista dei peer e delle loro risorse ed agiscono effettivamente da Server.

Per questo motivo, queste reti sono particolarmente vulnerabili ad attacchi a queste directory centrali e alla congestione dei Server.

Decentralizzate le reti P2P Decentralizzate sono le vere e proprie reti P2P, senza "contaminazioni" Client-Server.

Possono essere organizzate in vario modo:

- *Strutturate*: reti in cui i peer sono organizzati in modo tale da ottenere una rendere più efficienti le operazioni, tuttavia diventa costoso il mantenimento della struttura quando si aggiungono molti peer;
- *Non Strutturate*: reti in cui i peer sono disposti in maniera casuale, le richieste sono inviate in flooding attraverso tutta la rete rendendone la risoluzione meno efficiente;
- *Gerarchica*: reti in cui vi sono dei peer detti **super peer** e che agiscono come le directory delle reti P2P Centralizzate. Le richieste vengono inviate in flooding solo attraverso i super peer, a differenza delle reti Non Strutturate, e in modo da non sovraccaricare i singoli super peer.

Livello Trasporto

Il livello Trasporto fornisce servizi al livello Applicazione mentre ne riceve dal livello Rete; il suo compito principale consiste nel **far comunicare i processi di due host collegati**, dove per *processo* si intende un programma in esecuzione; il processo Client e Server devono quindi essere identificati in maniera univoca e questo è possibile grazie ai *Socket Address*.

3.2.0.1 Socket Address un generico Socket Address è composto dall'indirizzo IP del suo host e dal numero di porta del processo (per l'approfondimento sui numeri di porta si rimanda al relativo paragrafo nell'introduzione del livello Applicazione), nel seguente modo:

Indirizzo IP | # di Porta

Controllo di Flusso e degli Errori

Il livello Trasporto, nel caso di determinati protocolli che vedremo in seguito, deve implementare due controlli che sono fondamentali per un'efficiente comunicazione fra due host: il primo è il *Controllo di Flusso* ed il secondo è il *Controllo degli Errori*.

3.2.1.1 Controllo di Flusso l'obiettivo del Controllo di Flusso è l'evitare il sovraccarico del produttore (ovvero il mittente) o del consumatore (ovvero il destinatario), che porterebbe all'eliminazione dei pacchetti eccessivi, con conseguente perdita di informazioni. Affinché ciò non avvenga, vengono utilizzati due buffer, uno dal lato del mittente e uno dal lato del destinatario:

- dal *lato mittente*, il livello Trasporto segnala al livello Applicazione quando ha il buffer pieno, in modo da fermare la produzione di pacchetti;
- dal *lato destinatario*, il livello Trasporto del destinatario segnala al livello Trasporto del mittente quando ha il buffer pieno, in modo da evitare l'invio di ulteriori pacchetti.

3.2.1.2 Controllo degli Errori poiché il protocollo principale del livello Rete (IP) è inaffidabile, il controllo dei pacchetti va implementato necessariamente al livello Trasporto.

Per effettuare il Controllo degli Errori i pacchetti che vengono inviati sono contrassegnati con un *Numero di Sequenza*, che viene specificato da m bit nell'intestazione del livello Trasporto; di conseguenza, questi valori vanno da 0 a 2^m-1.

Quindi, il destinatario, una volta **ricevuto senza errori** il pacchetto contrassegnato dal Numero di Sequenza x, invia al mittente un pacchetto detto *ACK* (ovvero Acknowledgment) che indica la corretta ricezione del pacchetto x. Tuttavia, poiché le ACK sono inviate solo se il pacchetto è ricevuto ed è corretto, il mittente non ha modo di sapere se il pacchetto è perso oppure è stato danneggiato, setta quindi per ogni pacchetto un timer che, una volta scaduto, indica la sua *potenziale* perdita (notare il "potenziale": il pacchetto potrebbe

essere bloccato nel traffico ma non effettivamente perso) o corruzione: a questo punto il mittente ne spedisce una copia, settando nuovamente il timer.

Per verificare che un messaggio ricevuto sia corretto, il Destinatario utilizza la Checksum e, facendo varie operazioni su di essa (come spiegato nella sezione della Checksum), determina se il pacchetto è intatto o corrotto.

3.2.1.3 Integrazione dei Controlli sfruttando le soluzioni che sono state implementate singolarmente per i due Controlli, è possibile implementare un buffer circolare su cui si posta una *finestra scorrevole* che racchiude determinate posizioni: queste sono le posizioni dei pacchetti che possono essere inviati o sono in attesa di un ACK dal destinatario; ad ogni ACK, viene liberata la relativa posizione ed eventualmente si fa scorrere la finestra scorrevole (se il pacchetto era il primo della finestra).

La grandezza del buffer è data dagli m bit che sono usati per rappresentare il Numero di Sequenza: ad m bit corrisponderanno quindi 2^m posizioni nel buffer.

Checksum

La Checksum è una sequenza di bit che viene inserita nell'header del livello Trasporto: essa serve al Destinatario per verificare che il messaggio ricevuto sia effettivamente uguale a quello che è stato trasmesso. Nei protocolli sicuri, mentre dal lato Destinatario è obbligatorio calcolare la Checksum, dal lato Mittente è opzionale, infatti, nel caso in cui il Mittente decidesse di non volerla calcolare, la pone uguale a 0 nell'header, se nel calcolo effettivo ottiene come valore proprio 0, allora setta a 1 tutti i bit.

Il calcolo della Checksum varia leggermente fra Mittente e Destinatario, di seguito sono elencati tutti i passaggi da compiere in entrambi i lati per poterla calcolare:

- *Lato Mittente*
  1. Dividi il messaggio in parole da 16 bit;
  2. checksum $\leftarrow$ 0;
  3. **Somma** tutte le parole (checksum compresa) col **complemento a 1**;
  4. Infine fai il complemento a 1 del risultato.
- *Lato Destinatario*
  1. Dividi il messaggio in parole da 16 bit;
  2. Somma tutte le parole col complemento a 1;
  3. Esegui il complemento a 1 del risultato;
  4. Se il valore è **diverso da 0** allora il pacchetto **va scartato**.

Servizi Connectionless e Connection-Oriented

Il livello Trasporto prevede vari protocolli che vengono classificati, in base alle loro caratteristiche, in due famiglie; anche i protocolli più usati (UDP e TCP) rientrano in questa classificazione.

3.2.3.1 Connectionless nei servizi *Connectionless* i protocolli sono molto più semplici e hanno meno overhead rispetto a quelli Connection-Oriented. Di seguito sono elencate le caratteristiche più importanti:

- Mittente e Destinatario sono **sempre pronti** a ricevere pacchetti;
- i pacchetti sono **indipendenti fra di loro** e soprattutto **non sono numerati**;
- di conseguenza, né il Controllo di Flusso, né il Controllo degli Errori sono implementati;
- garantiscono una connessione veloce, al prezzo della possibile perdita o dell'arrivo disordinato di pacchetti;
- l'**UDP** è un protocollo Connectionless;

3.2.3.2 Connection-Oriented i servizi *Connection-Oriented* garantiscono invece una maggiore sicurezza nell'invio dei pacchetti, al costo di un maggiore overhead rispetto ai Connectionless:

- prima di potersi scambiare i pacchetti, Mittente e Destinatario devono **stabilire una connessione**, tramite la *Three-Way-Handshake* (che è approfondita in questo sottoparagrafo della Connessione TCP);
- i pacchetti sono **numerati**;
- vengono quindi implementati il Controllo di Flusso ed il Controllo degli Errori;
- l'overhead come già detto è più pesante rispetto a quello dei Connectionless, tuttavia la trasmissione dei pacchetti è sicura;
- il **TCP** è un protocollo Connection-Oriented.

Protocolli Livello Trasporto

Di seguito verranno introdotti alcuni protocolli del livello Trasporto, con esempi sia di protocolli Connectionless che Connectio-Oriented. Seguirà quindi un approfondimento sui protocolli UDP e TCP.

3.2.4.1 Semplice il protocollo Semplice è un protocollo Connectionless che prevede che Mittente e Destinatario siano sempre pronti a inviare o ricevere pacchetti.

Il protocollo UDP usa una variante molto simile per gestire il trasferimento dei pacchetti.

3.2.4.2 Stop-and-Wait il protocollo Stop-and-Wait è un protocollo Connection-Oriented (implementa quindi sia il Controllo di Flusso che il Controllo degli Errori, proprio come è stato spiegato in precedenza) che prevede, sia per Mittente che per Destinatario, una finestra scorrevole di **dimensione unitaria**. Consente quindi l'invio di un solo pacchetto alla volta, di cui bisogna poi attendere l'ACK (che, per convenzione, contiene il Numero di Sequenza del prossimo pacchetto atteso).

Poiché la finestra scorrevole ha dimensione 1, per semplicità si preferisce usare come Numeri di Sequenza soltanto 0 e 1.

Questo protocollo è particolarmente inefficiente su bande che possono supportare un traffico molto più elevato di un singolo pacchetto alla volta.

3.2.4.3 Go Back N protocollo Connection-Oriented che prevede due finestre scorrevoli di **dimensioni diverse per Mittente e Destinatario**:

Mittente il Mittente ha una finestra scorrevole di dimensione $\mathbf{2^m\text{-}1}$, che chiameremo S_{size} (non si usa 2^{m} perchè crea problemi con i duplicati; per queste problematiche si rimanda al libro di testo e alle slide).

Dei pacchetti nella finestra, ci interessano particolarmente il **Numero di Sequenza** del **primo** (S_{f}) e dell'**ultimo** (S_{n}) pacchetto per determinare se il Mittente deve mettersi **in attesa delle ACK**: infatti, se $S_{\text{n}} = S_{\text{f}} + S_{\text{size}}$, allora il Mittente non può inviare altri pacchetti.

Un grande vantaggio di questo protocollo è la possibilità per il Mittente di inviare **più pacchetti** alla volta, senza dover attendere l'ACK per ognuno di essi; ciò permette anche l'utilizzo di un singolo timer valido soltanto per il primo pacchetto (S_{f}): se il timer scade allora il Mittente invia nuovamente tutti i pacchetti in **attesa di ACK** (per questo motivo è detto "Go Back N").

Destinatario il Destinatario ha invece una finestra scorrevole di dimensione **unitaria** che gli consente quindi di inviare l'ACK per un solo pacchetto alla volta: le ACK inviate dal Destinatario sono quindi dette **cumulative** poiché segnalano al Mittente che tutti i pacchetti con Numero di Sequenza minore di quello dell'ACK sono stati ricevuti correttamente (di conseguenza, al Destinatario non interessa se le ACK vanno perse).

Nel caso in cui il Destinatario ricevesse un pacchetto di cui ha già inviato l'ACK, invia al Mittente un'ACK col Numero di Sequenza del **prossimo** pacchetto atteso.

Per rendere la trasmissione ancora più efficiente si può implementare un buffer dal lato del Destinatario che gli consenta di tenere in memoria per un periodo tutti i pacchetti che hanno Numero di Sequenza maggiore del prossimo atteso, in modo da non scartare i pacchetti successivi.

Aritmetica mod 2^m una particolarità di questo protocollo (e del prossimo che vedremo) è che i calcoli relativi ai Numeri di Sequenza sono tutti da fare in aritmetica mod 2^m, in modo da utilizzare effettivamente un buffer circolare che, una volta completato il giro dei 2^m pacchetti ricominci daccapo.

Si noti che il protocollo Stop-and-Wait è un caso particolare del Go Back N con m = 0.

3.2.4.4 Selective Repeat protocollo Connection-Oriented che prevede sia per Mittente che per Destinatario una finestra scorrevole di dimensione $\mathbf{2^{m-1}}$ (come per il Go Back N, rimando al libro e alle slide per la spiegazione sul perché la dimensione non sia 2^m-1); questo protocollo in realtà è uno dei modi per risolvere il problema del Go Back N con i pacchetti che hanno Numero di Sequenza superiore al pacchetto in attesa (l'altro è l'allocazione di un buffer come abbiamo visto in precedenza) perché consente al Destinatario di poter ricevere e mandare le ACK di più pacchetti alla volta.

A differenza del Go Back N, nel Selective Repeat il Mittente setta un timer **per ogni** pacchetto in attesa di ACK, in questo modo ritrasmette solo quelli che sono in attesa da troppo tempo (da questo deriva il nome Selective Repeat).

Il Destinatario invece non invia più ACK cumulative, in quanto adesso invia un'ACK **per ogni** pacchetto ricevuto (anche se in disordine); la finestra scorrevole scorre solo quando viene ricevuto il primo pacchetto in attesa, a questo punto vengono passati tutti pacchetti per cui ha già inviato l'ACK.

UDP

UDP (*User Datagram Protocol*) è un protocollo per il livello Trasporto di tipo **Connectionless**, di conseguenza non è affidabile, non implementa né il Controllo di Flusso né il Controllo degli Errori e **non prevede la ritrasmissione** dei pacchetti dal lato del Mittente.

Tuttavia, proprio grazie a queste caratteristiche, è **vantaggioso in termini di velocità e di overhead**: è dunque molto utile per quelle applicazione che possono "permettersi" di perdere dei pacchetti nelle comunicazioni ma devono garantire ottime prestazioni a livello di velocità (ad esempio applicazione per chat vocali).

Per diminuire ulteriormente l'overhead, l'UDP non divide i pacchetti in User Datagram (Datagrammi Utente) e se li aspetta già divisi dal processo; inoltre, prevede pacchetti di **dimensione massima** pari a **65536** byte (si noti che di questi però solo 65508 al più possono essere byte di dati, in quanto 8 byte sono occupati dall'header dell'UDP e 20 dall'header dell'IP).

3.2.5.1 Formato Datagramma Utente

Porta Sorgente	Porta Destinazione
Dim. Datagramma	Checksum
Dati Pacchetto (↑ 65508 byte)	

TCP

TCP (*Transmission Control Protocol*) è un protocollo per il livello Trasporto di tipo **Connection-Oriented**, di conseguenza è affidabile , prevede l'apertura di una connessione prima delle trasmissioni, del Controllo di Flusso e del Controllo degli Errori.

Una particolarità del TCP è che i Numeri di Sequenza e di Riscontro sono **relativi ai byte** (la loro numerazione inizia da un valore casuale concordato nella creazione della connessione) del pacchetto con la seguente caratterizzazione:

- *Sequenza*: **primo** byte del pacchetto **attuale**;
- *Riscontro*: **primo** byte del **prossimo** pacchetto atteso.

3.2.6.1 Formato Segmento

Porta Sorgente		Porta Destinazione	
# Sequenza			
# Riscontro			
Lunghezza Header (4)	Riservati (6)	**Flag** (6)	Dim Finestra Ricevente (16)
Checksum		Puntatore Dati Urgenti	
Opzioni (↑ 40 byte)			

Lunghezza Header la lunghezza dell'Header varia da 20 a 60 byte (in base alle Opzioni), di conseguenza la sua dimensione viene rappresentata con 4 bit, rappresentandola in maniera sintetica con i valori che vanno da $\frac{20}{4}$ a $\frac{60}{4}$.

Flag in ogni header ci sono 6 bit di Flag che danno varie informazioni riguardo al pacchetto:

1. *URG*: se il segmento contiene dati detti "urgenti" (si rimanda al sottoparagrafo sull'Invio Dati per la spiegazione);
2. *ACK*: se il segmento contiene un riscontro valido;
3. *PSH*: se il segmento che è in arrivo contiene dati che vanno "pushati" (si rimanda al sottoparagrafo sull'Invio Dati per la spiegazione);
4. *RST*: segnale per indicare l'azzeramento della connessione per un errore grave;
5. *SYN*: utilizzato quando si apre la connessione per sincronizzare i # Sequenza;
6. *FIN*: segnala l'inizio della chiusura della connessione da parte di un host.

Puntatore Dati Urgenti puntatore che viene utilizzato **solo** nel caso in cui il bit **URG** è **settato a 1**: indica, a partire dall'inizio del segmento, fino a dove si trovano i dati *urgenti*.

3.2.6.2 Connessione TCP la connessione TCP si divide in 3 fasi: **Apertura**, **Invio Dati** e **Chiusura**:

Apertura l'Apertura della Connessione garantisce che la comunicazione fra gli host sia in modalità *full-duplex* (ovvero ogni host può sia inviare che ricevere segmenti) ed è effettuata con la ***Three-Way-Handshake***.

Nella Three-Way-Handshake il Server deve sempre trovarsi in uno stato di apertura *passiva* (ovvero è sempre pronto ad aprire connessioni) e bisogna eseguire i seguenti 3 passi:

1. il Client invia al Server un segmento col bit **SYN attivo** e col # Sequenza da cui comincerà ad inviare i segmenti;
2. il Server risponde al Client con un segmento con i bit **SYN** e **ACK attivi**, oltre ad inizializzare il suo # Sequenza;
3. il Client invia infine un segmento col bit **ACK attivo**.

Si noti come nel passo 2 il Server invii, oltre all'informazione del # Sequenza, anche l'ACK del segmento del Client inviato precedentemente: questa tecnica è detta ***piggybacking*** e consiste nell'incapsulamento di un messaggio (in questo caso l'ACK) all'interno di un altro (il SYN); essa viene usato regolarmente nel TCP al fine di ottimizzare la comunicazione.

Invio Dati come nell'Apertura, per inviare le ACK si usa la tecnica del *piggybacking*.

Per l'invio dei dati esistono due modalità speciali, il *Pushing* e l'*Urgent*, date rispettivamente dai bit **PSH** e **URG**:

- *Pushing*: se il pacchetto dal lato del Destinatario va inviato immediatamente al processo al livello Applicazione;
- *Urgent*: se ci sono dati nel pacchetto detti *urgenti*, essi possono essere riscontrati fuori sequenza (è molto importante in questo caso l'utilizzo del Puntatore Dati Urgenti).

Chiusura la Chiusura può essere effettuata da uno qualsiasi dei due host, in due modi diversi:

- *Three-Way-Handshake*: identica all'Apertura, ma usa il bit **FIN** al posto del SYN;
- *Four-Way-Handshake*: utile quando un host va messo in stato di *half close*, in cui può ancora ricevere dati ma non inviarli. Si può schematizzare nei seguenti 5 passaggi (solo 4 in realtà sono relativi alla Chiusura effettiva):
  1. FIN $\rightarrow$;
  2. ACK $\leftarrow$: qui si esegue l'half close poiché l'altro host non ha inviato a sua volta il segmento col bit FIN attivo;
  3. Dati $\leftarrow$;
  4. FIN $\leftarrow$: qui infine avviene la chiusura effettiva delle comunicazioni;
  5. ACK $\rightarrow$.

3.2.6.3 Controllo di Flusso e Controllo degli Errori poiché il TCP prevede una comunicazione full-duplex, ad ogni connessione equivalgono esattamente quattro finestre scorrevoli: due per host (una di Invio e una di Ricezione) e che sono relative ai byte. Una particolarità del TCP è che, in base alle ACK, alla congestione della rete e al Controllo di Flusso prevede una variazione delle dimensioni delle finestre, che va quindi comunicata in ogni segmento per tenere l'altro host sempre al corrente (formato del segmento, si veda il campo *Dim Finestra Ricevente*).

Le ACK possono essere sia **cumulative** che **selettive** (in questo caso sono dette SACK; disponibili solo dall'ultima versione del TCP), c'è soltanto un timer di invio per host e i segmenti possono essere bufferizzati da entrambi i lati (per evitare di scartare quelli successivi a quello atteso).

Per determinare la dimensione della finestra di Ricezione (*rwnd*) in un dato momento per un host, basta calcolare la seguente differenza:

$$rwnd = dim_{\text{buffer}} - \#byteDaConsumare$$

Controllo di Flusso il Controllo di Flusso è implementato con un sistema di feedback da parte del Destinatario che informa, in ogni segmento, il Mittente della dimensione attuale della sua finestra di Ricezione, di conseguenza è quest'ultimo ad adattare la sua finestra di Invio.

- *Mittente*: la finestra di Invio del Mittente si sposta da sinistra verso destra nel momento in cui riceve le ACK per i dati che ha inviato, può inoltre essere aumentata o ridotta in base al valore della *rwnd* che riceve nei segmenti inviatogli dal Destinatario;
- *Destinatario*: nel Destinatario la finestra di Ricezione è ridotta quando riceve dati che vengono bufferizzati, mentre è aumentata nel momento in cui i dati ricevuti vengono inviati al processo; in base a questi movimenti la *rwnd* che viene inviata nel segmento di ACK varia, diminuendo nel primo caso e aumentando nel secondo.

Si noti che la dimensione della finestra di Invio del Mittente è **sempre uguale** a quella della finestra di Ricezione del Destinatario.

Controllo degli Errori essendo il TCP un protocollo **affidabile**, deve gestire gli errori nei segmenti e soprattutto le **ritrasmissioni** quando i segmenti corrotti vengono scartati oppure persi; di conseguenza, implementa la Checksum in ogni segmento e vengono inviate le ACK per ogni segmento (tranne che per altre ACK).

Per la ***generazione di ACK*** il TCP usa sei regole:

1. implementazione del *piggybacking*: ogni segmento contiene sempre l'ACK attuale del Mittente;
2. *Delayed ACK*: quando si riceve un pacchetto viene settato un **timer di 500 ms** per l'eventuale arrivo di un altro segmento. Se alla fine del timer non è arrivato nessun segmento allora si invia l'ACK;
3. se invece si riceve un segmento entro i 500 ms viene **subito** inviata un'ACK;

4. se il Destinatario riceve un segmento fuori sequenza allora invia un'ACK del byte atteso, mentre bufferizza il segmento ricevuto;

5. appena si riceve **il segmento mancante** del punto 4, si invia l'ACK relativa (considerando anche i segmenti successivi ricevuti in precedenza che sono stati bufferizzati);

6. se si riceve un segmento **duplicato** allora si invia la copia dell'ACK relativa.

Invece, per quanto riguarda la ***ritrasmissione dei segmenti***, vanno considerate i seguenti due casi:

1. *Timer Scaduto*: viene settato un unico timer per il **primo** segmento, allo scadere si ritrasmette il primo segmento e si resetta il timer;

2. *3 ACK Duplicate*: quando si ricevono 3 ACK duplicate per lo stesso segmento probabilmente è perché è stato perso, di conseguenza avviene la cosiddetta *ritrasmissione veloce* del pacchetto.

3.2.6.4 Controllo della Congestione il Controllo del Flusso controlla solo la congestione dal lato Destinatario, di conseguenza la **congestione nei Nodi intermedi** è ancora possibile: bisogna quindi predisporre una **finestra di congestione** (*cwnd*), ponendola uguale al **numero di segmenti inviati e non riscontrati**.
Di conseguenza, abbiamo che la dimensione della finestra di invio è pari al seguente valore:

$$\min(rwnd, cwnd)$$

Sintomi della Congestione i sintomi della congestioni sono fondamentalmente i seguenti due:

1. *Timeout*: quando scade il timer d'invio (indica la presenza di una congestione più severa);

2. *3 ACK Duplicate*: quando il mittente riceve 3 ACK uguali (indica la presenza di congestione, ma non severa come nel caso del Timeout).

Questi due segnali vengono trattati in maniera diversa, ma solo dal **TCP Reno** in poi.

Gestione della Congestione il TCP gestisce la Congestione con tre fasi diverse, in cui al verificarsi di diverse condizioni, la cwnd viene modificata nel seguente modo:

$$cwnd = cwnd + 1$$

1. **SS** (*Slow Start*): in cui la cwnd è incrementata in **modo esponenziale** perché **ad ogni riscontro** viene modificata.
   Questa fase è bloccata nel momento in cui viene raggiunto il *ssthreshold* (*slow start threshold*);

2. **CA** (*Congestion Avoidance*): in cui la cwnd è incrementata in **modo lineare** perché viene modificata solo quando **è riscontrata tutta la cwnd**;

3. **FR** (*Fast Recovery*): usata **solo in Reno**, si attiva quando sono **ricevute 3 ACK duplicate** ed incrementa la cwnd in **modo esponenziale**, ma **solo** quando vengono **ricevute ACK duplicate**.
   Il TCP rimane in questa fase finché:
   - riceve una ACK **non** duplicata: va in *Congestion Avoidance*;
   - si verifica un Timeout: va in *Slow Start*.

3.2.6.5 Versioni del TCP di seguito verranno analizzate due versioni del TCP che trattano in maniera diversa i due sintomi della congestione e, mentre la prima utilizza soltanto le fasi SS e CA, la seconda le usa tutte e tre.

TCP Taho prevede solo le fasi SS e CA, e tratta in maniera **uguale** il Timeout e le 3 ACK Duplicate:

- se sta in **SS** allora *pone* $cwnd = 1$ e $ssthresh = \frac{cwnd}{2}$;
- se sta in **CA** allora *passa alla* **SS**, ed esegue le *stesse operazioni del primo punto*.

TCP Reno considera in maniera differente il Timeout e le 3 ACK Duplicate e prevede tutte le tre fasi descritte in precedenza:

- *Timeout*: va in **SS** *ponendo* $cwnd = 1$ e $ssthresh = \frac{cwnd}{2}$;
- *3 ACK Duplicate*: va in **FR** *ponendo* $cwnd = ssthresh + 3$ e $ssthresh = \frac{cwnd}{2}$ (in ordine, prima cambia l'ssthresh e poi la cwnd);
- invece, se si trova in **FR** e riceve una ACK **nuova**, allora va in **CA** ponendo $cwnd = ssthresh$.

3.2.6.6 Calcolo Timer d'Invio il valore del Timer d'Invio (RTO d'ora in poi) deve tenere conto di due considerazioni:

1. deve essere **maggiore** del **RTT** (*Round Trip Time*) che però, essendo molto variabile, può essere solo *stimato* con una *media ponderata*;
2. tuttavia, se viene impostato troppo grande, allora si ha una reazione lenta alla perdita dei segmenti.

In base al *SampleRTT* (ovvero l'RTT registrato al tempo *t + 1*) si calcolano sia l'*EstimatedRTT* che il *DevRTT* (questo serve ad avere un margine di sicurezza):

1. $EstimatedRTT_{t+1} = (1 - \alpha)EstimatedRTT_t + \alpha SampleRTT_{t+1}$;
2. $DevRTT = (1 - \beta)DevRTT + \beta|SampleRTT - EstimatedRTT|$

Si noti che, solitamente, $\alpha \sim 0,125$ e $\beta \sim 0,25$.

Infine, abbiamo che:

$$TimeoutInterval = EstimatedRTT + 4DevRTT$$

Livello Rete

Il Livello Rete ha il compito di garantire la **consegna dei pacchetti dati fra host** (i pacchetti sono detti "**datagrammi**"), prevede quindi l'utilizzo di **indirizzi logici** e di vari **protocolli di instradamento**.

Il Livello offre i seguenti servizi:

1. *Suddivisione in Pacchetti*: divide i dati in datagrammi, senza modificarli (tranne la **frammentazione**, quando serve);
2. *Instradamento* o *Routing*: instradamento del pacchetto da mittente a destinatario sul percorso migliore;
3. *Inoltro* o *Forwarding*: inoltro del pacchetto al prossimo host lungo il percorso.

3.3.0.1 Packet Switching descrive la struttura del Forwarding, che può essere di due tipi diversi:

1. *a Datagramma*: **Connectionless**, i pacchetti di uno stesso messaggio possono quindi seguire strade diverse (viene usato in Internet);
2. *a Circuito Virtuale*: **Connection-Oriented**, fra mittente e destinatario viene stabilita una connessione che viene detto *circuito virtuale*. Ad ogni circuito viene assegnato un numero, che viene usato anche per contrassegnare i pacchetti che viaggiano in tale ciruito (i pacchetti dello stesso messaggio attraversano la stessa strada).

3.3.0.2 Struttura dei Router il Livello Rete è l'**ultimo livello** che può essere implementato da un Router; essi sono dunque dedicati allo "smistamento" dei pacchetti, senza però potervi accedere.

In particolare, è importante analizzarne la struttura per capire come funziona il Livello Rete a livello hardware:

- Porte di *Input* e di *Output*: attraverso le prime ricevono i pacchetti mentre con le seconde li inviano;
- *Processore di Routing*: implementa le funzionalità del Livello Rete, in particolare ha anche il compito di organizzare la **forwarding table** in modo da rendere efficiente le operazioni di ricerca;
- *Switching Fabric*: si occupa del trasferimento dei dati dalle porte di Input a quelle di Output designate (in base al prossimo hop, si userà una certa porta di Output); la commutazione può essere eseguita in tre varianti:
    1. *Commutazione nella RAM*, che viene effettuata dalla CPU (occupandola quindi nel tempo e nelle risorse);
    2. *Commutazione tramite Bus*, che permette di trasferire un solo pacchetto alla volta dalle porte di Input a quelle di Output;
    3. *Commutazione a Crossbar Switch*, come la Commutazione *tramite Bus*, ma permette di trasferire più pacchetti alla volta.

Protocolli Livello Rete

Il Livello Rete utilizza vari protocolli (che vedremo in seguito) per garantire i servizi richiesti:

- *IP* (v4 o v6);
- *IGMP* (multicasting);
- *ICMP* (segnalazione errori e testing);
- *ARP* (associazione indirizzi IP - indirizzi MAC);
- *DHCP* (associazione di nuovi nodi ai loro indirizzi IP, in realtà si trova al Livello Applicazione).

IP

IP (*Internet Protocol*) implementa i **servizi del Livello Rete** ed è un protocollo **inaffidabile**, basato sui **datagrammi**.

3.3.2.1 Formato Datagrammi

Version (4)	HeaderLen (4)	ServiceType (8)	DatagramLen (16)
MessageID (16)		Flags (3)	FragmentedOffset (13)
TTL (8)	TransportLayerProtocol (8)	HeaderChecksum	
SourceIPAddress			
DestinationIPAddress			
Options (↑ 40 byte)			
DATI			

Dove abbiamo che:

- *HeaderLen*, ovvero la Lunghezza dell'Header, che viene espressa in unità di 4 byte (si noti che l'header va da 20 a 60 byte, quindi questo valore sarà sempre compreso fra 5 e 15);
- *Flags*, tre bit che danno informazioni sulla frammentazione del datagramma:
  1. il primo è *riservato*;
  2. *Do Not Fragment*: indica se il datagramma può essere frammentato ulteriormente;
  3. *More Fragments*: indica se ci sono altri frammenti per lo stesso datagramma oltre a questo (è 0 solo per l'ultimo);
- *FragmentedOffset*: serve a specificare l'**ordine** del frammento per poter **riassemblare a destinazione** il datagramma ed è pari al $\frac{\#primoByteFrammento}{8}$.

3.3.2.2 Frammentazione la frammentazione è un'operazione necessaria, in quanto diverse reti supportano **Maximum Transfer Unit diverse**, ovvero grandezze massime diverse: di conseguenza, quando un pacchetto che poteva transitare integro in una rete arriva ad un'altra rete in cui la *MTU* è minore della sua grandezza, esso va frammentato, per non poter perderne le informazioni.

I pacchetti vengono poi riassemblati quando arrivano al Destinatario, utilizzando il valore del campo *FragmentedOffset* per poterli riordinare.

3.3.2.3 Indirizzamento IPv4 ogni **interfaccia** che si trova nella rete (per interfaccia si intende una qualsiasi porta che collega un host ad un altro) deve poter essere identificata, per permettere l'invio di datagrammi ma anche il routing ed altre varie operazioni.

Il generico indirizzo IPv4 è formato nel seguente modo:

prefisso | **suffisso**

Sono formati da 32 bit (quindi possono esservi 2^{32} indirizzi in totale), il prefisso (che **identifica la rete**) ha n bit, mentre il suffisso (che **identifica l'host**) ne ha $32 - n$.

Indirizzamento Con Classi l'indirizzamento con Classi definisce in maniera **statica** la lunghezza dei prefissi, in base alla rete in cui l'host si trova:

- *A*: 8 bit prefisso, il primo bit è sempre 0;
- *B*: 16 bit prefisso, i primi due bit sono sempre 10;
- *C*: 24 bit prefisso, i primi tre bit sono sempre 110;
- *D*: usato per il multicasting, i primi quattro bit sono sempre 1110;
- *E*: riservato ad usi futuri, i primi quattro bit sono sempre 1111.

L'unico vantaggio di questo tipo di indirizzamento è che è **semplice capire il tipo di rete** in base agli indirizzi dei suoi host, tuttavia si è rivelato troppo **vincolante** per le reti.

Indirizzamento Senza Classi l'indirizzamento senza Classi prevede una lunghezza **variabile** per il prefisso che viene specificata nell'indirizzo a seguito di un `"/"`.

Notazione CIDR

byte . **byte** . **byte** . **byte** / **n**

Dove n è la lunghezza del prefisso (in base a questo valore, si avranno 2^{32-n} possibili indirizzi per una rete).

Maschera la *Maschera* è utilizzata per ottenere varie informazioni sulle caratteristiche di una rete: sia n il numero di bit del prefisso negli indirizzi della rete, allora la maschera è composta dai bit del prefisso settati a 1 ed i restanti $32 - n$ settati a 0.

Con la Maschera è possibile (si noti che le operazioni sono da eseguire tutte in binario):

- calcolare il **# di indirizzi della rete**: NOT(mask) + 1;
- *a partire da un qualsiasi indirizzo di quella rete*:
  - ottenere il **primo indirizzo**: indirizzo AND mask;
  - ottenere l'**ultimo indirizzo**: indirizzo OR (NOT(mask)).

Indirizzi IP Speciali esistono vari indirizzi IP che sono dedicati ad utilizzi speciali, di seguito sono elencati alcuni:

- 0.0.0.0 : è l'indirizzo che si dà l'host quando entra in una rete e ancora non gliene è stato assegnato uno;
- 255.255.255.255 : è l'indirizzo di *broadcast* sulla rete attuale;
- 127.x.y.z : è l'indirizzo di *loopback* (usato per i pacchetti che devono essere elaborati come pacchetti in arrivo ma non vengono immessi nella rete);
- da 224.x.y.z a 239.x.y.z : indirizzi usati per il *multicast* (si rimanda al relativo paragrafo).

DHCP

Il protocollo **DHCP** (*Dynamic Host Configuration Protocol*) permette agli host di **ottenere il proprio indirizzo IP** quando si collegano ad una rete; si noti che tale indirizzo può essere o *temporaneo* o *persistente* (in questo caso viene sempre associato lo stesso IP all'host quando si collega in quella rete).

In realtà, il DHCP è un **programma Client/Server** e si trova quindi al **livello Applicazione**.

3.3.3.1 Panoramica Funzionamento

si noti che **tutti** i messaggi previsti sono inviati in **broadcast**: questo per permettere ai Server DHCP di conoscere quali indirizzi sono occupati.

1. l'Host (Client) invia un messaggio *DHCP Discover*;
2. un Server risponde in con un messaggio *DHCP Offer*;
3. l'Host risponde in **unicast** al Server con un messaggio *DHCP Request*;
4. il Server infine, se l'indirizzo nel frattempo non è stato assegnato, risponde al Client inviandogli un messaggio con **indirizzo**, **maschera di rete**, **indirizzo del Router** e del **DNS**, oltre al *DHCP ACK*.

3.3.3.2 Formato Messaggi

OpCode	**HWType**	**HWLen**	**HCount**
Transaction ID			
Elapsed Time		Flags	
Client IP Address			
Your IP Address			
Server IP Address			
Gateway IP Address			
Client HW Address			
Server Name (↑ 64 byte)			
Boot File Name (↑ 128 byte)			
Options (↑ 64 byte			

Dove abbiamo:

- *OpCode*: è 1 per le **richieste**, 2 per le **risposte**;
- *HWLen*: è la lunghezza dell'indirizzo HW (si rimanda alla sezione sull'indirizzamento MAC);
- *HCount*: è il numero massimo di hop (ovvero **link attraversati**) che il pacchetto può fare;
- *Transaction ID*: viene usato dai Client e Server per identificare lo scambio dei messaggi;
- *Flags*: il primo bit indica se il messaggio è in broadcast (1) o in unicast (0), gli altri sono inutilizzati;
- *Client IP Address*: IP del Client (dal Client al Server, al primo punto è impostato a 0.0.0.0);
- *Your IP Address*: IP del Client (dal Server al Client);
- *Server IP Address*: IP del Server (è l'indirizzo broadcast se il Client ancora non lo conosce).

Sottoreti e NAT

Una *sottorete* è una parte di rete **isolata** che è **collegata** al resto della rete tramite **l'interfaccia di un solo router o host**.

L'indirizzo IP di un qualsiasi terminale nella sottorete possiede un **prefisso di sottorete** di **n bit** che serve ad individuarla, seguito dall'indirizzo effettivo dell'host nella sottorete, che ha $32 - n$ bit.

3.3.4.1 Reti Private vista la rapida diffusione delle sottoreti, è stata necessaria l'introduzione del concetto di *Rete Privata* per poter **"inglobare"** le sottoreti sotto un **unico indirizzo**, il che ha portato vari vantaggi:

1. la possibilità di usare indirizzi uguali per host in sottoreti diverse, aumentando di fatto il numero di host totali collegati alla rete;

2. aggiungere un livello di sicurezza per le sottoreti (gli host non possono essere indirizzati esplicitamente da fuori);

3. cambiare l'IP di un host di una sottorete senza dover informare tutta la rete del cambiamento.

3.3.4.2 NAT Il **NAT** (*Network Address Translation*) si occupa della **traduzione degli indirizzi privati** e di **"nascondere" le reti private locali**; ciò è possibile grazie ad un **router NAT** che mette in comunicazione la rete privata col resto della rete (si trova in mezzo).

Funzionamento NAT il router NAT mantiene una **tabella di traduzione NAT** in cui **associa ad ogni host** nella rete privata una **porta effimera** che usa quando invia messaggi all'esterno: il messaggio inviato avrà infatti come indirizzo del mittente l'indirizzo della rete privata e la porta effimera assegnata all'host; quando invece il messaggio viene da un host fuori dalla rete, la traduzione è effettuata al contrario (la porta di destinazione sarà la porta effimera usata precedentemente).

Forwarding IP

Consiste nell'invio del datagramma tramite l'hop corretto, usando la ***tabella di routing***, che **associa** gli **indirizzi di rete** con le **interfacce del router**. Il datagramma contiene l'IP dell'host destinatario, senza però specificarne la rete (per trovarla bisogna confrontare i prefissi della tabella di inoltro con l'indirizzo, e prendere il prefisso **più lungo** che combacia).

ICMP

ICMP (*Internet Control Message Protocol)* è un protocollo che offre servizi destinati alla **gestione degli errori** ed al **testing** (ad esempio usando i comandi quali il *ping* od il *traceroute*).

ICMP si trova "figurativamente" **sopra IP**, perché lo **usa per inviare i messaggi**.

3.3.6.1 Messaggi ICMP

Type (8)	**Code** (8)	**Checksum**
IP Header		
primi 8 byte datagramma		

L'Header IP e i "primi 8 byte datagramma" si riferiscono al datagramma che ha causato l'errore.

Routing

Il **Routing** consiste nell'operazione preliminare di **costruzione della tabella di routing**, usata poi dal Forwarding per inoltrare i datagrammi.

Per la creazione della tabella vengono usati ciò che sono detti "*Algoritmi di Instradamento*" per ottenere i **percorsi più efficienti**.

Tabella di Routing le tabelle di routing che vengono implementate in RIP ed OSPF contengono tre colonne, la prima indica la rete di destinazione, la seconda il prossimo router che si trova sul percorso per raggiungerla (quest'informazione viene usata nel Routing) e la terza il costo, ovvero il numero di hop, per raggiungerla:

Rete di Destinazione | **Prossimo Router** | **Costo**

3.3.7.1 Caratterizzazione Algoritmi di Instradamento gli Algoritmi di Instradamento sono differenziati in base a due caratteristiche:

1. Globale o Decentralizzato:
   - *Globale*: se ogni nodo riceve in input **tutti** i nodi ed i costi dei link → *link-state algorithm*;
   - *Decentralizzato*: se ogni nodo elabora il suo **vettore delle distanze** che poi invia ai suoi vicini → *distance-vector algorithm*;
2. Statico o Dinamico:
   - *Statico*: se non ricalcola spesso i cammini;
   - *Dinamico*: se ricalcola i cammini in base alla congestione ed al tipo di rete.

3.3.7.2 Link-State Algorithm il Link-State Algorithm è un algoritmo di instradamento globale che viene usato dal protocollo OSPF.

Prima di tutto, va costruito l'**LSDB** (*Link-State Database*): per fare ciò ogni nodo invia ai suoi vicini il **costo dei link ricorsivamente**, effettuando il cosiddetto *flooding* della rete.

A questo punto, ogni nodo, usando l'**algoritmo di Dijkstra** e l'LSDB, crea la sua tabella d'inoltro, usando se stesso come nodo sorgente.

3.3.7.3 OSPF **OSPF** (*Open Shortest Path First*) è un **protocollo di routing** basato sul *Link-State Algorithm*.

Il protocollo esegue l'algoritmo descritto precedentemente, definendo però un **timer di 30 minuti** al cui scadere viene rieseguito l'algoritmo.

Messaggi OSPF

- *HELLO*, usato dai router per annunciare la propria esistenza;
- *DATABASE DESCRIPTION*, usato in risposta a *HELLO*, contiene l'intero LSDB;
- *LINK-STATE REQUEST*, usato per richiedere informazioni su un determinato collegamento;
- *LINK-STATE UPDATE*, usato principalmente per la costruzione dell'LSBD, o in risposta a *LINK-STATE REQUEST*;
- *LINK-STATE ACK*, usato per il riscontro del *LINK-STATE UPDATE*.

Implementazione OSPF OSPF è implementato a **Livello Applicazione**, sulla **porta 89** ed **utilizza direttamente IP**, senza passare tramite TCP o UDP (il primo non supporta il multicasting, ed il secondo non è affidabile) ma usando un proprio sistema.

3.3.7.4 Distance-Vector Algorithm il Distance-Vector Algorithm ha le due seguenti caratteristiche:

- *Asincrono*, perché ogni nodo non opera necessariamente insieme agli altri;
- *Distribuito*, perché ogni nodo opera con le informazioni ricevute dai vicini, e non con una struttura centralizzata.

Esso è basato su:

- *Vettore delle Distanze*: vettore monodimensionale che contiene i costi minimi per andare da un nodo sorgente a qualsiasi altro nodo raggiungibile nella rete;
- *Equazione di Bellman-Ford*: equazione che viene usata per determinare i cammini minimi:
$$D_x(y) = \min_v\{c(x,v) + D_v(y)\}$$
dove abbiamo:
  - x è il nodo **radice**;
  - v sono tutti i nodi **vicini di x**;
  - y è il nodo **che si vuole raggiungere**;
  - $c(x,v)$ è il **costo per andare da x a v**.

Creazione del Distance Vector il Distance Vector è creato attraverso i seguenti sei passi:

1. il nodo inizializza il Distance Vector;
2. con un messaggio di *HELLO* il nodo contatta i suoi vicini e ne ottiene le distanze, aggiornando il vettore;
3. il nodo invia una copia a tutti i suoi vicini del suo Distance Vector;

4. quando il nodo riceve un vettore da un vicino, allora **applica Bellman-Ford** per aggiornare il suo vettore;
5. se il Distance Vector è cambiato, allora notifica i suoi vicini;
6. ritorna al punto 4 e rimane in attesa.

Modifiche dei Costi un problema che sorge quando viene inviato il Distance Vector è che un nodo non può inviare ad un suo vicino i costi delle rotte che passano attraverso il vicino stesso (ciò porterebbe a problemi con l'aggiornamento del vettore del vicino); di conseguenza ci sono due approcci possibili per risolvere la questione:

- *Split Horizon*, in cui si **cancellano** i costi delle rotte ai nodi che passano attraverso il vicino a cui si sta inviando il vettore;
- *Poisoned Reverse*, in cui i costi delle rotte ai nodi che passano attraverso il vicino vengono **posti uguali a** $+\infty$ (più sicuro perché dà al vicino l'informazione sulla rotta, a differenza di *Split Horizon*).

3.3.7.5 RIP

RIP (*Routing Information Protocol*) è un **protocollo di routing** che si basa sul *Distance-Vector Algorithm*, a differenza di questo però, i nodi inviano la loro **completa tabella di routing**, invece del Distance Vector.

In RIP, **ogni hop costa 1**, e 15 è il costo massimo che può essere assegnato ad un percorso (di fatto, 16 equivale a $+\infty$).

Una particolarità di RIP è che misura misura solo i **costi interreti**, e non al loro interno.

Messaggi RIP in RIP vi sono due tipi di messaggi, usati per costruire le tabelle:

1. *RIP Request*, usato per richiedere l'invio della tabella di routing, a fini diagnostici o di costruzione;
2. *RIP Response*, usato per inviare le tabelle di routing, viene detto anche *Advertisement*, e può essere una risposta ad una *RIP Request* oppure viene inviato allo scadere del *Timer RIP*.

Formato Messaggi

Command (8)	**Version** (8)	Reserved
Entry tabella di routing		

si noti che il numero massimo di entry allegabili ad un messaggio è 25.

Formato Entry

Family	Tag
Network Address	
Subnet Mask	
Next Hop	
Distance	

dove *Family* indica la famiglia del protocollo (per il TCP/IP il valore è 2), mentre *Tag* dà informazioni sul sistema autonomo.

Timer RIP RIP **implementa tre timer** per migliorare la gestione delle informazioni di routing, per l'invio delle tabelle e per l'eliminazione di rotte non più valide:

1. *Timer Periodico* $\sim$ 30 secondi, allo scadere del quale il nodo invia la sua tabella di routing ai suoi vicini;
2. *Timer di Scadenza* $\sim$ 180 secondi, allo scadere del quale, se non si è ricevuto nessun *advertisement* su una rotta per una rete, allora il costo di quella rotta viene impostato a 16 e la rotta è considerata **non valida**;
3. *Timer per Garbage Collection* $\sim$ 120 secondi, settato per ogni rotta non valida: se essa rimane in questo stato per 120 secondi allora viene eliminata dalla tabella.

Caratteristiche di RIP

- *Split Horizon with Poisoned Reverse*: si modifica a 16 il costo di tutte le rotte che passano tramite il vicino a cui si sta inviando la tabella;
- *Triggered Updates*: quando una rotta per una rete è modificata si invia immediatamente l'informazione ai vicini, senza attendere lo scadere del timer;
- *Hold-Down*: quando una rotta diventa non valida, allora tutte le advertisement per quella rotta vengono scartate per un certo periodo.

Implementazione RIP RIP è implementato al **Livello Applicazione** sulla **porta 520** ed usa il **protocollo UDP**; è eseguito con un processo che è chiamato **routed** (*route daemon*) che è sempre in esecuzione.

Internet Routing

I due protocolli di routing visti in precedenza (RIP e OSPF) vengono usati all'interno di un ISP (routing **intra-dominio**), che succede però quando bisogna effettuare il routing fra due o più ISP? Il routing **inter-dominio** si occupa proprio di questo.

3.3.8.1 Autonomous System ogni ISP è considerato come un **AS**, ovvero un *Autonomous System*, e viene **identificato** da un valore univoco di **16 bit**.

Esistono tre tipologie di AS:

1. *AS Stub*: gli AS che sono collegati **ad un solo AS** e **non consentono traffico** attraverso se stessi;
2. *AS Multihomed*: gli AS che sono collegati **a più AS** e **non consentono traffico** attraverso se stessi;
3. *AS di Transito*: gli AS che sono collegati **a più AS** e **consentono traffico** attraverso se stessi.

I vari AS vengono connessi fra di loro tramite dei router che vengono detti *Gateway.*

3.3.8.2 BGP

BGP (*Border Gateway Protocol*) è un protocollo **dedicato al routing inter-dominio**, basato sui ***Path Vector***, ovvero un Distance Vector in cui l'obiettivo primario è la **costruzione di un percorso ottimale in base a determinate priorità decise dagli AS**.

Le destinazioni sono date da **prefissi CIDR**, che rappresentano sottoreti o collezioni di sottoreti.

BGP è un protocollo particolare perché in realtà è **diviso in due sottoprotocolli**: **eBGP** (*External BGP*) e **iBGP** (*Internal BGP*); mentre il secondo è usato da tutti i router, il **primo è usato solo dai router Gateway**.

eBGP si stabilisce una **connessione fra due router Gateway**, che si scambiano le informazioni su come **instradare i pacchetti all'AS vicine**.

iBGP all'interno di un AS, si stabilisce una **connessione fra ogni possibile coppia di router**, in modo da poter scambiare le informazioni su come raggiungere le altre AS (ottenute dai router Gateway con l'eBGP).

eBGP ed iBGP con la combinazione di eBGP ed iBGP si costruiscono infine le **Tabelle di Percorso** verso le altre AS; queste nuove tabelle vengono aggiunte alle tabelle di routing dei router Gateway e alle tabelle degli altri router vengono aggiunte linee che passano tramite i router Gateway per andare verso le altre AS.

- negli *AS Stub*, il router Gateway aggiunge **1 linea** per l'unico altro Gateway a cui sono collegati;
- negli *AS di Transito*, i router Gateway aggiungono **tutte le linee** per **tutti i nodi delle altre AS** ed imposta il costo per raggiungerle (la metrica **dipende dal protocollo intra-dominio usato**).

Attributi del Percorso i percorsi in BGP vengono annunciati insieme a dei valori che sono detto *attributi* e danno ulteriori informazioni sul percorso; di seguito sono elencati i tre attributi **obbligatori** che vengono sempre inviati con un percorso:

1. *ORIGIN*: descrive l'origine dell'aggiornamento, può essere uno di tre valori:
   - **IGP**, se viene da un protocollo interno all'AS;
   - **EGP**, se viene da un protocollo esterno all'AS;
   - **Incomplete**, se l'informazione dell'origine non è disponibile.
2. *AS-PATH*: elenca tutti gli AS attraverso cui l'aggiornamento è passato;
3. *NEXT-HOP*: dà l'indirizzo IP della prossima interfaccia attraverso cui passare per raggiungere la destinazione dell'aggiornamento.

Il prefisso di rete, unito agli attributi, annunciano ciò che viene detto *rotta*: ogni Gateway quando riceve un annuncio di una rotta decide, **in base alle sue politiche di importazione**, se accettarla o meno.

Messaggi BGP

- *OPEN*, usato per aprire nuove connessioni;
- *UPDATE*, usato per annunciare le nuove rotte;
- *KEEPALIVE*, usato per mantenere una connessione attiva;
- *NOTIFICATION*, usato per segnalare errori nel messaggio precedente o annunciare la chiusura della connessione.

Implementazione BGP viene implementato al **Livello Applicazione** sulla **porta 179** ed utilizza il **protocollo TCP**.

Routing Multicast

In precedenza abbiamo visto come vengono effettuati i routing *Unicast* (IP sorgente → IP destinatario) e *Broadcast* (IP sorgente → IP broadcast);

vediamo adesso come è stato approcciato il routing ***Multicast*** (IP sorgente → **IP gruppo multicast**).

Il routing **1 a N** può essere di due tipologie diverse:

- *Multicast*: la sorgente invia **un solo messaggio**, che viene duplicato quando serve;
- *Unicast Multiplo*: la sorgente invia **n messaggi**, risulta però inefficiente.

3.3.9.1 Indirizzamento Multicast per poter inviare i messaggi in multicast bisogna poter indirizzare **contemporaneamente** tutti gli host; per permettere ciò, l'indirizzamento IPv4 prevede una **serie di indirizzi** dedicati esclusivamente a questo compito e che **vanno da 224.0.0.0 a 239.255.255.255**.

3.3.9.2 IGMP **IGMP** (*Internet Group Management Protocol*) è il protocollo **dedicato alla trasmissione di messaggi in multicast**, esso ha il compito di **coordinare i router *Multicast*** ed usa il **protocollo IP**.

Messaggi IGMP

- *MEMBERSHIP QUERY*, router → host, usato per chiedere i gruppi di appartenenza;
- *MEMBERSHIP REPORT*, host → router, risposta alla *MEMBERSHIP QUERY*;
- *MEMBERSHIP LEAVE*, usato da un host per lasciare un gruppo.

Riguardo alle *Membership*, ognuna di esse ha un **timer** ed ogni router mantiene una lista delle Membership degli host delle reti e sottoreti a cui è connesso.

Costruzione del Sottoalbero di Instradamento nel momento in cui viene inviato un messaggio in multicast, esso attraversa ciò che viene detto *Albero di Instradamento*, per poter raggiungere tutti gli host del gruppo; la costruzione di tale albero può avvenire in due modi:

1. *Condiviso dal Gruppo*, se c'è un albero **unico** per il **gruppo**, con un nodo che viene detto "centro" e che serve ad inviare i messaggi;

2. *Basato sull'Origine*, se c'è **un** albero per **ogni nodo** del gruppo, che è radice del suo albero.

Livello Collegamento

Il Livello Collegamento ha il compito di garantire **la comunicazione *hop-to-hop***, quindi il trasferimento di pacchetti dati, che sono detti "**frame**", da un host all'altro.

I protocolli del Livello Collegamento devono essere particolarmente **versatili**, in quanto devono poter essere usati da tutte le tecnologie hardware che si connettono ad una rete; deve inoltre gestire collegamenti **punto-punto** oppure **broadcast**.

Il Livello Collegamento è diviso in **due sottolivelli**:

- *Data-Link Control* (**DLC**): si occupa del **trasferimento dei frame**;
- *Media Access Control* (**MAC**): si occupa di gestire la **comunicazione broadcast**.

Notazione

- host e router → nodi/stazioni;
- collegamento/link → canale di comunicazione;
- R = rate bit/s di un link.

Servizi Offerti

- *Framing*: ovvero l'incapsulamento ed il decapsulamento dei dati;
- *Consegna Affidabile*;
- *Indirizzamento degli host*: viene utilizzato l'indirizzamento MAC;
- *Controllo di flusso e di errori.*

3.4.2.1 Controllo Errori gli errori nel trasferimento di frame a questo livello sono **generalmente dovuti ad interferenze** che possono influenzare o singoli bit (**più raramente**) od un gruppo (in questo caso è detto "*burst*").

Per poter verificare l'integrità dei frame, vengono aggiunti dei bit di controllo (si usano bit di **parità unidimensionale o bidimensionale** solitamente) che sono detti **EDC** (*Error Detection and Control*).

Protocolli di Accesso Multiplo

Hanno il compito di **regolare la trasmissione e la ricezione di frame** e sono divisi in tre categorie:

1. Protocolli *a Suddivisione del Canale*;
2. Protocolli *ad Accesso Casuale*;
3. Protocolli *a Rotazione.*

3.4.3.1 Protocolli a Suddivisione del Canale

- **TDMA** (*Time Division Multiple Access*): suddivide il canale in **intervalli temporali**: siano N gli intervalli, allora per ogni nodo il **tasso trasmissivo** è pari a $\frac{R}{N}$ bit/s;
- **FDMA** (*Frequence Division Multiple Access*): suddivide il canale in **fasce di frequenza** e ne **assegna una ad ogni stazione**;
- **CDMA** (*Code Division Multiple Access*): verrà approfondito in seguito nell'apposita sezione.

3.4.3.2 Protocolli ad Accesso Casuale l'accesso è detto casuale perché **non programmato**: quando un nodo deve **inviare** un frame allora **accede al canale**. Il problema maggiore di questi protocolli è la **collisione dei frame**: essi devono quindi permettere l'individuazione di collisioni e la conseguente ritrasmissione dei frame.

ALOHA Puro **ALOHA Puro** fa trasmettere ai nodi **quando vogliono** e prevede la **trasmissione di ACK**, con un relativo timer di ricezione ACK dal lato mittente.

Se si rileva una collisione, allora si attende un tempo casuale detto di "*back-off*" al termine del quale vengono ritrasmessi i dati; se dopo k_{MAX} (che è pari a 15) tentativi la trasmissione non ha buon fine allora viene interrotta.

- *Timeout* $= 2T_p$, dove $T_p = RTT$ dei due nodi più lontani;
- *Back-Off* $= rand * T_{fr}$, dove $rand \in [0, 2^{k-1}]$ e T_{fr} è il tempo per inviare un frame;

Il **Tempo di Vulnerabilità**, ovvero il tempo in cui può verificarsi una collisione durante la trasmissione, in ALOHA Puro è pari a $2T_{fr}$, in quanto un frame inviato a t_0 collide se c'è un altro frame inviato fra $[t_0 - T_{fr}, t_0 + T_{fr}]$.

Di conseguenza, l'**efficienza** di ALOHA Puro, ovvero la **% di banda usata con successo**, è calcolata nel seguente modo (si noti che la p sta per "*probabilità di*"):
*p(il nodo trasmette) * p(no trasm. in* $[t_0 - T_{fr}, t_0]$*) * p(no trasm. in* $[t_0, t_0 + T_{fr}]$*)*

$$p(1-p)^{N-1}(1-p)^{N-1} = p(1-p)^{2(N-1)} \sim \frac{1}{2e} = 0,18$$

L'efficienza di ALOHA Puro è quindi pari al **18%**.

Slotted ALOHA **Slotted ALOHA** prevede la **divisione del tempo in intervalli lunghi un** T_{fr}, ponendo le restrizioni di poter inviare frame con una **dimensione fissata e solo all'inizio di uno slot**.

Il **Tempo di Vulnerabilità** in questo caso è pari ad **uno slot**, quindi un T_{fr}, mentre l'**efficienza** è pari a:

$$p(1-p)^{N-1} \sim \frac{1}{e} = 0,37$$

ovvero al **37%**.

CSMA **CSMA** (*Carrier Sense Multiple Acces*) prevede l'**ascolto del canale**: se il **canale è libero** allora il nodo può trasmettere, altrimenti, in base al tipo di **persistenza**, avrà un certo comportamento.

In questo caso il **Tempo di Vulnerabilità** è pari al **ritardo di propagazione** T_p perché due nodi possono iniziare a trasmettere prima che il frame di uno arrivi all'altro, ipotizzando che il canale sia libero.

Come detto in precedenza, il CSMA può avere diversi tipi di persistenza, elencati di seguito:

- *Non Persistente*: **attende un tempo casuale** prima di rimettersi in ascolto se rileva il **canale occupato**, altrimenti trasmette subito;
- *p-Persistente*: rimane **in ascolto** se rileva il canale occupato, altrimenti **trasmette con probabilità p**;
- *1-Persistente*: come il *p Persistente*, ma **p è uguale a 1** e quindi trasmette appena rileva il canale libero.

CSMA/CD **CSMA/CD** (*Collision Detection*) prevede, oltre all'ascolto prima della trasmissione, l'ascolto del canale **anche durante la trasmissione**: appena viene rilevata la collisione, allora si ferma la trasmissione.

Poiché il nodo deve rimanere in ascolto durante **tutta la trasmissione** del frame, allora tale tempo deve essere **maggiore o uguale a due volte il ritardo di propagazione** (due volte per poter rimanere in ascolto per frame che sono trasmessi prima o dopo quello inviato dal nodo); di conseguenza la **dimensione minima** del frame è pari al prodotto $R2T_p$.

L'**efficienza** del CSMA/CD è pari al **50%**, se *1-Persistente*.

3.4.3.3 Protocolli a Rotazione

i Protocolli a Rotazione realizzano un compromesso fra i due tipi di protocolli descritti precedentemente.

Polling protocollo **centralizzato**, in cui un nodo detto ***poll*** **gestisce la trasmissione sul canale**, ordinando a quale nodo trasmettere ed eliminando le collisioni e gli slot vuoti dello Slotted Aloha; gli unici svantaggi sono il ritardo derivato dal polling e la vulnerabilità a guasti del poll.

Token-Passing protocollo **decentralizzato**, in cui i nodi si passano un **token che consente la trasmissione**; l'unico problematica si genera nel momento in cui si guasta il nodo col token: a quel punto l'intero canale è bloccato.

Indirizzamento MAC

L'indirizzamento MAC prevede l'utilizzo di **48 bit** per indirizzo, permettendo di identificare **univocamente ogni interfaccia**.

Il valore dell'indirizzo è in **esadecimale**, quindi ad ogni byte (in totale ogni indirizzo ne ha 6) corrispondono 2 cifre esadecimali.

L'indirizzo MAC di **broadcast** è uguale a quello usato in IP, il suo valore in esadecimale è infatti FF-FF-FF-FF-FF-FF.

Poiché il **Livello Collegamento usa l'indirizzamento MAC per gli host**, l'header è formato dall'indirizzo MAC della prossima interfaccia e quello dell'interfaccia attuale:

MAC Interfaccia seguente | **MAC Interfaccia attuale**

ARP

ARP (*Address Resolution Protocol*) è il protocollo dedicato a **associare gli indirizzi IP agli indirizzi MAC**. Viene implementato a **Livello Rete**, e costruisce una **tabella per mantenere le associazioni**, il cui record generico è formato nel seguente modo:

< **Indirizzo IP** ; **Indirizzo MAC** ; TTL >

3.4.5.1 Funzionamento un nodo, per ottenere un indirizzo MAC da un indirizzo IP invia una **richiesta ARP in broadcast**, a cui soltanto il **nodo di cui si chiede il MAC risponde**.

Ogni messaggio ARP contiene le seguenti informazioni:

- il **protocollo del Livello Rete e Collegamento**, due sequenze di 16 bit;
- le **lunghezze** degli indirizzi del **Protocollo** (Liv. Rete) e dell'**Hardware** (Liv. Collegamento), due sequenze di 8 bit;
- l'**operazione** (o *Richiesta* o *Risposta*), lunga 16 bit;
- gli **indirizzi** del **mittente** e del **destinatario** (sia MAC che IP).

Ethernet

Ethernet è uno **standard di interconnessione** creato per poter connettere dispositivi di produttori diversi. Come standard, deve definire le **funzionalità** del **Livello Fisico** e del **Livello Collegamento** nelle LAN.

La documentazione di Ethernet si trova nel documento **IEEE 802**.

3.4.6.1 Ethernet Standard fu il primo standard di Ethernet e supportava LAN con un **bit rate di 10 Mbps** (*MegaBit per secondo*).

Formato Messaggio i messaggi di Ethernet Standard sono strutturati nel seguente modo:

- l'**Header del Livello Fisico**, che contiene:
  - il **Preambolo**, formato da 7 byte tutti uguali (10101010) e usato per **sincronizzare le interfacce**;
  - l'**SFD** (*Starting Frame Delimiter*), formato da 1 byte (10101011) e che **segnala l'inizio del frame**;
- il **Frame**, che contiene sia l'**Header del Livello Collegamento** che i **Dati**:
  - l'**indirizzo del Destinatario**, formato da 6 byte;
  - l'**indirizzo del Mittente**, formato da 6 byte (si noti che sono gli indirizzi MAC);
  - il **Tipo**, formato da 2 byte che **indica il protocollo del Livello Rete**;
  - i **Dati** ed il **padding**, che possono andare da un **minimo di 46 byte** ad un **massimo di 1500 byte** (il padding, ovvero una sequenza di bit uguali a 0, è usato solo nel caso in cui i dati non arrivino al minimo di 46 byte);
  - il **CRC**, formato da 4 byte, per il **controllo degli errori**.

I motivi per cui i Dati debbano andare da un minimo di 46 byte (creando un frame lungo **64 byte**) ad un massimo di 1500 (creando un frame lungo **1518 byte**) sono due:

1. **minimo 46**, perché poiché Ethernet Standard **utilizza il CSMA/CD** e prevede LAN in cui la **distanza massima** è di **2500 m**, allora il frame deve essere lungo almeno 64 byte per far funzionare bene il CSMA/CD;
2. **massimo 1500**, per evitare che non ci siano messaggi che **monopolizzino il canale**.

Caratteristiche

- **Connectionless**;
- **Non affidabile**;
- **CSMA/CD**, con la seguente caratterizzazione:
  1. **1-Persistente**;
  2. quando viene **rilevata una collisione**, si interrompe la trasmissione e viene **inviato un segnale di disturbo lungo 48 bit**, detto "*Jam*";
  3. ha un **Backoff Esponenziale**, e si aspetta un tempo pari a kT_{512bit} (ovvero il tempo per trasmettere un frame lungo 64 byte) dove k si sceglie casualmente fra i valori $\{0, 1, 2, \ldots, 2^{m-1}\}$ in cui m = $\min\{n, 10\}$ (n è il numero di collisioni consecutive).

Topologie Ethernet Standard prevede due possibili topologie per le LAN:

1. *a Bus*;
2. *a Stella con Hub* (l'Hub è un ripetitore broadcast).

3.4.6.2 Fast Ethernet creato per supportare le LAN con bit rate **fino a 100 Mbps**, per poter continuare ad usare il CSMA/CD con frame da 64 byte furono ideate due soluzioni: si pensò o di **diminuire** la distanza **massima** a **250 m** oppure di **usare** uno **Switch di Collegamento** con una topologia a stella, senza bisogno di usare il CSMA/CD.

Switch di Collegamento è un dispositivo che permette lo **smistamento dei pacchetti in maniera selettiva**, eliminando quindi le collisioni e la necessità di utilizzare il CSMA/CD.

Per **apprendere gli indirizzi** utilizza un sistema di **auto apprendimento**, costruendo la tabella mano a mano che gli vengono inviati messaggi oppure con **un flooding di richeste ARP**.

Di seguito sono elencate le proprietà dello Switch:

- elimina le collisioni;
- può connettere reti con tecnologie diverse;
- aggiunge un livello di sicurezza;
- non deve essere inizializzato manualmente;
- risulta trasparente ai nodi.

3.4.6.3 Gigabit Ethernet versione successiva di Fast Ethernit, supporta reti LAN con bit rate fino a **1000 Mbps**, ovvero **1 Gbps** (da qui il nome); prevede una topologia **a stella con Switch**.

VLAN

Una **VLAN** (o *LAN Virtuale*) è una **configurazione di una LAN mediante software**, ovvero una **divisione in gruppi logici** dei nodi.

La creazione di VLAN è possibile grazie all'**utilizzo di Switch**, in cui è possibile assegnare alle VLAN delle porte dedicate attraverso cui inviare i messaggi destinati a nodi di quella rete.

Reti Punto-Punto

Le reti Punto-Punto prevedono **collegamenti dedicati fra 2 nodi** ed usano il **PPP** (*Point-to-Point Protocol*).

3.4.8.1 PPP

PPP è un protocollo molto semplice, in quanto **non prevede indirizzamento** (il nodo che riceve è quello che non trasmette) e si deve occupare soltanto del **framing** e della **rilevazione e correzione degli errori**.

Formato Messaggio

- **Flag**, **all'inizio e alla fine del messaggio** (serve a **delimitarlo**), è sempre 01111110;
- **Address**, sempre 11111111;
- **Control**, sempre 00000011;
- **Protocol**, che descrive il protocollo del Livello Rete ed è formato da 1 o 2 byte;
- **Dati**;
- **Checksum**, 2 o 4 byte.

LAN Wireless, Bluetooth e CDMA

LAN Wireless

Le LAN Wireless permettono la connessione di dispositivi senza dover usare mezzi cablati, ma usando come mezzo trasmissivo l'**aria**; queste reti sono poi collegate alle altre reti tramite degli **access point cablati** (ma solo in Reti *con Infrastruttura*).

Poiché vengono inviati segnali attraverso l'aria, queste reti sono soggette a varie problematiche:

- **Attenuazione del Segnale**;
- **Riflessioni dei segnali su Ostacoli**;
- **Interferenza**.

Per quantificare gli **errori nei trasferimenti dei dati** si usa il **SNR** (*Signal to Noise Ratio*), un rapporto dato dalla seguente formula:

$$\frac{forzaSegnale}{forzaInterferenza}$$

Inoltre, una rete LAN Wireless può appartenere ad una di due tipologie:

1. Rete *con Infrastruttura*, che usano gli **AP** (*Access Point*);
2. Rete *ad Hoc*, in cui **gli host si organizzano da soli**.

4.1.0.1 Architettura Reti le reti Wireless possono avere due tipi di architetture:

1. **BSS** (*Basic Service Set*): una singola rete (sia *con Infrastruttura* che *ad Hoc*, anche se è più diffusa la prima categoria);
2. **ESS** (*Extended Service Set*): due o più *BSS con Infrastruttura*, **collegati** tramite **Access Point**.

Canali e Associazioni

Lo **spettro delle frequenze Wireless** è diviso in **11 fasce**, che vengono scandite dai dispositivi per cercare segnali.

L'associazione di un dispositivo ad una rete Wireless è una procedura che si basa proprio su questa divisione delle frequenze e sulla potenza dei segnali inviati dagli AP:

1. gli AP inviano periodicamente segnali con il loro **SSID** (*Service Set ID*) e indirizzo MAC;
2. l'host scandisce le 11 fasce;
3. viene scelto l'AP col **segnale più forte**;
4. l'AP a questo punto accetta la richiesta d'accesso (in alcuni casi serve un'autenticazione) e permette all'host di fare richieste DHCP.

Accesso al Mezzo

Nelle reti Wireless per l'accesso al mezzo esistono due tecniche diverse:

1. **DCF** (*Distributed Coordination Function*), ovvero una contesa non regolata (verrà approfondita in seguito);
2. **PCF** (*Point Coordination Function*), ovvero un accesso regolato dagli AP e quindi senza contesa.

DCF

Poiché il mezzo trasmissivo è condiviso, i nodi **competono per poterlo usare**.

Per come sono strutturate le reti Wireless, è impossibile **fare Collision Detection** per vari motivi:

- Sarebbe troppo costoso dotare i dispositivi di adattatori che gli consentano di captare **segnali molto più deboli** di quelli che vengono trasmessi dal dispositivo stesso;
- *Problema dell'Hidden Terminal*, ovvero la possibilità che un host non possa accorgersi che un altro stia trasmettendo:
  - per l'**attenuazione del segnale** (i due host sono troppo lontani);
  - per la **presenza di un ostacolo**, che può bloccare il passaggio dei segnali da un host all'altro.

Per questo motivo le reti Wireless utilizzano il **CSMA/CA**.

CSMA/CA

Il **CSMA/CA** (*Collision Avoidance*) è pensato per **evitare collisioni ascoltando il canale prima di trasmettere**.

4.1.4.1 Funzionamento

1. Ascolto **persistente** del canale;
2. Canale **libero** → **attesa DIFS** (*DFS Interframe Space*);
3. Canale **ancora libero** → **attesa tempo Back-Off**, scelto a **caso con la Contention Window** (la dimensione della *CW* raddoppia per ogni messaggio inviato, fino ad un limite dopo il quale viene chiusa la connessione);
4. Invia l'**RTS** (*Request To Send*) **al destinatario**;
5. Il **destinatario** invia a **tutti** gli host una **CTS** (*Clear To Send*), dopo aver **atteso un SIFS** (*Short Interframe Space*): in pratica **informa tutti che il canale è occupato dal mittente**;
6. Il mittente **attende un SIFS**, dopo il quale comincia la **trasmissione** e **setta il timer per l'ACK**;
7. Il destinatario, **ricevuto il messaggio**, **attende un SIFS** dopo il quale invia l'ACK.

4.1.4.2 NAV ogni host nell'RTS **include il tempo per cui occuperà il canale**: gli host che vengono raggiunti dal CTS usano il **NAV** (*Network Allocation Vector*) per mantenere quest'informazione.

Formato Messaggio ogni frame inviato contiene i seguenti dati:

- **FC** (*Frame Control*), 2 byte di informazioni sul frame, i cui dati più importanti sono:
  - **Type**, 2 bit che indicano di che tipo è il frame:
    - *Gestione* (00);
    - *Controllo* (01);
    - *Dati* (10);
  - **SubType**, 4 bit che vengono usati solo se il **Type** è *Controllo*:
    - *RTS* (1011);
    - *CTS* (1100);
    - *ACK* (1101);
  - **To DS** e **From DS**, 2 bit che danno informazioni sull'**indirizzamento**:
    - **To DS = 0**, **From DS = 0**: da **Host** a **Host**
      1. Address 1: destinatario;
      2. Address 2: mittente;
      3. Address 3: BSS ID;
      4. Address 4: N/A;
    - **To DS = 0**, **From DS = 1**: da **AP** a **Host**
      1. Address 1: destinatario;
      2. Address 2: AP mittente;
      3. Address 3: mittente;
      4. Address 4: N/A;
    - **To DS = 1**, **From DS = 0**: da **Host** a **AP**
      1. Address 1: AP destinatario;
      2. Address 2: mittente;
      3. Address 3: destinatario;
      4. Address 4: N/A;
    - **To DS = 1**, **From DS = 1**: da **AP** a **AP**
      1. Address 1: AP destinatario;
      2. Address 2: AP mittente;
      3. Address 3: destinatario;
      4. Address 4: mittente;
- **Durata trasmissione**, 2 byte;
- **Tre Indirizzi**, ognuno da 6 byte (**indirizzi MAC**);

- **SC** (*Sequence Control*), 2 byte:
  1. # **Frammento**, 4 bit;
  2. # **Sequenza**, 12 bit;
- **Un Indirizzo**, 6 byte;
- **Frame Body**, da 0 a 2312 byte;
- **FCS** (*Frame Check Sequence*), un **CRC** da 4 byte.

Bluetooth

Una **rete Bluetooth** consiste in una connessione di **dispositivi con funzionalità diverse**, in una **rete Wireless *ad Hoc*** e con un **massimo di 8 host**.

Le reti Bluetooth ricadono nelle **PAN** ed ha un raggio di $\sim$ 10 m.

Struttura

Le reti possono essere di due tipi:

1. *Piconet*, una singola rete in cui **una stazione** è detta **primaria** mentre le **altre secondarie**;
2. *Scatternet*, un **insieme** di *Piconet*.

Accesso al Mezzo Trasmissivo

Viene usata una **variante** del TDMA: il **TDD-TDMA** (*Time Division Duplexing TDMA*); questa variante prevede una comunicazione **half-duplex**, con **slot temporali** da **625** μ**s**.

La trasmissione è half-duplex perché negli slot **pari trasmette la primaria**, mentre negli slot **dispari le secondarie**, che trasmettono **singolarmente dopo esser scelte dalla primaria**; inoltre la trasmissione può durare **al più 366** μ**s**, poiché il tempo restante è impiegato per effettuare il **salto di frequenza**.

CDMA

Il **CDMA** (*Code Division Multiple Access*) è un protocollo ad **Accesso Casuale** che permette la **trasmissione contemporanea di tutte le stazioni** sfruttando una particolare **codifica del segnale**.

Per effettuare questa codifica, ad **ogni stazione** è assegnato un **codice** con le seguenti **proprietà**:

- se **moltiplicato per se stesso**, dà il **numero di stazioni**;
- se **moltiplicato per un altro codice**, dà **0**.

Codifica del Segnale

I bit trasmessi vengono codificati con i seguenti valori:

- 0: -1;
- 1: +1;
- *Stazione Muta*: 0.

A questo punto, ogni segnale viene **moltiplicato per il codice della sua stazione**, e poi vengono **tutti sommati**.

Di conseguenza, in base anche alle proprietà descritte precedentemente, per ottenere i dati trasmessi da una generica stazione D_i basta moltiplicare tale somma per il codice della stazione e dividerlo poi per il numero di stazioni:

$$\frac{(d_1c_1 + \cdots + d_ic_i + \cdots + d_Nc_N)c_i}{N} =$$

$$\frac{d_1c_1c_i + \cdots + d_ic_ic_i + \cdots + d_Nc_Nc_i}{N} =$$

$$\frac{d_10 + \cdots + d_iN + \cdots + d_N0}{N} =$$

$$\frac{d_iN}{N} =$$

$$d_i$$

Generazione dei Codici

Per generare i codici da assegnare alle stazioni si usano le ***Tabelle di Walsh***, ovvero delle matrici quadrate costruite nel seguente modo:

$$W_1 = \begin{bmatrix} \pm 1 \end{bmatrix}$$

(con una sola stazione si può scegliere o +1 o -1 come codice.)

$$W_{2i} = \begin{bmatrix} W_i & W_i \\ W_i & \overline{W_i} \end{bmatrix}$$

A questo punto, all'*i-esima* stazione verrà assegnato come codice l'*i-esima riga* della matrice.