Reti di Calcolatori

Parte 0: Ei fu

→ Le domande sono in fondo al file ←

Questo file è stato scritto come riassunto personale per imparare le cose assimilate dal libro e durante le lezioni.
Molte informazioni sono ridondanti, altre potrebbero mancare, in generale non è stata posta alcuna cura per rendere questo contenuto assimilabile da un altro essere umano.
Se state ancora leggendo e avete la pervicacia di volerli utilizzare per studiare, non sarò io a fermarvi, ma lasciate ogni speranza voi ch'entrate.
Come consiglio personale, date un'occhiata almeno alle slide del professore per capire cosa è superfluo o cosa va integrato.

Buona fortuna, ve ne servirà.

Parte 1: Introduzione

Tipologie di rete, dalla PAN alla WAN:

1) Le **PAN (Personal Area Network)** sono tipologie di rete che permettono la comunicazione a distanza di persona, come lo possono essere delle cuffie wireless o la connessione presente tra uno smartwatch e il telefono. Le PAN possono essere costruite su diverse tecnologie a raggio corto, come per esempio il Bluetooth, in modo da semplificare la comunicazione tra dispositivi vicini.

2) Le **LAN (Local Area Network)** sono tipologie di rete che agiscono a distanza di una abitazione o edificio. Le LAN consentono il collegamento tra più dispositivi attraverso l'utilizzo di un AP (Access Point), per esempio il Wireless Router, che gestisce la comunicazione tra i dispositivi della LAN e anche tra essi e l'Internet.

La LAN può essere anche cablata, la topologia più comune è l'Ethernet, in particolare nello switched Ethernet ogni dispositivo si connette al protocollo Ethernet attraverso un dispositivo detto switch, con il compito di trasferire i pacchetti fra i computer ad esso collegati, utilizzando l'indirizzo di ogni pacchetto per decidere dove trasferirlo. E' possibile suddividere una LAN in più piccole LAN logiche, dette VLAN (Virtual LAN), dove il router suddivide logicamente la connessione tra le VLAN, questo può essere utile per esempio nelle aziende dove ogni dipartimento ha una propria VLAN.

3) Le **MAN (Metropolitan Area Network)** sono tipologie di rete che coprono intere città, la MAN più conosciuta è la rete cablata televisiva, inizialmente utilizzata solo per la TV, col tempo è stata utilizzata anche per provvedere al collegamento Internet, dove entrambi i segnali arrivano a un cable head-end centralizzato.

4) Le **WAN (Wide Area Network)** si basano su ampie aree geografiche e collegano delle macchine dette hosts, il collegamento tra gli host è detto communication subnet o semplicemente subnet (n.d.r. da non confondere con la subnet degli indirizzi). Nelle WAN, la subnet consiste in diverse linee di trasmissione che hanno il compito di muovere i bit tra le macchine. Tali linee di trasmissione sono poi connesse attraverso dei device di switch, detti comunemente router. Se due router non sono connessi direttamente è necessario decidere quale strada utilizzare attraverso un algoritmo di routing. Il modo in cui il router decide dove mandare i pacchetti è invece detto algoritmo di forwarding.

5) L'**Internetwork** è l'unione di network in un'unica grande rete: esistono moltissimi network diversi nel mondo, operati da hardware e software differenti, diventa quindi in qualche modo necessario permettere la connessione tra di essi, per internetwork o semplicemente internet, si intende tale unione, l'Internet globale è uno specifico internet che ricopre tutto il mondo. Il dispositivo che permette la comunicazione tra due o più reti è il gateway, che può essere per esempio un router.

Topologie di rete

Esistono due macro tecnologie di comunicazione:

- **Reti punto a punto**, dove ogni calcolatore deve connettersi direttamente ad un altro calcolatore, se due calcolatori non sono direttamente collegati è necessario creare un instradamento attraverso altri calcolatori. Questa topologia può essere costosa per via dell'alto numero di connessioni dedicate.
- **Reti broadcast**, dove tutti i calcolatori sono connessi tramite un unico canale di comunicazione condiviso, i messaggi sono ricevuti da tutti ma letti solamente dal destinatario.

In ogni caso, non tutti i calcolatori possono trasmettere simultaneamente, quindi diviene necessario un sistema di regole o supervisione. Le regole possono essere statiche, ovvero prefissare in anticipo assegnando per esempio uno slot temporale tramite round-robin, oppure dinamiche, ovvero decide di volta in volta che inizia la trasmissione.
Le trasmissioni statiche sono inefficienti, mentre le trasmissioni dinamiche possono portare a delle collisioni.
Le reti possono essere di diverse **topologie**:

- In una **rete a bus** le informazioni viaggiano su un unico canale, e tutti i nodi possono leggere le informazioni in viaggio. I nodi non destinatari semplicemente scartano i pacchetti non destinati a loro. Il vantaggio principale è che è semplice da realizzare ed estendere, ma può essere più lenta.
- Una **rete ad anello** fa viaggiare le informazioni in unico canale, formando una sorta di catena dove viaggiano i pacchetti. La direzione può essere unica o bidirezionale. In questa rete ogni nodo non destinatario inoltra il pacchetto al successivo fino a quando non raggiunge il destintario. Il vantaggio principale è la velocità e la facilità di estensione, ma è poco tolleranete ai guasti.
- La **rete a stella** è invece costruita a partire da un nodo centrale che gestisce la comunicazione, il nodo centrale indirizza il pacchetto verso il destinatario. Questa topologia è semplice da realizzare e abbastanza efficiente, mentre la tolleranza ai guasti è concentrata nel nodo centrale.
- In una **rete a maglia** le informazioni viaggiano su un unico canale, e tutti i nodi possono leggere le informazioni in viaggio. Un nodo non destinatario scarta un pacchetto ricevuto. Questa topologia è semplice da realizzare ed estendere ma può avere velocità ridotte.

Protocolli di rete
L'eterogeneità della rete ha portato alla necessità di stabilire delle regole per poter comunicare, queste regole prendono il nome di **protocollo:**
"Insieme coordinato di regole che permettono a due interlocutori di scambiarsi rapidamente e univocamente dati e messaggi"
La rete è organizzata a livelli o strati, con lo scopo di semplificarne la progettazione. Ogni livello *n* interagisce con il livello *n-1* oppure *n+1*, e introduce un certo grado di astrazione. In questo modo, non è necessario per gli altri livelli conoscere i dettagli implementativi dei vicini, ma solo l'interfaccia che essi mettono a disposizione. Inoltre ognuno di essi interagisce con il suo corrispettivo sull'host mittente/destinatario, attraverso un protocollo di livello *n*.
La progettazione dei protocolli di rete deve far fronte ad alcuni problemi comuni ad ogni livello:

- L'**affidabilità**, ovvero la possibilità per un rete di operare correttamente nonostante essa sia composta da elementi singolarmente non affidabili (basta pensare al fatto che i bit che attraversano la rete devono viaggiare per mezzi fisici facilmente soggetti a interferenze), vengono introdotti quindi meccanismi per **rilevamento degli errori** oppure di **correzione degli errori.** Vi è inoltre la necessità di instradare la connessione attraverso la via migliore, anche nel caso alcuni collegamenti o router in quel momento non siano funzionanti, e la rete deve prendere automaticamente questa decisione: tale meccanismo prende il nome di **routing**.
- **Allocazione delle risorse,** ovvero la necessità di allocare dinamicamente la banda a seconda della necessità degli host, oppure evitare che un mittente molto veloce riempia di dati un destinatario che non è in grado di processarli altrettanto velocemente: quando ciò accade si ha un sovraccarico della rete che prende il nome di **congestione.**
- **Evolubilità**, ovvero la possibilità per la rete di evolversi e ingrandirsi in base alle necessità. Il meccanismo dei livelli è ciò che permette ad ogni strato di modificare i dettagli implementativi in maniera indipendente dagli altri.
- **Sicurezza,** ovvero la necessità di difendere la rete da diversi tipi di minacce, come per esempio l'intercettazione dei dati. Ogni livello implementa diversi meccanismi come **l'autenticazione**, la **confidenzialità** e altri sistemi per garantire **l'integrità** dei dati.

In realtà, ogni livello non comunica direttamente con il corrispettivo sull'altro host, ma il messaggio da inviare parte da livello più alto e scende fino ad arrivare sul mezzo fisico, viaggia attraverso di esso, e raggiunge il destinatario, dove risale i livelli da quello più basso fino a quello più alto.

Ogni livello può offrire due tipi di servizi ai livelli successivi: **connection-oriented** e **connectionless**, ed entrambi possono essere **reliable** o **unreliable**:
Un servizio connection-oriented implica la creazione di una connessione, ovvero un canale di comunicazione dove i messaggi vengono inviati, un servizio connectionless invece invia i messaggi associandovi l'indirizzo di destinazione simile al funzionamento delle poste.
Un servizio reliable assicura che tutti i messaggi inviati vengano ricevuti, ciò avviene tipicamente attraverso il meccanismo dell'acknowledgement, ovvero la conferma della ricezione del messaggio, mentre un servizio unreliable invia i messaggi senza assicurarsi che arrivino a destinazione.
Possiamo quindi avere generalmente quattro tipi di servizi, la maggiore affidabilità e la creazione di una connessione garantisce che i dati non vadano persi ma introduce overhead e ritardi.

Un servizio è formalmente definito da un insieme di primitive, ovvero di operazioni fondamentali disponibili ai processi per usufruire del servizio. (Per esempio, alcune primitive potrebbero essere le operazioni di LISTEN, ACCEPT, SEND, RECEIVE, etc.. per stabilire una connessione)
In generale però, il concetto di servizio **è distinto** da quello di protocollo: un servizio è l'insieme di primitive che un livello offre al livello superiore, mentre un protocollo è l'insieme delle regole che governano la comunicazione tra *peer* di uno stesso livello. Le entità implementano i servizi utilizzando un protocollo, ma sono liberi di cambiare tale protocollo a patto che il servizio visibile agli utenti rimanga lo stesso, in maniera simile a quanto accade nella programmazione a oggetti con l'incapsulamento.

Modelli ISO/OSI e TCP/IP
I modelli ISO/OSI e TCP/IP sono due modelli di riferimento che hanno lo scopo di definire le funzionalità e caratteristiche dei livelli di astrazione.
Il modello ISO/OSI (International Standards Organization - Open Systems Interconnection) è un modello basato su sette strati: il livello applicazione, presentazione, sessione, trasporto, network, data link e fisico. E' un modello rigoroso, ovvero viene tracciata la distinzione tra protocolli, servizi ed interfacce, ma complesso e di non facile implementazione, inoltre i livelli presentazione e sessione non hanno particolare utilità.
Il modello TCP/IP è invece composto da quattro livelli: applicazione, trasporto, internet e link. E' più semplice del modello ISO/OSI ma non marca particolarmente la distinzione tra protocolli, servizi ed interfacce, inoltre non differenzia il livello fisico dal livello data link.

In generale, i compiti e i protocolli di ogni livello sono:

- Il livello link/network/accesso rete, contiene per esempio i protocolli 802.11 e Ethernet.
- Il livello internet effettua il routing dei pacchetti e controlla la congestione della rete, vi fanno parte i protocolli IP (Internet Protocol) e ICMP (Internet Control Message Protocol).
- Il livello trasporto si occupa della trasmissione tra i peer, attraverso i protocolli TCP (Transmission Control Protocol), un protocollo reliable connection-oriented e UDP (User Datagram Protocol), un protocollo unreliable connectionless.
- Il livello applicazione contiene tutti i protocolli di alto livello, come per esempio TELNET (terminale virtuale), FTP (trasferimento di file), SMTP (e-mail), HTTP (protocollo per accedere alle pagine del World Wide Web),

RTP (protocollo per consegnare media in real-time), DNS (Domain Name System, il protocollo che mappa i nomi degli host ai loro indirizzi di rete.

Parte 2: Il livello fisico

La trasmissione delle informazioni parte dal livello fisico, ovvero dal collegamento e lo scambio di dati tramite mezzi fisici, che possono essere cavi, wireless e satellite.
Le connessioni possono essere di vario tipo: connessioni che possono essere utilizzate in entrambe le direzioni nello stesso momento sono dette **full-duplex**, connessioni che possono essere utilizzate in entrambe le direzioni, ma una direzione alla volta, sono dette **half-duplex**, connessioni che possono essere utilizzate in una sola direzione sono dette **simplex**.
La larghezza di banda (bandwidth), ovvero la capacità di trasmissione di un mezzo, si misura in **Hz**.

Mezzi di trasmissione guidata

Uno dei mezzi di trasmissione più vecchi e utilizzati è il **doppino intrecciato:** si tratta un cavo formato da una coppia di conduttori di rame intrecciati in forma elicoidale, tale intreccio riduce i disturbi causati dalle interferenze. Il segnale viene trasportato come differenza dei voltaggi dei due cavi, in questo modo, in caso di un eventuale disturbo, i due cavi vengono colpiti in maniera simile e la loro differenza non ne subisce particolarmente.
Il doppino intrecciato può viaggiare per alcuni chilometri prima che vi sia bisogno di un ripetitore, ed è utilizzato principalmente per le linee telefoniche e linee ADSL, e in generale è abbastanza popolare grazie al basso costo, ampia disponibilità e relativa efficacia.
Una categoria molto diffusa di doppino è la **categoria 5e**, che consiste in quattro coppie di fili di rame intrecciati: per esempio, una tipica connessione a 100 Mbps lavora in full-duplex utilizzando una coppia per ogni direzione, una connessione più performante potrebbe invece utilizzare due coppie per ogni direzione.
Successivamente sono state introdotte anche le categoria 6, 7 e 8, con la sesta categoria si parla di UTP (Unshielded Twisted Pair), in quanto fatti solo di cavi e isolanti. Con la categoria 7 si parla di STP (Shielded Twisted Pair), dove i cavi sono rivestiti di protezioni per ridurre le interferenze. Gli STP offrono alte prestazioni al costo di essere più pesanti e più costosi.

Un altro mezzo molto utilizzato è il **cavo coassiale**, costruito a partire da un singolo filo di rame rigido di forma cilindrica, ricoperto sequenzialmente da uno strato isolante, un conduttore intrecciato e una guaina di plastica. I cavi coassiali hanno una elevata larghezza di banda (fino a 6 GHz) e sono utilizzati principalmente per la TV cablata. In passato erano utilizzati molto per le linee telefoniche ma sono stati gradualmente rimpiazzati dalla fibra ottica ove possibile.

Infine, la **fibra ottica** consiste in cilindri di vetro al cui interno viaggiano raggi di luce. Questo mezzo di trasmissione utilizza una proprietà di rifrazione della luce: quando questa passa da un mezzo a un altro (per esempio dal vetro all'aria) con incidenza uguale a un certo angolo α la luce viene rifratta con un altro certo angolo β, se α viene scelto accuratamente, la luce può essere fisicamente "intrappolata" nel vetro.
La fibra ottica è costruita similmente al cavo coassiale: vi è un cilindro interno in vetro detto *core*, che può assumere dimensioni di una decina o cinquanta micron a seconda del tipo di fibra: la fibra ottica monomodale utilizza il core più piccolo, può contenere un solo raggio ed è più costosa ma raggiunge distanze maggiori, la fibra multimodale utilizza il core più grande, può contenere più raggi (raggi diversi possono propagarsi nella stessa fibra utilizzando angoli diversi) ma copre distanze inferiori.
Il core è rivestito da un *cladding* in vetro con un indice di rifrazione più basso per mantenere la luce nel core. Infine, il tutto è rivestito da un ulteriore strato protettivo in plastica.
La luce può essere inserita all'interno della fibra attraverso un LED o un laser semiconduttore, in ambo i casi vi è un *photodiode* alla fine che trasforma il segnale luminoso in elettrico (un impulso luminoso rappresenta un 1, mentre la sua assenza 0)
Ipoteticamente la fibra ottica potrebbe consentire fino a 50 Tbps di velocità, ma la conversione della luce in segnale elettrico limita molto il data rate.
La fibra porta a una attenuazione molto bassa, grazie all'utilizzo di tre particolari bande per la trasmissione, la cui lunghezza d'onda è tra l'infrarosso e l'UV, e rispetto al rame è più veloce, ha un'ampissima capacità di banda e può coprire distanze di gran lunga maggiori, oltre a pesare molto meno, ma una singola fibra può utilizzare solo la comunicazione simplex e nel breve termine è più costosa.

Mezzi di trasmissione wireless

La trasmissione wireless avviene sfruttando le proprietà delle onde elettromagnetiche. Il numero di oscillazioni per secondo di un'onda è detto frequenza f, la lunghezza tra due maxima o minima consecutive è detta lunghezza d'onda λ.

Nel vuoto: $\lambda \cdot f = c$.
Le informazioni possono essere trasmesse modulando le caratteristiche fisiche del segnale stesso, come **l'ampiezza (A)**, ovvero la differenza tra il valore massimo ed il valore minimo, il **periodo (T)**, ovvero la quantità di tempo prima che il segnale si ripeta, e la **fase (**ϕ), ovvero lo scostamento dall'origine, o la frequenza $f = \frac{1}{T}$.

La maggior parte delle trasmissione utilizza intervalli di frequenze molto ristretti, ma esistono alcuni metodi che utilizzano ampi intervalli di frequenze:

- **Frequency Hopping Spread Spectrum**, dove un trasmettitore "saltella" su un centinaio di frequenze al secondo, utile per trasmissioni militari data la difficoltà di jamming della comunicazione, questa tecnica è usata commercialmente nel Bluetooth e in vecchie versioni di 802.11.
- **Direct Sequence Spread Spectrum**, che utilizza un sistema di codici per dividere il segnale su diverse frequenze, utilizzato per la rete mobile 3G e per il GPS.
- **Ultra-Wideband Communication**, che funziona inviando una serie di impulsi a bassa energia, generando un segnale diffuso sottilmente su un ampio intervallo di frequenze. Usato prevalentemente per guardare attraverso oggetti solidi o per sistemi di localizzazione precisi.

Le onde con intervalli di frequenze più ristretti comprendono invece

1) Le onde radio a bassa frequenza, che si propagano in tutte le direzioni e attraversano gli edifici, e possono percorrere lunghe distanze specialmente a frequenze più basse, ma sono suscettibili alle interferenze.
2) Le onde radio ad alta frequenze, seguono una linea più retta delle onde radio a bassa frequenze, rimbalzano sugli ostacoli e sono assorbite dal terreno.
3) Le microonde, che sono direzionali e fanno uso di torri per ripetere il segnale in quanto hanno un raggio inferiore.. A differenze delle onde radio non attraversano gli edifici e possono essere soggetti al *multipath fading*, ovvero quando alcune onde arrivano in ritardo (fuori fase) cancellando il segnale. Alcune bande sono anche assorbite dalla pioggia.
4) Trasmissione infrarossi, utilizzata per comunicazione su piccole distanze (telecomandi televisori), non passano attraverso corpi solidi, sono direzionali e relativamente poco costosi.

Dalle onde ai bit

Attraverso le serie di Fourier, è possibile ricostruire una funzione periodica $g(t)$ con periodo T come somma di (possibilmente infiniti) seni e coseni:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \cdot sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cdot cos(2\pi nft)$$

Dove $f = \frac{1}{T}$, a_n e b_n sono le amplitudini della nth armonica, e c è una costante che determina il valore medio della funzione.

Per il **teorema di Nyquist**, un segnale che passa attraverso un canale senza rumore con larghezza di banda B, può essere interamente ricostruito effettuando almeno 2B campioni al secondo, andare oltre questo valore è superfluo. Se il segnale consiste di n livelli discreti, allora:

$Maximum\ data\ rate = 2B \cdot log_2(n)$ bits/sec

Successivamente Shannon ampliò questo teorema anche per canali rumorosi. Tenendo in considerazione un rumore casuale con un certo **Signal-to-noise Ratio** come S/N, (tipicamente espresso in decibel (dB) come $10 \cdot log_{10} S/N$), il **teorema di Shannon** dice che:

$Maximum\ data\ rate = B \cdot log_2(1 + S/N)$ bits/sec

Definite le proprietà dei canali cablati e senza fili, bisogna in qualche modo trasformare il segnale analogico in digitale. Il processo che converte tra bit e segnali è detto **modulazione.**

Ciò può essere fatto con la trasmissione a **banda base**, dove le frequenze occupano valori da zero fino ad un massimo che dipende dal segnale, tipico per esempio nei cavi, oppure con la trasmissione in **banda passante**, dove il segnale occupa frequenze vicine a quelle del vettore di trasmissione, come accade comunemente per esempio nei canali wireless e ottici.

Poiché i canali spesso condividono diversi segnali, bisogna in qualche modo poter fare in modo che un singolo cavo possa portarne molteplici, ciò avviene attraverso il **multiplexing**, che può essere realizzato attraverso metodi per il tempo, frequenza e *code division*.

Per convertire il segnale in bit, è possibile utilizzare uno specifico schema che associa il segnale ricevuto a una sequenza di bit. L'idea più semplice è quella di associare a un voltaggio positivo un bit uguale a 1, e a un voltaggio negativo un bit uguale a 0, questo schema prende il nome di **NRZ (Non-Return to Zero)**, che tuttavia non è praticamente adeguato al mondo reale. NRZ presenta alcuni problemi che possono essere risolti in altri schemi:

Con NRZ, il segnale può variare tra livelli positivi e negativi fino ad ogni due bit, per il teorema di Nyquist ciò significa che se la larghezza di banda è B, la performance massima ottenibile in termini di bit rate è 2B. Tuttavia la larghezza di banda può spesso essere un fattore limitato, di conseguenza quello che si può fare è utilizzare più di due livelli di segnale. Il tasso di cambiamento del segnale è detto **symbol rate** o **baud rate**, il bit rate è equivalente al baud rate moltiplicato per il numero di bit per simbolo.
Per tutti gli schemi è importante per il ricevitore capire quando un simbolo finisca, ciò può risultare difficoltoso se dovesse esserci una lunga fila continua di 0 o 1, ciò viene effettuato nel cosiddetto **Manchester Encoding**, dove il segnale di dati viene messo in XOR con un segnale di clock, ma ciò richiede il doppio della larghezza di banda.
In un altro schema, detto **NRZI (Non-Return to Zero Inverted)**, il segnale viene rappresentato codificando un 1 come un'inversione del segnale, e uno 0 come una non-inversione, in questo modo lunghe linee di 1 non causano problemi, ma rimane la questione degli 0.
Per risolvere ciò si possono utilizzare delle codifiche come il **4B/5B** dove gruppi di 4 bit vengono tradotti in un gruppo di 5 bit attraverso una tabella di traduzione predefinita, questo schema risolve il problema degli zeri ma aggiunge un 25% di overhead.
Un altro schema possibile è quello dello **scrambler**, dove il segnale viene messo in XOR con una sequenza pseudocasuale con un certo seed senza aggiungere overhead o sprecare banda, ma il problema è che è occasionalmente possibile essere “sfortunati” ed avere lunghe sequenze di 0 o 1.

Per le trasmissioni a banda passante, bisogna considerare il fatto che la larghezza di banda funziona allo stesso modo: è possibile prendere un segnale in banda base, che va da 0 a B, e trasformalo in banda passante che va da S a S+B senza cambiare la quantità di informazioni trasportate (anche se il segnale sarà diverso).
La modulazione digitale avviene modulando l’ampiezza, la frequenza o la fase e ognuno di questi metodi prende il nome apposito:

- L’**ASK (Amplitude Shift Keying)** dove due livelli differenti di ampiezza sono usati per rappresentare 0 e 1.
- L’**FSK (Frequency Shift Keying)** dove sono invece utilizzati due toni diversi.
- **L’PSK (Phase Shift Keying)** dove l’onda portante viene spostata opportunamente, nel caso di due fasi, abbiamo il **BPSK (Binary Phase Shift Keying)**

Quest’ultimo schema può essere ampliato utilizzando più spostamenti, lo schema **QPSK (Quadrature Phase Shift Keying)** utilizza quattro shift a 45, 135, 225 e 315 gradi, in modo da trasmettere due bit per simbolo.

E’ possibile rappresentare graficamente questi schemi attraverso i *constellation diagram*. Per ogni punto, l’angolo indica la sua fase, mentre la distanza dall’origine indica la sua ampiezza.
Ampliando ulteriormente il QPSK, è possibile trasmettere un numero maggiore di bit per simbolo: il **QAM-16 (Quadrature Amplitude Manipulation)**, può trasmettere 4 bit per simbolo attraverso 16 combinazioni, il **QAM-64**, può trasmettere 6 bit per simbolo con 64 combinazioni.
E’ importante notare che dato il fitto numero di combinazioni, un segnale arrivato con un certo quantitativo di rumore potrebbe portare al ricevitore la lettura di una combinazione vicina piuttosto che quella effettiva, bisogna quindi fare in modo che combinazioni adiacenti nel constellation diagram differiscano solo di 1 bit per limitare il margine di errore. Questo schema prende il nome di **gray code.**

Il multiplexing permette di trasmettere più flussi di bit sullo stesso mezzo, e può essere implementato utilizzando diverse tecniche:
Il **FDM (Frequency Division Multiplexing)** funziona dividendo la banda passante in bande di frequenza esclusive a ogni utilizzatore, ogni banda di frequenza allocata è leggermente più grande di quella effettivamente utilizzata in modo da evitare le interferenze, questo gap viene detto **guard band**. L’FDM è utilizzato principalmente per segnali analogici.
Quando si inviano dati digitali, è possibile rendere più efficiente questa tecnica riducendo la grandezza delle guard band, ciò avviene nell’**OFDM (Orthogonal Frequency Division Multiplexing)**, che funziona grazie all’utilizzo di un **guard time** utilizzato per ripetere una porzione del segnale. L’OFDM è utilizzato in 802.11, nel 4G, e nelle reti cablate.
Il **TDM (Time Division Multiplexing)** funziona in stile round-robin, ovvero per un determinato lasso di tempo ogni utente prende tutta la banda, aggrega la sequenza di bit da inviare alla sequenza di output, e poi si rimette in coda. L’**STDM (Statistical Time Division Multiplexing)** associa il lasso di tempo in base alle necessità di ogni utente.
Il **CDM (Code Division Multiplexing)**, comunemente chiamato **CDMA (Code Division Multiple Access)** funziona diffondendo il segnale su una banda di frequenze più larga, in modo che più utenti possano utilizzare la stessa frequenza e riducendo le interferenze. Per fare ciò utilizza il concetto di **chip sequence**, ovvero un sequenza di bit univoca a ogni segnalatore. Per trasmettere un 1, un utente trasmette la propria chip sequence così com’è, per trasmettere uno 0, trasmette l’inverso della chip sequence. Conoscendo i chip sequence di ogni trasmettitore e utilizzando alcune tecniche è possibile risalire da un unico segnale ai singoli segnali di ogni stazione, a patto che i chip sequence siano assegnati utilizzando un metodo conosciuto come **Walsh Codes**

Il **WDM (Wavelength Division Multiplexing)**, è fondamentalmente un FDM applicato alla fibra ottica, con ogni canale avente una sezione delle lunghezze d'onda. I raggi di bande differenti convergono in un'unica fibra, che viene divisa all'altro capo attraverso un prisma o sintetizzatore ottico.

La rete pubblica telefonica commutata

Con l'evoluzione della rete, il sistema telefonico è stato trasformato in modo da poter gestire anche la trasmissione di dati oltre che di chiamate, questo perché utilizzare le linee telefoniche già presenti è fondamentalmente più economico e semplice.
Il sistema telefonico è formato da tre componenti fondamentali:

- Il **local loop**, ovvero il collegamento tra le case e la più vicina centrale terminale. Il doppino è il metodo più diffuso ma negli ultimi anni si sta diffondendo anche l'utilizzo della fibra per questa componente.
- I **trunks**, che collegano le centrali di commutazione, utilizzando collegamenti con elevata larghezza di banda.
- Le **centrali di commutazione**, dove la trasmissione viene commutata da un trunk a un altro.

Nei punti terminali, per trasformare il segnale analogico in digitale e viceversa è necessario un **modem (modulator demodulator)** attraverso i metodi visti in precedenza.
Nel caso della linea telefonica, venivano utilizzati dei modem telefonici: poiché la banda telefonica lavora con 3000 Hz, per il teorema di Nyquist il baud rate necessario per ricostruire interamente il segnale nel caso ideale è di 6000. Nella pratica i modem telefonici permettevano un baud rate di 2400 simboli/sec: nello standard V.32 con 4 bit per simbolo, la velocità di connessione raggiungeva i 9600 bps, con lo standard V.34 bis e 14 bit per simbolo la velocità raggiungeva i 33600 bps.
In realtà la velocità era troncata anche dalla presenza di due local loop: rimuovendone uno, aumentando quindi l'SNR, il data rate massimo era quasi raddoppiato.
Ben presto il limite di velocità di questo schema iniziò a farsi sentire, con il **DSL (Digital Subscriber Line)**, viene rimosso il filtro del local loop che limita la banda a 3100 Hz, e viene utilizzata l'intera banda di 1 Mhz, che viene divisa in 256 canali da 4312.5 Hz l'uno, il primo viene utilizzato per la telefonia (chiamato anche **POTS, Plain Old Telephone Service**), 5 canali sono utilizzati per impedire interferenze tra il POTS e la linea internet, e i restanti 250 canali sono utilizzati per la trasmissione di dati, divisi tra dati in downstream e in upstream. Nella pratica circa l'80% dei canali viene utilizzato per il downstream in quanto la maggior parte degli utenti tende a

scaricare molto di più di quanto carica, per questo motivo la linea viene anche detta **ADSL (Asymmetric DSL)**.
L'ADSL si è evoluto col tempo cercando di spremere ogni possibile prestazione dalla linea telefonica, le versioni moderne sono in grado di offrire alte prestazioni, che però degrada velocemente all'aumentare della lunghezza del local loop.
Poiché la velocità del local loop è limitata fortemente dall'utilizzo del rame, un'alternativa costosa è quella del **FTTH (Fiber To The Home)** per aumentare ulteriormente le performance del local loop.

I trunk invece funzionano in modo diverso dal local loop non solo per il tipo di collegamento utilizzato, ma anche perché le informazioni sono digitali, piuttosto che analogiche. Inoltre vi è la necessità di trasportare migliaia e milioni di chiamate simultaneamente, ciò avviene attraverso il multiplexing.
Inizialmente, veniva usato l'FDM per effettuare questa operazione con canali di 4000 Hz, 12 canali venivano uniti in un gruppo, 5 gruppi formavano un supergruppo.
Successivamente il TDM è diventato più diffuso in quanto può essere gestito interamente in maniera digitale,
I segnali analogici sono digitalizzati nelle centrali terminali attraverso i **codec (coder-decoder)** usando una tecnica chiamata **PCM (Pulse-Code Modulation)**. Il codec crea 8000 samples al secondo, (abbastanza per i 4 kHz di un canale telefonico) quindi 125 microsecondi per sample. Se si inviano 8 bit ogni 125 microsecondi si ha una connessione di 64 Kbps.
Il modo in cui più canali TDM sono trasmessi su un unico circuito è detto **T-Carrier**, in America e Giappone si utilizza il T1 carrier, in Europa l'E1 carrier.
I carrier vengono poi multiplexati in carrer di livello più alto, che combinano la larghezza di banda dei singoli carrier più alcuni bit extra per framing e recovery.
Il **SONET (Synchronous Optical Network)** permette invece di fare il multiplexing di network ottici, con l'idea di rendere diversi carrier interoperabili. Un frame SONET è formato da 810 byte (6480 bit) inviati 8000 volte al secondo.

Infine, la commutazione può avvenire in due modi: **commutazione di circuito** e **commutazione di pacchetto.**
Nella commutazione di circuito, viene instradata un'unica via nella rete tra due estremi, e tutti i dati attraverso questa via. Nella commutazione di pacchetto, i dati vengono divisi in pacchetti inviati in maniera simile al funzionamento della posta, sono cioè instradati router per router fino alla destinazione. I pacchetti potrebbe non arrivare in ordine, devono quindi essere eventualmente riordinati a destinazione, inoltre non è necessario il setup della connessione ma potrebbero esservi ritardi in caso di congestione.

La rete cellulare

La rete cellulare è la rete utilizzata dai telefoni in modo da soddisfare le emergenti necessità di utilizzare la rete quando si è in movimento. Nella storia, la rete cellulare sia è evoluta in diverse generazioni, ma tutte hanno mantenuto alcuni concetti chiave: l'area geografica viene divisa in **celle**, ogni cella utilizza un qualche insieme di frequenze non utilizzato dalla celle immediatamente adiacenti, ma questo insieme può essere riutilizzato da cellule più distanti: in questo modo si utilizza un numero contenuto di frequenze evitando comunque le interferenze. Ogni cella è gestita da un trasmettitore, connesso a un device chiamato **MSC (Mobile Switching Center)**, in una rete cellulare molto grande, diversi MSC potevano a loro volta essere connessi in MSC di secondo livello, e così via.
In ogni caso, quando il segnale di una cella diventa troppo basso, è necessario effettuare il passaggio da una cella a quella più vicina, questa operazione viene detta **handoff.**
Per identificare un utente, dalla seconda generazione in poi si utilizzano le **SIM (Subscriber Identity Module)**, la SIM viene poi associata a un handset, e si connette alla cella più vicina. L'MSC mantiene un database con tutte le SIM connesse e l'ultima cella dove essere si trovavano.
Sempre dalla seconda generazione, la gestione dell'handoff viene effettuata attraverso il design **MAHO (Mobile Assisted HandOff)**
Nelle ultime generazioni, a causa dell'elevatissimo traffico, sono state introdotte delle **microcelle** (dalla quinta generazione anche delle picocelle o persino femtocelle), ovvero delle celle di dimensioni sempre più ridotte in modo da far fronte al numero maggiore di utenti, oltre a cercare di aumentare costantemente le velocità di trasferimento dati fornite.

Reti cablate e satellitari

La rete cablata nasce con l'intento di trasmettere dati attraverso la rete utilizzata per la televisione. Poiché la TV utilizza la banda dai 50/70 Mhz fino ai 550/700 Mhz, per usufruire dei cavi già presenti si utilizzano i primi 50 Mhz per l'upstream e una banda dai 550/700 in poi per il downstream. Dal 1997 lo standard **DOCSIS (Data Over Cable Service Interface Specification)** definisce i metodi per trasmettere dati sul cavo televisivo. Il problema principale con questa tecnologia è che a differenza della rete telefonica dove ogni utente ha il suo local loop, in questo caso è solitamente presente un modem che si interfaccia con **l'headend** dell'ISP attraverso un cavo coassiale o doppino, mentre l'headend è connesso alla rete dell'ISP con la fibra e deve gestire un certo numero di modem. Con il passare degli anni si sta cercando sempre di più di avvicinare la fibra alle case "splittando" i nodi e collegandoli direttamente alla

fibra **(node split)**, in modo che ognuno di essi debba gestire meno utenti, pertanto ottenendo velocità maggiori.

La rete satellitare si basa invece sull'utilizzo di satelliti in orbita per ricevere e riflettere il segnale. I satelliti possono essere di tipo **GEO (Geostationary Earth Orbit)**, **MEO (Medium-Earth Orbit)** e **LEO (Low-Earth Orbit).**
I satelliti GEO sono utilizzati per alcune comunicazioni grazie al fatto che il loro movimento attorno alla terra è più lento, i satelliti MEO sono utilizzati per il GPS, i satelliti LEO sono infine utilizzati per la connessione satellitare verso gli utenti.

Parte 3: Il livello data link

Il livello data link ha lo scopo di ricevere e inviare dati su canali di comunicazione possibilmente inaffidabili, che possono anche perdere dati. In particolare, questo livello deve garantire alcune importanti funzioni:

- Fornire un'interfaccia ben definita al livello Network
- Inquadrare sequenze di bit in blocchi autonomi
- Individuare e/o correggere errori nella trasmissione
- Regolare il flusso di dati in modo che un mittente veloce non inondi di dati un destinatario più lento

Per adempiere a questi obiettivi, il livello data link utilizza il concetto di **frame**, ovvero una sequenza di bit il cui inizio e fine sono marcati rispettivamente da un **header** e da un **trailer.**

Il livello data link di un mittente comunica virtualmente con il collega nel destinatario, sebbene il flusso di dati effettivo passi attraverso gli altri livelli. In ogni caso, è in grado di offrire alcuni servizi a seconda del protocollo:

- **Unacknowledged connectionless service**, dove il mittente invia frame indipendenti, senza aspettare una conferma di ricezione, può essere per esempio usato se il livello fisico sottostante è abbastanza affidabile (Ethernet)
- **Acknowledged connectionless service**, dove il mittente invia frame indipendenti ma attende una acknowledgment, in caso non arrivi entro la scadenza di un timer invia nuovamente quel frame. Può essere usato se il canale di trasmissione è meno affidabile (802.11, sistemi wireless in generale)
- **Acknowledged connection-oriented service**, dove il mittente e il destinatario instaurano una connessione prima di iniziare a inviare dati, utile soprattutto per canali lunghi e altamente inaffidabili.

La gestione dei frame può avvenire in modi diversi, ma bisogna fare in modo che il destinatario sia in grado di capire esattamente dove inizia e dove finisce un frame, sprecando meno banda possibile.
Il metodo più semplice è quello del **byte count**, dove l'header di un frame contiene semplicemente il numero di byte del frame che segue. Il problema è che ci può essere una desincronizzazione nel caso in cui i bit del byte count siano errati.
Il secondo metodo è quello del **byte stuffing**, dove ogni frame è racchiuso tra due **flag**, ovvero una sequenza di bit, che ne marca l'inizio e la fine. Nel caso in cui la sequenza sia contenuta all'interno dei dati, essa viene preceduta da una sequenza di escape. La sequenza di escape viene inserita e rimossa direttamente dal livello data link, e i dati vengono passati così come sono in origine al livello Network.
Il terzo metodo è quello del **bit stuffing**, dove il codice di escape è un unico bit piuttosto che un intero byte. Per esempio nel **HDLC (High-Level Data link Control)**, il frame flag era 01111110, nel momento in cui il livello data link incontrava 5 bit 1 consecutivi nei dati, automaticamente inseriva uno zero. Dall'altro lato, il livello data link del destinatario si accorgeva dei 5 bit 1 e rimuoveva lo 0.
L'ultimo metodo per il framing è detto di **"coding violations"** del livello fisico, e dipende appunto dal livello fisico, ovvero nel caso in cui la trasmissione di bit effettiva avvenga utilizzando un encoding come per esempio 4B/5B, si hanno alcune sequenze non utilizzate. Il livello data link può quindi sfruttare queste sequenze per indicare la fine e l'inizio di un frame.
Nella pratica, i protocolli data link utilizzano una combinazione di questi metodi, una tecnica comune usata per esempio in Ethernet e 802.11 è quella di utilizzare un preambolo lungo 72 bit, seguito da un campo nell'header che indichi la lunghezza del frame.

La gestione degli errori deve garantire che nelle connessioni con acknowledgment un mittente non rimanga bloccato in attesa di esso, si introducono quindi dei timer, ovvero dei tempi massimi di attesa dell'acknowledgment. Tuttavia, bisogna anche risolvere il problema nel caso in cui il frame sia ricevuto ma l'ack venga perso, per fare ciò si possono numerare i frame inviati.

Il controllo del flusso di dati può essere invece implementato tramite feedback (**feedback-based flow control**) o limitando il data rate (**rate-based flow control**). Nel primo caso il mittente invia *n* frame al destinatario, e attende un feedback da esso che indichi se è possibile inviare altri *n* frame o meno.

La gestione degli errori può avvenire in due modi: gli errori possono essere automaticamente corretti **(error-correcting codes)** o semplicemente individuati,

richiedendo una nuova trasmissione del frame **(error-detecting codes)**. In entrambi i casi, bisogna aggiungere dei dati ridondanti, ovvero dei bit che non fanno effettivamente parte dei dati. La correzione degli errori richiede più ridondanza, ma in alcuni casi può essere un'opzione migliore per canali molto rumorosi se il reinvio del frame può essere nuovamente danneggiato, in altri casi può invece essere più conveniente inviare nuovamente i dati, se il tasso di errore del canale è molto basso (fibra, diverse LAN, etc..).
In generale, un frame, fatta eccezione dei delimitatori, consiste in m bit di dati, e r bit di ridondanza, con $d=m+r$ bit totali. Una **codeword** è una sequenza di d bit. Un insieme prefissato di codeword costituisce un codice. **La distanza di Hamming** di un codice è il minimo delle distanze di Hamming fra tutte le possibili codeword del codice.
Per **rilevare** d errori, serve un codice con distanza di Hamming uguale almeno a $\boldsymbol{d+1}$.
Per **correggere** d errori, serve un codice con distanza di Hamming uguale almeno a $\boldsymbol{2d+1}$.

Consideriamo quattro codici di correzione degli errori:

1. Nei **codici di Hamming**, si utilizzano dei bit di parità (ndr leggi sotto), per ogni frame, vengono posti i bit di parità nelle posizioni potenze di due, e i bit di dati nelle altre posizioni, in modo che ogni bit di dati venga controllato da un certo numero di bit di parità. Per esempio il primo bit è ottenuto controllando la parità di tutti i bit la cui posizione rappresentata in binario contiene 1 nella posizione meno significativa. Per correggere un errore, si controlla la parità di tutti i bit di parità, se la sequenza ottenuta è diversa da 0, il bit la cui posizione è il numero ottenuto da questa sequenza viene invertito.
2. **Codici binari convoluzionali**, che generano una sequenza di bit di output a partire da alcuni bit di input.
3. I **Codici Reed-Solomon** si basano sul fatto che ogni polinomio di grado n può essere ottenuto da $n+1$ punti. Per esempio se si inviano 3 punti che giacciono sulla stessa retta, ma 1 di questi è sbagliato, è possibile trovare la retta a partire dagli altri due punti e correggere il bit sbagliato. Questo metodo è molto potente nella correzione di burst di errori, ed è utilizzato per esempio nei CD (per evitare la perdita di dati in seguito a graffi, etc..)
4. Nei **Codici LDPC (Low-Density Parity Check)**, si utilizza un codice dove ogni bit di output è formato da solo una piccola parte dei bit di input, formando una matrice sparsa. Le codeword ricevute sono iterativamente approssimate fino a quella più vicina.

Consideriamo tre codici di rilevamento degli errori:

1. Nel **Codice di parità** si aggiunge un unico bit alla fine di una sequenza di bit, ottenuto in base alla parità o meno della sequenza (per esempio facendo lo XOR). E' un metodo molto semplice che può rilevare errori su un singolo bit ma non è efficace per burst di errori. Tuttavia i burst di errori si possono affrontare utilizzando la tecnica dell'interlacciamento, dove i dati vengono disposti in *n* colonne, e inviati in *k* righe, calcolando il bit di parità per ogni colonna.
2. Per **Codici di checksum** si intende un gruppo di check bits associati ad un messaggio. I codici di checksum calcolano la somma dei bit di check, in questo modo, se due bit sono diversi la parità rimane la stessa, ma la somma rileva l'errore.
3. **Controlli di ridondanza ciclici (CRCs)**. I CRC calcolano il checksum in maniera diversa, utilizzando un codice di ridondanza ciclico, ovvero una sequenza di *m* bit che corrisponde a un polinomio di grado *m-1*. Per effettuare il controllo sia il mittente che il destinatario utilizzano lo stesso polinomio generatore *G(x)*, dove il bit meno significativo e quello più significativo sono entrambi 1, successivamente si appende davanti al frame da inviare un un checksum tale che il polinomio frame+checksum sia divisibile per *G(x)*. Quando il ricevitore riceve il frame, effettua questa divisione, se il resto è 0, allora non ci sono errori, altrimenti c'è stato un errore. Il calcolo del checksum avviene in questo modo: si appendono *r* bit 0 a destra del frame da inviare, e si divide la sequenza ottenuta per *G(x)*, il resto divisione divisione è il checksum.

Protocolli data link

Consideriamo prima di tutto una struttura semplice di frame: abbiamo un frame formato da un campo **info**, contenente le informazioni effettive da passare al livello network. Un campo **kind** per indicarne il tipo (dati, controllo, etc..), un campo **seq**, per indicare il numero progressivo del frame, e un campo **ack** per leggere le informazioni relative all'acknowledgment. Questi ultimi tre campi fanno parte dell'header.
Il protocollo data link più semplice è il protocollo **utopia**, o **heaven**, in quanto assume le condizioni più semplici e ideali possibili: comunicazione simplex, il mittente ha sempre dati pronti da inviare, il ricevente ha un buffer infinito per processarli, non ci sono errori di trasmissione e non si possono perdere frame.
Il primo problema da risolvere è la situazione molto comune in cui un mittente troppo veloce riempia un destinatario di più dati di quanti ne possa processare, in questo secondo protocollo, detto **stop-and-wait**, il mittente si blocca in attesa che il destinatario invia un ack che consenta l'invio di un nuovo messaggio. La

comunicazione in questo caso è half-duplex, e la comunicazione è ancora esente da errori.
Il terzo tipo di protocollo, detto **ARQ (Automatic Repeat ReQuest)** o **PAR (Positive Acknowledgment with Retransmission)**, risolve il problema della possibilità di perdita di frame, in particolare di un ack: se un ack viene perso, e il mittente andato in timeout invia nuovamente lo stesso frame, il destinatario deve essere in grado di riconoscere il frame duplicato e scartarlo. Se il mittente deve attendere l'acknowledgement, questo problema può verificarsi solo tra un frame e il suo successivo, di conseguenza in questo protocollo è necessario un solo bit di sequenza.

Si possono apportare ancora alcuni miglioramenti a questi schemi: permettere al canale di trasportare dati in entrambe le direzioni contemporaneamente, e permettere l'invio di più frame contemporaneamente.
Il concetto di **piggybacking**, rende più efficiente la comunicazione bidirezionale: nel momento in cui A e B si scambiano dei frame, nel dover inviare i necessari frame di acknowledgment inseriscono l'ack in coda al messaggio da inviare, nel campo apposito.
L'invio di più frame contemporaneamente è implementato tramite il concetto di **sliding window**: il mittente mantiene una "finestra" di frame da inviare, nel momento in cui riceve un ack per il frame con il numero di sequenza più basso che si voleva trasmettere, si avanza la finestra di uno e si invia un nuovo frame. Se si riceve un ack diverso o scade il timer, il frame viene inviato nuovamente.
Il destinatario invece risponde con un ack nel momento in cui riceve il frame corretto o un duplicato, i duplicati vengono scartati.
Nel momento in cui la finestra ha grandezza di 1 bit, questo protocollo è fondamentalmente equivalente allo stop-and-wait.
Il problema principale tuttavia è che se il tempo di *round-trip* è alto, lo stop-and-wait risulta altamente inefficiente, questo che si può fare è quindi permettere l'invio di frame multipli senza attendere l'ack del precedente. Questo metodo prende il nome di **pipelining.**

Un primo protocollo viene detto **go-back-n**, in questo protocollo tutti i frame successivi al prossimo da inoltrare al livello network sono scartati, la grandezza della finestra di arrivo è quindi effettivamente 1. Il problema di questo protocollo è che nel momento in cui il canale di comunicazione presenta molti errori può essere particolarmente inefficiente.
Un secondo protocollo, detto **selective-repeat**, mantiene ogni frame ricevuto dal destinatario in un buffer, aspettando che arrivi il prossimo necessario per inoltrarli ordinatamente al livello network. Il vantaggio di questo metodo è che è più efficiente nell'utilizzo della connessione ma utilizza più memoria di buffer.

Questo protocollo utilizza inoltre un meccanismo di **acknowledgment negativo (NAK)**: è un frame inviato dal destinatario nel momento in cui riceve un frame fuori sequenza o vi è un errore di checksum. Il NAK serve fondamentalmente a evitare di attendere che scada il timer del mittente. Ad ogni modo, se il NAK va anch'esso perso non vi sono problemi, in quanto eventualmente il mittente andrà in timeout e invierà un nuovo frame.

I protocolli data link utilizzati in pratica nelle WAN dipendono dalla situazione, in particolare se si considera l'invio di pacchetti tramite SONET su fibra, ADSL presenti sul local loop oppure link DOCSIS per il local loop di rete cablata.

Il protocollo **HDLC (High-level Data Link Control)** è un protocollo bit-oriented che utilizza il bit stuffing. Un frame HDLC inizia e termine con un flag di 8 bit che è sempre 01111110. Il secondo campo, *address* serve a identificare i diversi terminali nelle linee multipunto, il terzo campo *control* contiene le informazioni di controllo come numeri di sequenza, ack, etc.., vi è poi il campo dati e infine un checksum implementato dal CRC a 16 bit.
Il campo control contiene diverse informazioni: i primi due bit indicano il tipo di frame (*Information* per i dati, *Supervisory* per comandare le modalità di trasmissione, *Unnumbered* per finalità di controllo)

Il protocollo **PPP (Point-to-Point Protocol)** è un miglioramento di un precedente protocollo SLIP, provvedendo tre caratteristiche principali:

- Un meccanismo di framing che definisce univocamente l'inizio e la fine di un frame e che gestisce il rilevamento degli errori.
- Un protocollo di controllo link che permette di instaurare una connessione, negoziarne le caratteristiche e chiuderla, chiamato **LCP (Link Control Protocol)**
- Un modo per negoziare le opzioni del livello network, il metodo scelto deve avere un diverso **NCP (Network Control Protocol)** per ogni livello network supportato.

Il frame PPP è simile al frame HDLC, ma si basa sul byte-stuffing piuttosto che sul bit-stuffing, ed è composto come segue:
Il primo e ultimo campo è sempre il flag 01111110 o *0x7E*. Il byte di escape è 0x7D, che precede immediatamente il byte da evitare e a cui viene fatto uno XOR con 0x20, per esempio il byte di escape di 0x7E è 0x7D 0x5E. La regola di destuffing consiste semplicemente nel rimuovere lo 0x7D e mettere 0x5E in XOR con 0x20. In questo modo è possibile cercare l'inizio e la fine di un frame semplicemente guardando il byte di flag.

Il campo successivo è l'*address* che è sempre 11111111, poi il campo *control*, il cui valore di default 00000011 indica un unnumbered frame.
Il campo *protocol*, indica il tipo di pacchetto presente nel Payload: se il primo bit è 1 allora il pacchetto contiene una configurazione del PPP, per esempio l'LCP o l'NCP, se il primo bit è 0 allora indica se si sta usando IPv4, IPv6, IPX, etc..
Il *Payload*, può essere di dimensione variabile, di default è di 1500 byte.
Infine, vi è il checksum di 2 o 4 byte, per implementare il CRC di grandezza apposita.

Parte 4: Il sottolivello MAC

Il sottolivello **MAC (Medium Access Control)** è un sottolivello del livello data link che si occupa di implementare i meccanismi per la gestione delle linee multi-accesso o ad accesso casuale.
L'assegnazione dei canali può avvenire in maniera statica o dinamica. L'allocazione statica è estremamente inefficiente e pertanto utilizzata molto raramente, per esempio nella radio dove ogni stazione ha il proprio spettro di banda. Tuttavia nella gestione pratica, dove il numero di utenti e l'utilizzo delle frequenze può variare ogni momento, diventa necessario l'utilizzo di un protocollo che consenta l'allocazione dinamica.
L'allocazione dinamica si basa su un modello con le seguenti cinque ipotesi:

- Il modello è costituito da **N stazioni** che inviano traffico in maniera indipendente
- Un **singolo canale** è disponibile per tutte le stazioni, che possono trasmettere e ricevere tramite esso.
- **Collisioni osservabili**, se due frame sono trasmessi contemporaneamente si ha una collisione e i due frame sono inutilizzabili, bisogna pertanto gestirle ed effettuare una ritrasmissione
- **Tempo continuo o discreto,** ovvero le stazioni possono trasmettere in qualsiasi momento OPPURE solo all'inizio di uno slot
- **Carrier sense o No carrier sense**, ovvero la possibilità o meno per le stazioni di ascoltare il canale e capire se qualcun altro sta trasmettendo.
-

ALOHA è nato con l'obiettivo di connettere tramite radio i computer dei laboratori delle Hawaii.
La versione iniziale è detta **pure ALOHA**, in questo protocollo ogni stazione può trasmettere a piacimento, le stazioni ascoltano il canale e se notano una collisione smettono di trasmettere, attendono un intervallo casuale e riprovano.

Detto *frame time* il tempo necessario per trasmettere un frame e N è il numero medio di frame per frame time definito da una distribuzione, se N > 1, vengono inviati più frame di quanti il canale può gestirne, e si avranno quasi sempre collisioni, mentre tra 0 e 1 il numero di collisioni può essere più o meno accettabile, ma richiedono comunque delle ritrasmissioni.
Considerando sia i frame inviati che ritrasmessi come G, chiaramente $G >= N$, se il carico è basso, quindi $N \approx 0$, allora $G \approx N$, se il carico è alto, $G > N$.
In ogni caso, se consideriamo il throughput del sistema come $S = G P_0$, dove P_0 è la probabilità di una trasmissione con successo.

La probabilità che *k* frame siano generati in un qualsiasi frame time è dato dalla **distribuzione di Poisson:**

$$Pr[k] = \frac{G^k e^{-G}}{k!}$$

La probabilità che 0 frame siano generati è quindi semplicemente

$Pr[0] = \frac{G^0 e^{-G}}{0!} = e^{-G}$

Poiché una collisione si verifica se un frame viene generato nel **periodo di vulnerabilità**, ovvero il frame time appena precedente all'invio del frame e durante l'invio del frame, consideriamo quindi due frame time, e la probabilità che non vi siano collisioni è e^{-2G}.

Quindi $S = G * P_0 = G e^{-2G}$

S è massimo quando $G = 0.5$ e $S = \frac{1}{2e} \simeq 0.184$, quindi un utilizzo del canale del 18%.

Un primo miglioramento ad ALOHA è stato lo **slotted ALOHA**, dove il tempo viene discretizzato, in questo modo il periodo di vulnerabilità viene dimezzato, e applicando nuovamente le formule precedenti otteniamo:

$S = G e^{-G}$, che è massimo se $G = 1$ e $S = \frac{1}{e} \simeq 0.368$, ergo un utilizzo del 37% del canale, il doppio più efficiente di pure ALOHA.

Il problema principale dei protocolli ALOHA è che non cercano in alcun modo di evitare attivamente le collisioni. Alcuni altri protocolli introdotti successivamente, detti **CSMA (Carrier Sense Multiple Access)**, utilizzano il carrier sense per decidere quando trasmettere:
Il **1-persistent CSMA**, è il protocollo CSMA più semplice, ogni stazione ascolta il canale per decidere se inviare: se il canale è idle invia i dati, altrimenti attende fino a quando non si libera. Il problema di questo protocollo è che se più canali attendono

che un altro canale finisca la trasmissione, invieranno contemporaneamente i propri frame causando collisioni. E' detto 1-persistent poiché ogni stazione invia con probabilità 1 nel momento in cui il canale è libero.
Il **non-persistent CSMA** è un'altra variante dove invece di aspettare attivamente che il canale si liberi, si aspetta un periodo casuale di tempo.
Il **p-persistent CSMA** è un'ulteriore versione che si applica ai canali con tempo discreto: ogni stazione controlla se il canale è libero, se lo è, trasmette con una probabilità p, oppure attende per un periodo casuale di tempo con una probabilità $q = 1 - p$, effettua ripetutamente questa decisione fino a quando il frame non è stato inviato oppure qualcun altro canale sta trasmettendo, in quest'ultimo caso aspetta fino allo slot successivo.

Questi ultimi due protocolli risolvono parzialmente le collisioni ma introducono problemi di ritardi dovuti alle numerose attese.
Infine, vi è il **CSMA/CD (Collision Detection),** in questo protocollo ogni stazione ascolta il canale mentre trasmette per rilevare le collisioni. Se due stazioni sono a distanza T, il rilevamento della collisione nel caso peggiore avviene dopo $t_0 + 2T$ dove t_0 indica l'inizio della trasmissione.
Concettualmente, il CSMA/CD funziona alternando periodi di invio del frame, di **contesa** e di inattività. Il periodo di contesa è modellato come uno slotted ALOHA con slot di durata 2T.

I protocolli **collision-free** sono invece protocolli che utilizzano alcuni metodi per evitare le collisioni.
Il **protocollo bit-map** è costituito da periodi di contesa di N slot, durante ogni periodo di contesa ogni stazione utilizza un bit per indicare se vuole trasmettere o meno (1 in caso positivo, altrimenti 0), finito questo periodo, ogni stazione può poi inviare il proprio frame nello stesso ordine.
In media, una stazione con un numero basso dovrà aspettare il periodo di contesa successivo (*0.5N + N*), mentre una con un numero alto potrà inviare in quello stesso periodo (*0.5N*), l'attesa media è quindi pari a *N*.
In generale, se una stazione deve inviare *d* bit, l'overhead per frame è di $d/(d + N)$, se tutte le stazioni devono inviare qualcosa, ogni frame avrà un overhead di 1 bit, quindi $d(d + 1)$.

L'essenza del protocollo bit-map è quello di dare un ordine nell'invio dei frame per evitare le collisioni, questo può essere ottenuto anche "ordinando" le stazioni a

formare un anello creando una **rete ad anello**, come per esempio nel protocollo **token ring**.
In questo protocollo, ogni stazione aspetta *(listen mode)* che arrivi un token, lo rimuove dal ring *(listen mode)*, trasmette il messaggio che deve inviare *(transmit mode)* e rigenera il token *(transmit mode)*. Il token effettivamente indica la possibilità per una stazione di poter trasmettere, ed evita le collisioni.
Quando tutte le stazioni hanno qualcosa da trasmettere, l'efficienza di questo protocollo è del 100%, se il carico è più basso,ogni stazione mediamente deve comunque attendere che il token effettui un giro di tutto l'anello.

Altri protocolli includono:
Binary countdown, dove a ogni stazione viene associato un certo indirizzo che ne indica la priorità, durante il periodo di contesa le stazioni confrontano i loro indirizzi bit per bit, quelle con bit pari a 1 vincono, quelle con bit pari a 0 si arrendono, alla fine la stazione con l'indirizzo più alto può inviare. L'overhead di questo metodo è $d/(d + log_2 N)$.
Protocollo ad albero, dove ogni stazione viene inserita sulle foglie di un albero binario, durante il periodo di contesa viene effettuata una ricerca depth-first sull'albero: se una stazione deve trasmettere, metterà il proprio bit a 1, se c'è almeno una stazione in un sottoalbero che deve trasmettere, sul nodo radice di quel sottoalbero ci sarà un 1. In questo modo la ricerca trova il primo nodo con bit a 1, e permette di migliorare l'efficienza con il pruning, ovvero evitando di continuare a cercare un sottoalbero se il nodo radice del sottoalbero ha il bit a 0.

Protocolli Ethernet

I protocolli Ethernet connettono più stazioni attraverso un cavo particolare, cavi multipli possono essere connessi con ripetitori. Ogni cavo Ethernet contiene dei **transceiver**, ovvero dei punti di aggancio a cui collegare i dispositivi.
L'Ethernet usa la codifica manchester e ogni stazione invia un segnale di **jamming** di 32 bit nel caso in cui rilevi una collisione per dire alle altre stazioni di scartare quanto ricevuto.
Il formato del frame Ethernet è il seguente:

- I primi 8 byte formano un **preambolo**, ogni byte è 10101010 ad eccezione dell'ultimo che è 10101011, per indicare che sta per iniziare il frame.
- I successivi 6 byte formano il **destination address**, secondo uno standard IEEE i primi 3 byte sono associati a dei *manufacturers* che utilizzano i restanti 3 byte per indicare univocamente i dispositivi. L'indirizzo speciale con tutti bit a 1 indica il *broadcasting*, ovvero che tutte le stazioni possono ricevere. Se il

primo bit dell'indirizzo è 0 la comunicazione è unicast, se 1 è multicast. Il secondo bit differenzia gli indirizzi globali (bit 0, assegnati da IEEE) da quelli locali (bit 1)

- Analogamente vi sono altri 6 byte per **l'indirizzo sorgente.**
- I successivi 2 byte indicano il **tipo** nell'Ethernet DIX e la **lunghezza** del frame nell'IEEE 802.3, per evitare ambiguità i dispositivi che utilizzano IEEE 802.3 associano numeri inferiori a 0x600 come lunghezza, e numeri superiori come tipo. Per esempio 0x800 indica un frame IPv4.
- Vi è poi il segmento di **dati** effettivo, limitato a 1500 byte per limitazioni hardware (RAM dei dispositivi molto costosa)
- Per evitare l'invio di frame troppo piccoli, ovvero per assicurarsi che ogni frame sia inviato almeno con un tempo 2τ per rilevare eventuali collisioni in tempo, si aggiungono alcuni byte di **padding**, da 0 a 46 a seconda della lunghezza dei dati.
- Infine, vi sono 4 byte per il **checksum**, che usa il CRC standard di 32 bit per rilevare errori, ma non correggerli.

L'Ethernet classico utilizza il CSMA/CD, ma dove ogni stazione aspetta un periodo casuale di slot di tempo prima di ritrasmettere dopo aver trovato una collisione, questo periodo è inizialmente o 0 o 1 slot, se rileva una ulteriore collisione attende 0, 1, 2 o 3 slot, in generale alla i-esima collisione attende $2^i - 1$, fermandosi a 1023. Alla 16 collisione si arrende e comunica un errore al computer.
Questo metodo, detto **binary exponential backoff**, permette dinamicamente alle stazioni di attendere poco se il carico è basso, e di evitare pseudo-infinite collisioni se molte stazioni stanno trasmettendo.

Nell'ipotesi di *k* stazioni sempre pronte a trasmettere con probabilità di trasmissione *p* costante per ogni slot, la probabilità *A* che sul canale vi sia una trasmissione è:
$A = kp(1 - p)^{k-1}$
In questa equazione, k indica ogni stazione disponibile, una stazione trasmette con probabilità p, ma le altre *k-1* no *(1-p)*.

Nel caso di carico costante il numero medio di slot di contesa è 1/A, e poiché ogni slot ha durata 2τ l'intervallo medio di contesa è $2\tau/A$. Per un frame di F bit la trasmissione attraverso un canale di banda B richiede T secondi, con $T = F/B$. Conseguentemente, l'efficienza del canale $E = T/(T + 2\tau/A)$ **diminuisce con la lunghezza della linea** τ, **aumenta con la dimensione** F del frame e **decresce al numero delle stazioni** pronte a trasmettere.

Con frame di 1024 byte, le performance Ethernet sono dell'85%, molto meglio del 37% di slotted ALOHA, tuttavia l'Ethernet classico non è in grado di supportare lunghe linee o bandwidth più elevate.

Normalmente l'Ethernet non funziona più come un'unica linea a cui i dispositivi si collegano, ma vi è un hub centrale a cui ogni dispositivo si connette con il proprio cavo. L'hub permette di rimuovere o aggiungere stazioni facilmente e se un cavo smette di funzionare il danno è fine a sé stesso, tuttavia all'aumentare dei dispositivi connessi le prestazioni degradano.
Per questo motivo sono nati gli **switched Ethernet**, dove al posto dell'hub vi è uno switch. Esternamente il funzionamento è identico, ma internamente lo switch è in grado di direzionare ogni frame evitando completamente le collisioni, inoltre a differenza dell'hub che funziona in maniera **promiscua**, ovvero ogni stazione sa se le altre stazioni stanno inviando o ricevendo traffico, nello switch ogni collegamento è come se fosse isolato. Lo switch non richiede uno standard, ogni produttore può utilizzare una architettura interna, a patto che il funzionamento visto dall'esterno rimanga identico.

Successivamente, lo standard 802.3*u*, detto anche **Fast Ethernet** introduce miglioramenti all'Ethernet classico con l'obiettivo di portare la velocità da 10 a 100 Mbps, a seconda del mezzo abbiamo:

- **100Base-T4**, utilizza una velocità di segnalazione di 25 Mhz, adottando 4 doppini Cat.3 per ogni stazione
- **100Base-TX**, utilizza 2 doppini Cat.5e sfruttando l'encoding 4B/5B.
- **100Base-FX**, utilizza due cavi in fibra e può raggiungere distanze maggiori a patto si utilizzi uno switch per evitare le collisioni.

Con gli anni, IEEE ha introdotto numerosi e sempre più veloci standard per 802.3, per supportare velocità da 1 Gbps fino alle più recenti 400 Gbps per *data center* e *backbones* della rete:

- **1000Base-SX** utilizza la fibra ottica multimodale. (S sta per Short)
- **1000Base-LX** utilizza la fibra ottica singola o multimodale su lunghe distanze (L per Long)
- **1000Base-CX** utilizza due doppini a piccole distanze (max 25m)
- **1000Base-T** utilizza quattro doppini Cat.5 per distanze fino a 100m
- **10GBase-SR** utilizza la fibra ottica multimodale per piccole distanze (max 300m)
- **10GBase-LR** utilizza la fibra ottica su distanze fino a 10 km
- **10GBase-ER** utilizza la fibra ottica su distanze fino a 40 km
- **10GBase-CX4** utilizza quattro doppini su piccole distanze (max 15m)
- **10GBase-T** utilizza quattro doppini Cat.6a per distanze fino a 100m.

Gli standard da 1 Gbps utilizza l'encoding 8B/10B, e sono retrocompatibili con versioni più vecchie grazie all'**auto negoziamento**, ovvero adattando la velocità a quella più lenta se necessaria, e la gestione del **flow control** attraverso l'utilizzo di un frame di controllo PAUSE. Un'altra aggiunta inserita con le versioni da 1 Gbps è l'utilizzo di **jumbo frame** che possono essere più lunghi di 1500 byte, fino a 9 KB. Inoltre, supportano ancora il CSMA/CD in caso si debbano interfacciare con un hub, anche se ciò accade raramente e si utilizzano praticamente sempre gli switch a queste velocità.
Infine, le versioni da 10 Gbps+ utilizzano un encoding 64B/66B e rimuovono completamente il supporto al CSMA/CD.

Protocolli Wireless

I protocolli Wireless permettono a più dispositivi di comunicare senza utilizzare direttamente l'Internet. Ciò può avvenire mediante l'utilizzo di un **AP (Access Point)** oppure con rete **ad hoc** tra dispositivi, il primo caso è praticamente l'unico effettivamente utilizzato.
Le differenze principali tra wireless e cavo che interessano il livello MAC sono fondamentalmente due: diventa quasi impossibile ascoltare il canale per delle interferenze in quanto un segnale inviato può tornare al mittente anche un milione di volte più debole, inoltre sebbene l'AP per definizione sia nel range di tutte le stazioni, non tutte le stazioni sono necessariamente in range con le altre. Bisogna quindi utilizzare un protocollo che cerchi di minimizzare le collisioni tenendo considerando questi due aspetti.
Il frame 802.11 è strutturato come segue:

- 2 byte sono dedicati al **controllo del frame**, in particolare abbiamo
    a. 2 bit per la **versione del protocollo**, impostati sempre su 00, sono qui per permettere retrocompatibilità con future versioni di 802.11
    b. 2 bit per il **tipo** (data, control o management)
    c. 4 bit per il **sottotipo** (RTS, CTS)
    d. 1 bit per il **To DS** e 1 per il **From DS**, che indicano se il frame va o viene dal network connesso all'AP
    e. 1 bit **More fragments** per in indicare se ci sono altri frammenti successivamente
    f. 1 bit **Retry** per indicare se si tratta di una ritrasmissione
    g. 1 bit **Power Management** per indicare che il mittente sta per andare in modalità di risparmio batteria

    h. 1 bit **More data** per indicare che il mittente ha altri frame per il destinatario.
    i. 1 bit **Protected** per indicare che il corpo del frame è stato crittografato
    j. 1 bit **Order** per dire al destinatario che i livelli più alti si aspettano i frame in ordine sequenziale
- 2 byte per la **durata**, ovvero il tempo in microsecondi che il frame e il suo ACK occuperanno il canale
- 6 byte per l'indirizzo del **ricevente**
- 6 byte per l'indirizzo del **destinatario**
- 6 byte per un **terzo indirizzo**, se il frame è destinato a un endpoint presente sul network connesso all'AP
- 2 byte per la **sequenza** in modo da poter rilevare frame diplicati
- 0-2312 byte per i **dati**
- 4 bit per il **CRC**

Bluetooth

L'idea dietro il Bluetooth è quella di connettere dispositivi molto vicini in maniera wireless utilizzando apparecchiature estremamente economiche. Per fare ciò i frame Bluetooth utilizzano un gran numero di bit di ridondanza (ripetendo fino a 3 volte il contenuto e il checksum effettivi in un frame) in modo da poter essere utilizzati efficacemente in un ambiente molto rumoroso e senza hardware sofisticati.
Inoltre, a differenza di per esempio 802.11, Bluetooth specifica direttamente dei "profili" relativi alle applicazioni supportate.

Data link layer switching

Gli switch Ethernet sono un esempio di **bridge** data link, i bridge permettono di unire più LAN in un'unica macro-LAN. E' possibile inoltre divide una grande LAN in diverse sotto-LAN dette **LAN Virtuali**.

I bridge connettono più segmenti in una stessa LAN, mantenendo la suddivisione al livello data link dividendo il traffico solo tra mittente e destinatario, ciò permette una tolleranza maggiore ai malfunzionamenti e un uso più efficiente della linea.
I bridge possono essere utilizzati per connettere diverse LAN, ma devono necessariamente rispettare altre necessità, per esempio idealmente un bridge dovrebbe essere plug-and-play. Per fare ciò vengono utilizzati i **learning bridges (transparent bridges)** oppure gli **spanning-tree bridges**:

Nel primo caso lo switch o bridge mantiene una tabella delle destinazioni e delle porte associate, per costruirla inizialmente invia tutti i frame ricevuti in *broadcast*, e inizia a inserire elementi in questa tabella in base ai frame ricevuti. Per permettere alla flessibilità della LAN la tabella viene periodicamente aggiornata con il tempo di arrivo dei frame cancellando quelli più vecchi.
Per ogni frame ricevuto, il bridge consulta la tabella e:

- Se la porta di destinazione è la stessa di quella di arrivo, lo scarta;
- Se la porta di destinazione è diversa, invia il frame sulla porta apposita
- Se la porta di destinazione è sconosciuta, invia il frame a tutte le LAN tranne quella di provenienza (**flooding)**

Nel secondo caso, i bridges si configurano inviando dei frame di configurazione, questi frame sono utilizzati per creare un grafo dove i bridge sono i nodi e i link punto-a-punto sono gli archi. Lo switch costruisce un albero ricoprente utilizzando il bridge con l'indirizzo MAC più piccolo come radice, in modo da evitare i loop.
In realtà è possibile utilizzare dei learning bridge che implementano anche uno spanning-tree.

Qual è la differenza tra ripetitore e bridge?
Un ripetitore agisce sul livello fisico direttamente, copia pedissequamente tutto ciò che riceve da una linea su tutte le altre, effettuando una pulizia del segnale e una amplificazione. Mentre il bridge, acquisito un frame, lo analizza, lo ricostruisce e lo inoltra, quindi può anche essere configurato per filtrare traffico di uno specifico protocollo.

Lo switch è simile a un bridge, ha la stessa modalità di funzionamento, ma aggiunge alcune tecnologie come la **shared memory**, ovvero memorizza i pacchetti in una memoria comune a tutte le porte, una *matrice di commutazione* e ha una architettura a BUS interno condiviso ad alta velocità.
Esistono alcuni tipi di switch particolari:

- **Cut-through switching**, dove il frame è subito reindirizzato sulla porta corretta
- **Store-and-forward**, dove il frame è letto completamente dallo switch controllando il CRC e scartandolo se necessario.
- **Port-based switching** dove ad ogni porta corrisponde un solo indirizzo Ethernet
- **Segment-based switching**, dove ad ogni porta corrispondono più indirizzi

Qual è la differenza tra router e switch?
Uno switch agisce nel livello data link e non ha alcuna idea sul contenuto del pacchetto, è in grado solo di leggere l'header data link. Mentre un router modifica un pacchetto rimuovendo header e trailer e passando il contenuto del payload a un

software di routing per scegliere una linea di output. Il router non conosce l'indirizzo del frame e non sa nemmeno se il pacchetto arriva da una LAN o una linea punto-a-punto.

Le LAN Virtuali servono per dividere logicamente un'unica LAN in diverse sotto-LAN, ciò permette di evitare qualsiasi problema dovuto alla posizione fisica dei cavi. Per introdurre le VLAN è stato modificato il frame Ethernet aggiungendo un campo che la identifica e che solo i nuovi Ethernet possono riconoscere, i vecchi Ethernet non dovrebbero ricevere questi frame a prescindere.

Parte 5: Il livello Network

Il livello Network è il livello che si occupa di portare i pacchetti fino al destinatario, attraverso anche diversi router intermedi, a differenza del livello datalink che deve trasportare una sequenza di frame da un punto ad un altro attraverso un cavo (virtuale), deve inoltre in qualche modo preoccuparsi del **flow control** e della co-presenza di diverse reti.

Il livello Network si deve preoccupare oltre alla trasmissione dei pacchetti anche all'interfaccia che fornisce al livello trasporto, deve cioè mascherare tutta la topologia della rete al livello trasporto, e decidere se implementare una connessione di tipo connectionless, detta **datagram network** o connection-oriented, detta **VC (Virtual Circuit)**

Un esempio di datagram network è il **protocollo IP**, nell'implementazione di un network layer connectionless, ogni pacchetto contiene l'indirizzo completo della destinazione, e viene instradato indipendentemente dagli altri attraverso un **algoritmo di routing**. I router mantengono una **routing table** e decidono per ogni pacchetto il percorso migliore da seguire in base alla situazione della rete.

Un virtual circuit, per esempio il **MPLS (MultiProtocol Label Switching)** utilizza una connessione connection-oriented al livello network, deve quindi preoccuparsi di poter fornire una connessione tra mittente e destinatario che viene seguita da tutti i pacchetti inviati. Per fare ciò ogni router mantiene una tabella delle connessioni aperte in modo da sapere dove inviare i pacchetti al seconda del mittente.

Gli algoritmi di routing quindi sono le componenti che si occupano di gestire il percorso che i pacchetti devono effettuare, e riempiono la rispettiva routing table sul router, viceversa il processo di **forwarding** si occupa semplicemente di aprire questa tabella per decidere dove inviare i packet ricevuti.
Gli algoritmi di routing devono in qualche modo fornire diverse caratteristiche: correttezza, semplicità, robustezza (ovvero la necessità che i router si possano adattare a problemi sulla rete), stabilità (ovvero che l'algoritmo di routing sia in grado di convergere a una sola strada, e possibilmente in fretta), equità e efficienza (spesso in contrasto fra di loro).

Gli algoritmi di routing utilizzano il **sink tree**, ovvero un albero che ha come radice un router ed è composto da tutti i cammini ottimali verso quel router.
Gli algoritmi di routing possono essere **statici** o **dinamici**, ovvero possono decidere la tabella di routing a priori o riempirla a run-time in base alla situazione della rete.

Per implementare un algoritmo di routing bisogna iniziare con un modo per trovare il cammino minimo tra mittente e destinatario, questo può essere fatto, visualizzando la rete come un grafo non orientato - dove i nodi sono i router e gli archi i collegamenti -, attraverso **l'algoritmo di Djikstra per i cammini minimi** e i pesi degli archi possono essere per esempio la distanza, la bandwith, il tempo medio di arrivo di un pacchetto, o altri valori, o una combinazione tra di essi.

Un primo possibile algoritmo di routing statico è il **flooding**, dove ogni pacchetto viene inoltrato a tutti i router adiacenti. Questo metodo è molto semplice in quanto basta conoscere solo i router adiacenti, ed è il più efficiente in quanto inviando il pacchetto su tutti i collegamenti utilizzerà sempre il cammino minimo. Tuttavia in pratica non è utilizzabile da solo in quanto ogni pacchetto viene inoltrato a tutti i router in maniera infinita. Questo problema può essere risolto introducendo per esempio un hop counter o un numero di sequenza, ma rimane comunque utilizzato tipicamente solo come building block per altri algoritmi di routing.

Nel **Distance Vector Routing** ogni router registra la propria tabella (vettore) di distanza che lo separa da ogni altro router, e per ognuno di essi la linea in uscita per arrivarci. La stima della distanza viene fatta misurando il tempo di risposta di speciali **pacchetti ECHO**, e condivide la propria tabella con tutti i vicini in modo da poter ricalcolare i percorsi.
Questo algoritmo è veloce a reagire quando viene instaurato un nuovo collegamento, ma molto lento quando ne va già uno,, il problema principale è il problema **count-to-infinity**, ovvero il fatto che a lungo andare tutti i router vedono aumentare la distanza di un collegamento rimosso verso l'infinito.

Il distance vector routing è stato utilizzato fino al 1979 da ARPANET con il nome **RIP (Routing Internet Protocol)**, per essere poi rimpiazzato dal link state routing.

Il **Link State Routing** è un routing dinamico basato sullo stato dei collegamenti, in questo schema ogni router esegue alcune operazioni fondamentali in ordine:

- Scopre i suoi vicini e i loro indirizzi di rete
- Misura la distanza o costo di ogni vicino
- Costruisce un pacchetto dove memorizza tutto ciò che ha scoperto
- Condivide questo pacchetto e riceve pacchetti analoghi con i vicini
- Calcola il cammino minimo verso ogni router

Per effettuare questa sequenza di operazioni, il router invia un **pacchetto HELLO** all'avvio su tutte le linee, e riceve in risposta gli indirizzi dei propri vicini. Dopodiché invia i pacchetti ECHO e misura i tempi di arrivo della risposta e calcola il ritardo delle linee.
La costruzione del pacchetto contiene informazioni quali l'identità del mittente, il numero di sequenza del pacchetto, l'età del pacchetto e la lista dei vicini con relativi ritardi. La ricostruzione e l'invio del pacchetto può avvenire a intervalli regolari e/o quando si verifica un evento significativo (e.g. quando cade o ritorna una linea). Per la distribuzione si usa il flooding con i numeri di sequenza e l'età per evitare il reinvio infinito.
Il link state routing viene evoluto successivamente nell'**OSPF (Open Shortest Path First)** che è uno degli algoritmi più utilizzati in Internet.

Poiché una rete può arrivare ad avere un elevatissimo numero di router, diventa necessario raggruppare i router in **regioni** in modo da ridurre la grandezza della routing table e semplificare il calcolo dei cammini minimi, attraverso il cosiddetto **routing gerarchico**: Ogni router conosce solo la topologia della propria regione, o del proprio **cluster** o **zona** o altro a seconda di quanti livelli di gerarchia ci sono, e associa alle altre regioni un unico collegamento che verrà poi gestito internamente. Il vantaggio migliore lo si ha quando il numero di livelli è simile a $ln\ N$. Lo svantaggio nell'uso di una gerarchia è che il cammino minimo potrebbe non sempre portare al cammino minimo effettivo per l'instradamento di pacchetti verso altre regioni, tuttavia questo aumento è sufficientemente piccolo da essere trascurabile.

Per le trasmissioni broadcast, oltre al flooding, è possibile anche utilizzare un algoritmo detto **reverse path forwarding**, è molto simile al flooding, ma funziona in modo che, quando un pacchetto arriva al router, il router controlla se il pacchetto è arrivato sul collegamento normalmente utilizzato per inoltrare verso la fonte del

broadcast, se lo è allora c'è una grande possibilità che il pacchetto abbia usato la strada migliore, se non lo è viene scartato come probabile duplicato.

Può essere qualche volte utile inviare un pacchetto a diverse destinazione, al punto da essere inefficiente inviare tanti pacchetti singoli, e inutilmente dispendioso utilizzare il broadcast in quanto i destinatari effettivi sono comunque di gran lunga meno degli utilizzatori della rete. In questo senso è possibile sfruttare il **multicast**, che utilizza gli stessi algoritmi di routing modificando però lo spanning tree della topologia della rete per dividere i router in gruppi in modo da poter decidere a chi inviare e a chi non inviare il pacchetto.

Congestione

Nel momento in cui la rete comincia ad ingrandirsi, aumentano anche il numero di host connessi. Fino a che il numero di pacchetti in circolazione rientra nelle capacità della rete allora il **goodput** (i.e., il numero di pacchetti effettivamente utili inviati) aumenta, ma nel momento in cui questo limite viene raggiunto, molti pacchetti iniziano a venire conservati nelle memorie dei router per andare in *timeout* ed essere semplicemente inviati, creando la **congestione**.

La congestione è diversa dal *flow control*: il flow control avviene quando un mittente inonda un destinatario più lento di informazioni, la congestione avviene quando tanti host fanno circolare più pacchetti di quanti la rete ne può contenere.

Il **traffic management** può essere fatto su diversi livelli di "velocità": può essere cioè preventivo alla congestione, o fatto in procinto di una congestione o durante.
L'approccio più lento è detto **open loop** (senza controreazione), l'approccio più veloce è detto **close loop** (con controreazione).
Il modo più diretto e lento è quello di costruire un network con abbastanza banda per sopperire alle necessità, detto **approviggionamento**.
Un altro meccanismo è quello del **routing traffic-aware**, ovvero la decisione dell'instradamento a seconda di pattern di traffico di rete durante lo scorrimento della giornata.
Il problema principale di questo meccanismo è che se la quantità di traffico ha un peso elevato nella decisione dell'instradamento, tutti i router cambieranno continuamente strada andando avanti e indietro tra due zone.
L'**admission control** indica invece un meccanismo usato ampiamente nei circuiti virtuali con l'idea di non inizializzare nuovi circuiti se il network non lo può sopportare.

Un altro meccanismo è quello del **load shedding**, ovvero dello scartare alcuni pacchetti in base a dei fattori come ad esempio la priorità, e l'origine (durante il trasferimento di un file sarebbe per esempio inutile mantenere i pacchetti da 7 a 10 dopo aver scartato un pacchetto numero 6, mentre nel traffico di streaming real-time sarebbe altrettanto inutile mantenere un pacchetto per troppo tempo)

Il meccanismo del **traffic shaping** indica invece la regolamentazione del rate medio e burstiness dei dati che entrano nella rete. Questo può essere fatto in tre modi:

- L'algoritmo **leaky bucket** ricorda fondamentalmente l'idea di un secchio riempito da un rubinetto (di cui si può regolare l'apertura) con un piccolo buco sul fondo che riversa l'acqua a ritmo costante. Se viene messa troppa acqua essa esce dal bordo superiore e si perde.
- L'algoritmo **token bucket** è simile al leaky bucket ma consente un grado di irregolarità controllato anche sul flusso che esce sulla rete, in questo modo un host che non trasmette accumula credito trasmissivo fino ad un certo data rate massimo consentito, quando invece ha dati da trasmettere sfrutta se necessario tutto il credito disponibile.
- **Flow specification**, ovvero l'accordo tra sorgente, subnet e destinatario riguardo le caratteristiche del traffico e la qualità del servizio,

Altri meccanismi includono l'utilizzo di **choke packet**, ovvero un pacchetto che un router invia a un host d'origine per invitarlo a diminuire il flusso, oppure gli **hop-by-hop choke packet**, dove i router sul percorso rallentano tutti il proprio flusso se ricevono un choke packet.

La **qualità del servizio (QoS, Quality of Service)**, è altresi importante a seconda del tipo di trasmissione. In particolare per assicurare una QoS bisogna rispondere a 4 problematiche:

- Cosa l'applicazione richiede dalla rete
- Come regolare il traffico che entra nella rete
- Come predisporre abbastanza risorse ai router per garantire le prestazioni
- Se la rete può o meno accettare altro traffico

Nessun meccanismo da solo può affrontare tutte allo stesso momento, pertanto viene implementato un diverso range di tecniche sia al livello network che al livello trasporto.

Il **packet scheduling** indica l'ordine in cui i pacchetti che arrivano vengono immessi sulle linee in uscita, il sistema più semplice è ovviamente di tipo FIFO, ma soffre del problema che un mittente che invia tanti dati potrebbe essere privilegiato, per questo

motivo è possibile implementare il **Fair Queeing** dove i pacchetti vengono divisi in diverse code scansionate in maniera round-robin, oppure una versione migliorata detta **WFQ (Weighted Fair Queueing)** dove le code ora hanno diverso peso. Ciò viene fatto per evitare che i mittenti che inviano pacchetti di dimensioni più elevate ottengano una quantità di banda maggiore.

I sistemi di QoS possono essere costruiti come **servizi integrati** o **servizi differenziati**. I servizi integrati sono costruiti creando delle linee nel momento in cui devono essere effettuate trasmissioni unicast o multicast, ma sono difficili da implementare in quanto dovrebbe essere instaurata una nuova linea per milioni di utenti. I servizi differenziati dividono invece i pacchetti in **classi** di diversa priorità, e il router divide le code di uscita in diverse code divise in priorità diverse.

Internetworking

Esistono diverse tipologie di reti che devono essere connesse: WANs, MANs, LANs, etc.., che formano un blocco fortemente eterogeneo, diventa quindi in qualche modo necessario poter connettere reti con diversi protocolli e funzionamento: per esempio una rete potrebbe offrire un servizio connectionless e un'altra connection-oriented, l'indirizzamento potrebbe essere diverso, la grandezza dei pacchetti diversa, la QoS diversa, etc..

Un primo meccanismo di internetworking è il **tunneling**, che è in grado di connettere due reti uguali collegate però da una rete diversa: il tunneling funziona impacchettando il pacchetto inviato (che utilizza il protocollo dei due host) utilizzando il protocollo della rete intermedia, in questo modo il pacchetto attraversa un tunnel come se fosse trasportato in una scatola.
Questo meccanismo è facile da implementare ed è molto usato per connettere host e network isolati attraverso network diversi (e.g. VPN), ma presenta il problema che non è possibile connettersi a host presenti sulla rete intermedia.

Il routing attraverso network diversi presenta quindi diversi problemi, per esempio due network potrebbero utilizzare diversi criteri per decretare i cammini minimi.
Per risolvere queste problematiche si utilizzano algoritmi di routing a due livelli: un internetwork è in genere composto da singole reti dette **AS (Autonomous System)**, si esegue quindi un routing interno o **intradomain**, detto **IGP (Interior Gateway Protocol)**, ovvero il routing utilizzato dentro un singolo network, e **interdomain**, ovvero il routing utilizzato per attraversare le diversi reti, detto **EGP (Exterior Gateway Protocol).**

Per supportare la differente grandezza dei pacchetti, è possibile agire in due modi: La grandezza minima che un pacchetto deve avere per attraversare un cammino è detta **Path MTU (Maximum Transmission Unit)**, tuttavia non è possibile sapere a priori quale sia l'MTU di un cammino.
La soluzione alternativa è quella di dividere il pacchetto in frammenti nel momento in cui il pacchetto raggiunge un router con un packet size consentito inferiore della grandezza del pacchetto effettiva. La frammentazione può avvenire semplicemente dividendo il pacchetto in frammenti che sono ordinati dall'header IP.
La frammentazione però presenta comunque dei problemi, in particolare le performance possono degradare a causa dell'header overhead e del fatto che la perdita di un frammento può portare alla perdita dell'intero pacchetto.
La soluzione utilizzata in realtà è quella del **path MTU discovery**: nell'header del pacchetto IP si inserisce un bit per indicare che non deve essere effettuata alcuna frammentazione, quando un router riceve un pacchetto troppo largo ed esamina questo bit, invia un pacchetto di errore al mittente che lo osserva e diminuisce la grandezza dei pacchetti da inviare, e li reinvia. Se riceve un ulteriore errore allora effettua nuovamente l'operazione fino a quando il pacchetto non è stato inviato. In questo modo la frammentazione viene gestita interamente tra gli host mittente e destinatario.

INTERNET PROTOCOL

L'Internet può essere visto come un insieme di network connessi a delle **backbones**, le backbones più grandi sono dette **Tier 1 Networks.** La colla che tiene insieme tutti diversi network è il livello network e il suo protocollo, che nell'Internet è **IP (Internet Protocol).**

Il protocollo IP è definito da una versione, il più comune è **IP Version 4** ma con l'aumentare dei dispositivi nel mondo è diventato necessario, con **IP Version 6** supportare un numero più elevato di dispositivi. Ogni pacchetto IPv4 è composto da un **header** e un **payload** o **body**.
L'header è composto da una parte fissa di 20 byte e una parte variabile, ed è formato da parole di 4 byte:

- I primi 4 bit sono usati per la **versione**, l'inclusione di questo valore all'inizio del datagram permette la transizione da una versione all'altra su un lungo periodo di tempo.
- I successivi 4 bit sono detti **IHL**, e indicano quanto è lungo l'header in termini di 4 byte words. Il valore minimo è 5, nel caso in cui non vi siano opzioni e

solo la parte fissa di 20 byte, il valore massimo è 15 per un totale massimo di 20 byte fissi e 40 byte di opzioni.
- 8 bit sono utilizzati per i **servizi differenziati.** Inizialmente questo campo serviva a indicare il tipo di servizio previsto, ora viene diviso in 6 bit sfruttati dai *differentiated services* e 2 bit per indicare se il pacchetto è stato soggetto a **congestione** durante il tragitto.
- 16 bit per la **lunghezza totale**, ovvero compresa sia di header che dati, e che quindi può essere massimo di 65.536 byte.
- 16 bit per il campo **identificazione**, utilizzato per determinate a quale pacchetto un frammento appartiene
- 1 bit inutilizzato
- 1 bit **DF (Don't Fragment)**, per indicare che il pacchetto non deve essere frammentato (utilizzato nel path MTU discovery)
- 1 bit **MF (More Fragments)** per indicare che un pacchetto non è l'ultimo frammento.
- 13 bit per un **fragment offset**, che indica dove appartiene il frammento rispetto al pacchetto completo, per un massimo di 8192 frammenti.
- 8 bit per il **Time to live**, per evitare che pacchetti girino all'infinito in caso di errori di routing. Inizialmente era pensato per contare i secondi, ma ora usa semplicemente il numero di hop
- 8 bit per il **Protocollo** (TCP, UDP, etc..)
- 16 bit per il **checksum** dell'header
- 1 word per il **source address,** e 1 word per il **destination address**
- Opzionali 40 byte per eventuali opzioni: *Security* (totalmente inutilizzato in quanto poco sicuro), *Strict source routing*, ovvero la strada completa da seguire, *Loose source routing*, ovvero dei router da attraversare necessariamente, *Record route*, utilizzato in teoria per memorizzare la strada intera, non è utilizzato in pratica per mancanza di spazio effettivo, e *Timestamp*, simile a record route ma invece di segnare l'indirizzo ogni router inserisce il tempo.

L'indirizzo IP a 32 bit identifica univocamente un dispositivo sulla rete, e sono rappresentati da 4 byte, ovvero 4 numeri decimali separati da punti.
Ogno indirizzo è composto da un **indirizzo di rete** o **prefisso**, e da un **indirizzo di host**. Talvolta i prefissi sono descritti semplicemente dalla loro lunghezza preceduta da uno /, e quando scritti in questo modo, costituiscono una **subnet mask**, ovvero una sequenza di 1 lungha tanto quanto la lunghezza del prefisso, seguita da quanti 0 necessari per arrivare a 32 bit. Quindi per esempio un indirizzo del tipo 200.2.23.53/8, utilizza 8 bit per il network, e ha come subnet mask 255.0.0.0, ovvero 8 bit 1 e 24 bit

0. Facendo **l'AND** tra la subnet mask e l'indirizzo e possibile ottenere l'indirizzo di rete.
Esistono alcuni indirizzi "noti", per esempio 0.0.0.0 indica *questo computer* ed è utilizzato in fase di boot, l'indirizzo formato dal prefisso+tutti 0 identifica un rete, l'indirizzo 1.1.1.1 indica un broadcast locale sulla propria rete, mentre l'indirizzo formato da un prefisso e tutti 1 indica un broadcast diretto su rete remota.
L'indirizzo 127.xx.yy.zz è detto indirizzo di **loopback**.

Può essere necessario per una organizzazione dividere il proprio indirizzo in sottogruppi, o subnetwork, questa operazione prende il nome di **subnetting** e i network ottenuti sono detti **subnets**. Per esempio se una organizzazione ha un indirizzo di rete /16, essa internamente può dividere i propri indirizzi in blocchi, che possono essere /17, /18, etc..

Poiché gli indirizzi nel mondo possono essere tantissimi, e poiché ogni ogni router deve effettuare ricerche continue sulle proprie tabelle di routing, diventa in qualche modo necessario velocizzare questa operazione.
Un primo approccio è stato quello di divdere gli indirizzi in classi, A, B, C e D (e altri indirizzi "riservati" di classe E). A ogni associazione o gruppo viene associata una classe di indirizzi, e il lavoro dei router viene semplificato dovendo conoscere solo l'indirizzo di rete di una classe. Il problema principale di questo metodo è che moltissimi indirizzi vengono sprecati.
Il metodo attualmente utilizzato è detto **CIDR (Classless InterDomain Routing)** e funziona in maniera opposta al subnetting, attraverso il meccanismo di **route aggregation**. Questo metodo funziona in modo che indirizzi vicini vengono condensati in base all'indirizzo comune da un router, per i router esterni basta conoscere solo il prefisso piuttosto che tutti i singoli indirizzi che lo compongono, sarà poi il router comune ad effettuare il routing interno.
Per evitare lo spreco di indirizzi, inoltre, i prefissi possono sovrapporsi, con la regola che i pacchetti vengono inviati all'indirizzo più specifico, ovvero si considera l'indirizzo con **prefisso più lungo**.

Infine, un altro meccanismo per risparmiare indirizzi è il **NAT (Network Address Translation)**, e funziona associando ad associazioni o abitazioni un unico indirizzo globale o al più un numero ristretto di indirizzi, viene poi collocato un **NAT box** che riceve tutti i pacchetti e che si occupa di far uscire i pacchetti e consegnare i pacchetti esterni all'host effettivo. Per fare ciò però deve ricorda i numeri di porte in uscita (e per fare questo, deve esaminare i payload TCP e/o UDP) ed effettuare una apposita traduzione degli indirizzi.

Gli indirizzi interni hanno subnet mask 10.255.255.255/8, 172.31.255.255/12 oppure 192.168.255.255/16 a seconda del NAT.
Il problema principale è che quindi il NAT viola diversi aspetti del livello network e della rete in generale, ma è oggi largamente utilizzato anche considerando il fatto che nel NAT box vengono spesso implementati anche sistemi di firewall e privacy.

IP Version 6 nasce succesivamente per rimpiazzare IPv4 dato il numero molto limitato di indirizzi IPv4 nel mondo rispetto alle necessità. IPv6 è stato realizzato cercando di rimuovere o scoraggiare feature alla fine non utilizzate in IPv4, cercando di velocizzare il lavoro dei router e ridurre ulteriormente la grandezza delle routing tables e supportare maggiormente diversi tipi di servizio.
L'header IPv6 è formato come IPv4 da 4-byte words, per un totale di 40 byte, ed è composto come segue:

- 4 bit per la **versione**, come in IPv4
- 8 bit per i **servizi differenziati**, come in IPv4, anche nell'utilizzo dei due bit per gestire la congestione
- 20 bit per il **flow label** utilizato da fonte e destinatario per creare una pseudo-connessione, instruisce ovvero la rete di trattare tutti i pacchetti allo stesso modo.
- 16 bit per la **lunghezza del payload**, non di tutto il pacchetto in quanto l'header ha lunghezza fissa
- 8 bit per il **next header**, che può puntare a un qualsiasi *header opzionale*, oppure, se questo è l'ultimo header, instruisce il network layer a chi gestore del transport layer dare il pacchetto (TCP, UDP, etc..)
- 8 bit per **l'hop limit**, equivalente al time to live di IPv4
- 16 byte per **l'indirizzo sorgente**
- 16 byte per **l'indirizzo destinatario**
- Eventuali header opzionali, detti **extension headers**, tra cui l'hop-by-hop header, utilizzato per implementare i jumbogram, ovvero particolari pacchetti molto molto grandi utilizzati principalmente nel supercomputing, il destination header, ovvero eventuali opzioni di destinazione riservate principalmente al futuro, un header di routing, simile al loose source routing dell'IPv4, un header per supportare la frammentazione, e due header dedicati alla sicurezza: autenticazione e payload crittografato di sicurezza

Gli indirizzi IPv6 sono formati da 8 gruppi di 8 byte, scritti in esadecimali, quindi 8 gruppi di 4 lettere esadecimali, collegati dai :.
Per esempio 8000:0000:0000:0000:0123:4567:89AB:CDEF, ma poiché era attendibile avere molti zero in quanto tantissimi indirizzi sono inutilizzati, gli zeri iniziali sono

eliminabili (0123 è equivalente a 123), inoltre se vi sono molteplici gruppi di 4 zeri questi possono essere riassunti in doppi due punti.
L'indirizzo di sopra diventa quindi 8000::123:4567:89AB:CDEF.
Gli indirizzi IPv4 possono essere scritti in IPv6, per esempio ::192.31.20.46, ma attenzione che IPv4 **non** è direttamente compatibile con IPv6. Tuttavia IPv6 è compatibile con moltissimi servizi presenti (TCP, UDP, etc..)

Oltre a IPv4, nell'Internet sono presenti altri protocolli "compagni" utilizzati, tra cui ICMP, ARP e DHCP e le loro versioni analoghe per IPv6, ICMPv6, ARPv6 e NDP (Neighbor Discovery Protocol)

ICMP (Internet Control Message Protocol)

L'ICMP è la notifica di un evento tramite un pacchetto che permette ai router di monitorare costantemente le operazioni nell'Internet. Esistono una dozzina di messaggi ICMP, tra cui:

- **Destination unreachable**, che si verifica quando il router non riesce a trovare la destinazione o quando un pacchetto con il *DF bit* settato non può essere consegnato perché un network con grandezza massima più piccola del pacchetto è sulla via
- **Time exceeded**, quando il TTL di un pacchetto scende a 0, usato per esempio per implementare l'utility *traceroute*
- **Parameter problem**, quando un field nell'header è invalido, per esempio a causa di un errore nel software del mittente
- **Source quench**, utilizzato inizialmente per dire al mittente di rallentare in quanto il network è in una situazione di congestione, ora poco utilizzato in quanto aggiunge solo altro carburante al fuoco
- **Redirect**, utilizzato quando un router nota che un pacchetto sembra seguire una route incorretta, per notificare il router mittente di scegliere una route migliore
- **Echo & Echo reply**, utilizzati per controllare se una destinazione è raggiungibile e in funzione
- **Timestamp request/reply**, stessa cosa di Echo, ma con i timestamp
- **Router advertisement/solicitation**, utilizzato per trovare un router vicino

e altri messaggi.

ARP (Address Resolution Protocol)

ARP è il protocollo utilizzato per associare gli indirizzi IP agli indirizzi MAC.
Di base l'assocazione può essere statica, attraverso la registrazione di una tabella aggiornata periodicamente, ma questo metodo è poco flessibile e macchinoso, viene quindi utilizzato ARP che la gestisce interamente:
Il mittente invia una richiesta ARP con l'IP di interesse in broadcast, la macchina con quell'IP risponde con il proprio indirizzo MAC, a questo punto il mittente può creare il frame data link e inviarlo. Per velocizzare l'operazione i dispositivi possono salvare in cache questa mappa IP-MAC per un breve periodo di tempo.
Se i due dispositivi sono sulla stessa rete, si parla di **consegna diretta**, se i dispositivi sono su reti diverse è necessario inviare il pacchetto al gateway (router predefinito, quello con l'indirizzo host più basso a parità di indirizzo di rete) ed effettuare la **consegna indiretta**.

DHCP (Dynamic Host Configuration Protocol)

DHCP viene utilizzato per assegnare automaticamente gli indirizzi IP alle macchine che lo richiedono, attraverso un server di configurazione. Questo server potrebbe essere situato sulla stessa rete o su un'altra rete.
Quando una macchina viene attivata, essa contiene solo il proprio indirizzo MAC incluso nel NIC, invia quindi un broadcast di richiesta di IP, attraverso un pacchetto **DHCP DISCOVER**, quando questo pacchetto arriva al server DHCP, questo alloca un indirizzo IP libero e invia all'host un pacchetto **DHCP OFFER** con tale indirizzo.
Per evitare che gli indirizzi rimangano bloccati per sempre anche quando una macchina non li utilizza più, si usa una tecnica chiamata **leasing**, ovvero appena prima che questo *lease* scada, l'host deve inviare un pacchetto al server DHCP chiedendo un rinnovo. Se il pacchetto non viene inviato o la richiesta è rifiutata l'indirizzo IP viene ritirato.

MPLS (MultiProtocol Label Switching)

MPLS è simile al circuit switching, è un layer intermedio tra il livello data link e il livello network che marchia ogni pacchetto con un label al fine di velocizzare notevolmente il routing e il forwarding dei pacchetti.
Ogni pacchetto IP viene coperto da un header MPLS sopra all'header IP, prima di essere coperto dell'header data link (per esempio PPP). L'header MPLS è di 4 byte e contiene informazioni tra cui il campo **Label**, un campo **QoS** e un campo **TTL.**
MPLS è multiprotocol, ovvero nasce con l'intenzione di trasportare pacchetti IP sopra reti non IP, utilizzando router chiamati **LSR (Label Switched Router)** che effettuano

il forwarding del pacchetto MPLS e router **LER (Label Edge Router)** ovvero i router di confine di MPLS che aggiungo e rimuovono l'header MPLS ispezionando il pacchetto IP contenuto.

Il routing nell'Internet avviene su due livelli: l'Internet è formato da un certo numero di reti indipendenti, dette **ASes (Autonomous Systems)**, il routing avviene sia internamente (intradomain) che esternamente (interdomain).

OSPF (Open Shortest Path First)

OSPF è utilizzato nell'Internet per il routing intradomain, utilizzando un grafo pesato da vari fattori (distanza, delay, etc..). Il routing viene effettuato utilizzando una tecnica di load balacing detta **ECMP (Equal Cost MultiPath)**, ovvero, nel momento in cui vi sono due strade equivalenti, OSPF divide il traffico tra le due per bilanciare il carico. Un AS internamente potrebbe essere diviso in diverse aree, un router potrebbe anche non appartenere a nessuna area, ma i router dentro un'area sono detti **router interni**. I router interni devono conoscere solo la topologia interna dell'area, e si interfacciano con i **router di confine dell'area** per inviare pacchetti esternamente all'area. Ogni AS ha inoltre una **backbone area**, chiamata area 0, a cui tutte le aree sono connesse. L'ultimo tipo di router è il **router di confine dell'AS**. A questi router arrivano tutte le route esterne all'AS.

OSPF funziona attraverso lo scambio di informazioni tra **router adiacenti**, non ogni router con ogni router, identificando un particolare router all'interno di una LAN che è connesso a tutti gli altri router della LAN.
Lo scambio di informazioni può avvenire attraverso alcuni messaggi OSPF:

- **Hello**, quando si avvia un router per scoprire chi sono i vicini
- **Link state update**, da al mittente il costo verso i suoi vicini
- **Link state ack**, acknoweledge per un link state update
- **Database description**, annuncia quale update possiede il mittente
- **Link state request**, richiede informazioni a un altro router

BGP (Border Gateway Protocol)

BGP è il protocollo interdomain per decidere il routing dei pacchetti tra diversi AS. Il problema di trasferire dati tra gli AS è che non è più sufficiente calcolare i cammini

minimi (cosa solitamente nemmeno possibile, in quanto ogni AS tende a nascondere la propria rete interna), ma bisogna tenere conto di policy e scelte dei diversi AS. Ogni AS deve quindi poter essere in grado di impostare delle policy in modo da favorire delle route piuttosto che altre.
Il BGP viene diviso in due varianti, l'uso regolare tra gli AS è detto **eBGP (external BGP)**, ma vi è un'altra varianta detta **iBGP (internal BGP)** utilizzata per propagare le route da un lato di un ISP all'altro.
Non esistono delle regole di BGP precise in quanto ogni AS utilizza le proprie policy, ma in generale i router seguono queste euristiche, in ordine:

1. Si preferisce la route con la preferenza locale maggiore
2. Si preferisce la route con la più corta **AS path** (ovvero il numero di AS da attraversare)
3. Si preferiscono le route ottenute da connessioni esterne (eBGP), piuttosto che internet (iBGP)
4. Tra le route ottenute da AS vicini, si preferiscono le route di uscita più facile
5. Si preferiscono le route con il minor costo di hop tra un indirizzo e l'altro

Parte 6: Il livello Trasporto

Il livello trasporto fa da colla essenziale tra il livello network e il livello applicazione, permette cioè di astrarre la scrittura delle applicazioni da tutti i problemi del livello network, in modo che i programmi possano quasi assumere che l'invio e la ricezione di byte sia esente da problemi.
Il livello trasporto è gestito in locale da una **transport entity**, che ha il compito di lavorare sull'header di trasporto, cioè sui **segmenti** (chiamati in passato **TDPU - Transport Protocol Data Unit**)
Il livello trasporto può essere connection-oriented e connectionless indipendentemente da ciò che viene fatto a livello network, anche se avrebbe poco senso setuppare una connessione a livello rete per un datagram connectionless del livello trasporto.
Il livello trasporto deve garantire delle **primitive** di servizio che possono essere implementate in diversi modi (nel sistema operativo, librerie, in hardware, etc..), e funzionano grossomodo con il server che invoca una qualche funzione per rimanere in ascolto su un indirizzo, il client che si connette, delle primitive per inviare e ricevere messaggi e una eventuali disconnessione.

Quando un processo applicazione vuole stabilire una connessione su un processo di una applicazione remota, deve specificare a **quale** processo connettersi, per fare ciò nell'Internet si utilizzano degli endpoint dette **porte.** Il termine generico è **TSAP**

(Transport Service Access Point), analogo agli **NSAP (Network Service Access Point,** per esempio gli indirizzi IP**).** Gli TSAP sono utilizzati per identificare i singoli processi su di un host, processi noti, per esempio una applicazione mail o web server, possono utilizzare delle porte note, quando non si utilizzano dei servizi noti viene invece utilizzato un **portmapper** che converte una stringa ASCII in un indirizzo di porta specifico sulla macchina.

Per stabilire una connessione, è necessario in qualche modo affrontare il problema dei pacchetti in ritardo o duplicati. Una prima soluzione è stata quella di utilizzare un meccanismo di clock in modo che un numero di sequenza venga utilizzato per rilevare i pacchetti duplicati, ma questo da solo non risolve il problema di una connection request duplicata.
La soluzione effettiva è il **Three-Way Handshake**, che funziona in questo modo: L'host 1 invia una richiesta di connessione con numero di sequenza *x*, l'host 2 risponde con un ACK per il numero di sequenza *x*, e annunciando il proprio numero di sequenza *y*, dopodiché l'host 1 invia un ACK per il numero di sequenza *y*.
Se arriva un pacchetto duplicato, l'host 2 invierà un ACK per accettare la connessione, ma nel momento in cui questo ACK raggiunge l'host 1 che non voleva aprire una connessione, questo verrà scartato e non risposto con un ulteriore ACK. In questo modo si evitano problemi dovuti a pacchetti duplicati o in ritardo.

La **terminazione della connessione** non è una questione altrettanto semplice, in teoria si potrebbe pensare semplicemente di inviare degli ACK per assicurarsi che entrambi gli host siano d'accordo sulla disconnnessione, ma in pratica è possibile dimostrare non solo che un three-way handshake non è abbastanza, ma che un qualsiasi numero di ACK risulterà in un protocollo il cui ultimo ACK è o essenziale o non essenziale: un ACK non essenziale può essere rimosso dal protocollo, ma se l'ACK essenziale viene perso il protocollo non è affidabile. (Problema dei due eserciti)
La soluzione generale è quella di introdurre dei timer: ogni host ha un timer interno per la connessione che resetta ogni volta riceve un segmento, se non riceve segmenti per un lungo periodo di tempo alla scadenza del timer proverà a inviare una fila di segmenti per assicurarsi che l'altro host sia ancora attivo, in caso negativo termina la connessione.
In generale però la gestione di chiusura della connessione è molto complicata e dipende anche dal livello applicazione, per esempio un web server potrebbe semplicemente terminare l'applicazione bruscamente.

Il problema dell'**error control** e **flow control** è gestito concettualmente similmente al livello data link, ma sono da tenere in considerazione le differenze di funzione e grado. Per esempio il checksum protegge un segmente mentre attraversa una interna

rete, mentre nel livello data link protegge un frame su una singola linea. Il controllo al livello trasporto è essenziale per la correttezza, mentre quello al livello data link per l'efficienza.
La gestione del flow control è più particolare, viene utilizzato un protocollo a *sliding window* simile al livello data link, ma bisogna fare attenzione a limitare la grandezza della finestra in relazione alla rete, che oltre a gestire il flow control gestisce inerentemente anche la congestione.

Altre possibilità del livello trasporto sono il **multiplexing** (implementato da SCTP) e il **crash recovery** che è una questione molto delicata e difficile da affrontare in maniera semplice. Il controllo della congestione deve tenere conto sia di efficienza che fairness, e può essere implementato utilizzando feedback dal livello network che possono essere espliciti o impliciti.

UDP

Il **protocollo UDP (User Datagram Protocol)** è un un protocollo al livello trasporto **connectionless,** che non si occupa di garantire tra le tante il flow control, congestion control, reliability ed essendo connectionless non crea una connessione.
Il vantaggio di non garantire questi aspetti è che UDP è concettualmente e praticamente più semplice, e fa da collante tra il livello network e il livello applicazione tramite i numeri di porta, in questo modo è possibile gestire queste problematiche al livello applicazione se necessario.
UDP trasmette segmenti che consistono in un **header di 8 byte** seguito dal payload.
L'header UDP è formato da:

- 8 byte per la **porta sorgente**
- 8 byte per la **porta destinazione**
- 8 byte per la **lunghezza**, che indica sia l'header che il payload, è può essere al minimo 8 byte (solo l'header) o al massimo *65,515* byte
- 8 byte per **l'UDP checksum**

UDP è utilizzato per implementare le **RPC (Remote Procedure Call)**. L'idea delle RPC è quella di permettere a un host di utilizzare una procedura presente su un host remoto come se fosse sull'host stesso. Per fare ciò si utilizano delle procedure di libreria dette **client stub** sul client e **server stub** sul server, e gestiscono la RPC come segue:

1. Il client chiama il client stub come una procedura normale
2. Il client stub impacchetta i parametri in un messaggio e fa una system call per inviare il messaggio. L'impacchettamento è detto **marshaling**

3. Il sistema operativo invia il messaggio da client a server
4. Il sistema operativo del destinatario passa il pacchetto al server stub
5. Il server stub chiama la procedura con i parametri *unmarshaled*

Il problema principale di RPC è la gestione dei parametri delle procedure, che può essere molto problematico, ma è molto utilizzato in pratica implementado delle restrizioni.

UDP fornisce anche la base per le applicazioni multimedia in real-time, attraverso **RTP (Real-time Transport Protocol)** che si trova a metà tra il livello trasporto e applicazione. (E' implementato a livello applicazione ma agisce come se fosse un livello trasporto).
RTP si occupa di gestire i segmenti delle applicazioni multimedia e di inviarli all'applicazione che si occupa poi di farli funzionare anche considerando segmenti perso o in ritardo.
L'header RTP è formato come segue:

- 2 bit per la **versione**
- 1 bit di **padding** e 1 bit **X** per indicare se è presente un extension header
- 4 bit per indicare quante **contributing sources** ci sono
- 1 **marker bit** che è application-specific
- 8 bit per il **payload type** che indica il tipo di encoding effettuato
- 16 bit pr il **numero di sequenza**, che viene incrementato con ogni pacchetto RTP.
- 32 bit per il **timestamp** utilizzato per ridurre il jitter
- 32 bit per il **synchronization source identifier** che dice a quale stream appartiene il pacchetto

Oltre RTP è stato definito un protocollo simile detto **RTCP (Real-time Transport Control Protocol)** che si occupa della sincronizzazione, feedback e user interface.

TCP

Il **protocollo TCP (Transmission Control Protocol)** è stato designato in maniera differente a UDP per provvedere un byte stream reliable attraverso un internetwork unreliable, ovvero deve essere un protocollo robusto e si deve adattare dinamicamente in base alle diverse topologie, grandezze di banda, delay e altri parametri dei diversi network.

Nel protocollo TCP le TSAP sono le **porte**, e le connessioni avvengono end-to-end (TCP non supporta multicast per esempio) tra mittente e destinatario, identificati da indirizzo IP e numero di porta.
Le porte sotto alla 1024 sono le **well-known ports**, ovvero delle porte riconosciute a cui si possono attaccare solo processi di root. In UNIX viene avviato un processo demone (**inetd, Internet Daemon)** per gestire le porte: nel momento in cui un processo si vuole connettere a una porta, inetd fa una fork per creare un nuovo processo demone che gestisce quella porta.

L'header TCP è formato da un fixed header di 20 byte e alcuni header opzionali, l'header di 20 byte è formato come segue:

- 16 bit per la **porta sorgente**
- 16 bit per la **porta destinazione**
- 32 bit per il **numero di sequenza**
- 32 bit per il **numero di acknowledge**
- 4 bit per la **lunghezza dell'header TCP**, utilizzato in caso vi siano header opzionali
- *4 bit non utilizzati*
- 8 bit per le **opzioni,** ovvero flag di 1 bit *(vedi sotto)*
- 16 bit per la **window size**
- 16 bit per il **checksum**
- 16 bit per **l'urgent pointer**, utilizzato in caso di arrivo di segmenti "urgenti", ma non è attualmente molto utilizzato
- Eventuali opzioni, per esempio la **MSS (Maximum Segment Size)** che un host è disposto ad accettare, la **window scale** option per negoziare una window scale più grande, la **timestamp** option per utilizzare i tempi di invio, e la **SACK (Selective ACKnowledgement)** utilizzato per notificare esplicitamente quali dati sono arrivati e quali vanno ritrasmessi

Gli 8 flag permettono di aggiungere alcune opzioni o marcare tipologie di segmenti, tra cui:

1. **CWR (Congestion Window Reduced)**, utilizzato dal mittente verso il destinatario per notificare che ha ricevuto l'ECE
2. **ECE** ovvero **ECN (Explicit Congestion Notification) - Echo** per notificare la congestione
3. **URG** settato a 1 quando è utilizzato l'urgent pointer
4. **ACK** se settato a 1 indica che l'acknowledgment number è valido e quindi indica un acknowledge.
5. **PSH** indica al ricevente di inviare immediatamente i dati all'applicazione senza bufferarli

6. **RST** utilizzato per resettare una connessione, un RST solitamente indica un qualche problema
7. **SYN** utilizzato per stabilire una connessione
8. **FIN** utilizzato per terminare una connessione

Per stabilire una connessione TCP utilizza l'**handshake a tre vie**: un host invia un segmento con SYN = 1 e ACK = 0 per indicare di voler stabilire una connessione, il numero di sequenza è un certo valore *x*, l'altro host risponde con un segmento con un numero di sequenza *y*, con SYN = 1 e ACK = 1 e l'acknowledge number a *x+1*, infine, dopo aver ricevuto questo segmento il primo host invia un ultimo segmento con SYN = 1 e ACK = 1, numero di sequenza *x+1* e numero di acknowledge *y+1*.
Se il l'entità TCP riscontra l'assenza di un processo in attesa sulla porta manda un RST di rifiuto.

La terminazione della connessione può essere vista come la terminazione di due connessioni simplex (TCP è full duplex in realtà) ovvero terminando singolarmente le due direzioni con FIN e i rispettivi acknowledge.

TCP utilizza la sliding window per gestire la connessione, con alcuni approcci per rendere il protocollo più efficiente:
L'approccio detto **delayed acknowledgements** (ACK ritardati) è un metodo che permette a TCP di attendere un certo tempo sperando che vi siano dei dati da spedire, in modo da poter effettuare il piggybacking.

Un altro metodo utilizzato è **l'algoritmo di Nagle**, che funziona quando i dati arrivano in piccoli blocchi: il mittente invia solo il primo blocco e inizia a bufferare tutti i blocchi successivi fino a quando non riceve l'ACK del primo blocco, a questo punto invia tutti i blocchi bufferati e buffera i successivi.

Un problema che può degradare le prestazioni è il **silly window syndrome**, quando una applicazione interattiva da parte del destinatario legge solo 1 byte alla volta e inviare un ACK dicendo di accettare un altro solo byte. Questo ovviamente è molto inefficiente, e per ovviare a questo problema si utilizza un limite minimo per l'aggiornamento della sliding window, ovvero il ricevente non può aggiornare la sliding window fino a quando il numero di byte da accettare è almeno grande quanto la MSS scelta durante l'inizializzazione della connessione o fino a quando il buffer è mezzo vuoto.

TCP utilizza diversi timer, il più importante è il **RTO (Retransmission TimeOut),** un timer utilizzato quando non arriva un ACK per un segmento inviato entro un tempo

stabilito, in quel caso si effettua la trasmissione. Il problema del RTO è che il delay della connessione può variare drasticamente, inoltre se il timer è troppo piccolo si fanno ritrasmissioni inutili, mentre se è troppo elevato si avranno ritardi di trasmissione, bisogna quindi addattarlo utilizzando algoritmi di stima del miglior timeout basato misurando il **RTT (Round-Trip Time)**
Altri timer sono:
Il **persistence timer**, utilizzato per evitare deadlock quando il ricevente invia un ack con grandezza della finestra pari a 0, in quel caso il mittente invia un segmento di probe per verificare che la grandezza della finestra sia ancora 0.
Il **keepalive timer**, utilizzato dopo un grande lasso di tempo senza ricevere segmenti se l'altra parte della connessione è ancora attiva
Il timer utilizzato nella **TIME WAIT**, ovvero durante la terminazione prima di chiudersi effettivamente si attende un timer pari a due volte il massimo periodo di vita dei pacchetti in modo da assicurarsi che tutti i pacchetti creati siano spariti.

La congestione viene gestita utilizzando la **congestion window**: all'apertura della connessione viene inviato un piccolo numero di segmenti contemporaneamente, per esempio 4, dopodiché si utilizza il meccanismo di **slow start,** ovvero ogni volta che vengono ricevuti tutti gli ACK per i segmenti inviati, si raddoppia il numero di segmenti. Appena un segmento viene perso, si imposta come **threshold** della connessione un numero pari alla metà della grandezza della congestion window quando il pacchetto è stato perso (si può dire con abbastanza certezza che un pacchetto è stato perso se vengono ricevuti tre ACK duplicati), dopodiché si imposta la congestion window di nuovo al numero iniziale, ma questa volta si aumenta a ogni ACK la grandezza di 1 (quindi non raddoppiando).
Un approccio simile e più veloce è quello di ripartire semplicemente dal threshold stesso.

Parte 7: Il livello Applicazione

Il **DNS (Domain Name System)** è il meccanismo che permette di associare gli indirizzi numerici ai nomi facilmente leggibili dei siti, questo è utile non solo in quanto è più facile per gli utenti ricordare i nomi, ma anche perchè un server potrebbe essere collocato su diversi indirizzi IP, oppure il suo indirizzo IP potrebbe cambiare molto frequentemente.

La ricerca di un indirizzo avviene in maniera ricorsiva a partire da un processo generato dall'applicazione, detto **stub resolver**, lo stub resolver invia una query a un

local DNS resolver, detto **local recursive resolver** o più semplicemente **local resolver**.
Tipicamente query e risposte sono inviate come pacchetti UDP: lo stub invia la query completa al local resolver, e il local resolver effettuare delle ricerce ricorsive verso i name server responsabili di un particolare dominio, detti **authorative name server**.
Il local resolver riceve le risposte parziali dai name server, e poi invia la risposta completa allo stub resolver. Lo stub resolver NON riceve risposte parziali.

Ogni indirizzo DNS è composto da parole divise da punti, e un punto alla fine che indica la root. La gerarchia degli indirizzi è formata da una serie di indirizzi **top-level** divisi in **gTLD (generic Top Level Domain) e ccTLD (country code Top Level Domain)**. I gTLD sono indirizzi di tipo internazionale, mentre i ccTLD vengono definiti per ogni nazione. Questi indirizzi sono gestiti dall'**ICANN (Internet Corporation for Assigned Names and Numbers)**, anche se dal 2011 è stata data la disponibilità alle aziende di creare gTLD addizionali.
I domini top-level sono operati da compagnie dette **registries**, un livello sotto nella gerarchia, i **registrar** vendono i domain names direttamente agli utenti.
I generale, per ottenere un dominio di secondo livello basta chiedere a un registrar, ottenuto un dominio è possibile creare qualsiasi tipo di sottodominio.

Le query DNS in realtà di base accedono a una generica risorsa, per esempio oltre agli indirizzi possono essere utilizzate per creare una blacklist degli indirizzi.
Un **resource record** prende la forma di:
Domain_name Time_to_live Class Type Value

Il **Domain Name** indica il dominio al quale il reocrd si applica.
Il **Time To Live** da una indicazione su quanto "stabile" il record sia, per esempio un indirizzo che viene molto raramente cambiato potrebbe avere un time to live più alto in modo da lasciare il record per più tempo in cache, mentre informazioni più volatili dureranno meno.
La **classe** indica se l'informazioni è una **IN** (**Internet Information)** o meno. Altri codici possono essere usati ma ciò è raramente necessario.
Il **tipo** indica il tipo di record, ne esistono diversi tra cui **A** per gli indirizzi IPv4, **AAAA** per gli indirizzi IPv6, **CNAME** per gli alias, **NS** che indica il name server per il dominio o sottodominio, **MX** che indica il dominio disposto ad accettare email, **TXT** che indica un testo ASCII e **SOA** (**Start Of Authority)** che provvede la fonte primaria di informazioni di un name server.

La risoluzione di un nome DNS avviene come segue:

Il local resolve invia una query al **root name server**, che non è detto conosca l'indirizzo completo, ma sicuramente conosce l'indirizzo di livello più alto (com, edu, etc..) e ritorna il nome e l'indirizzo IP per il name server di quella parte, il local resolver a questo punto effettua un'altra query a quel name server e ottiene il name server del sottodominio richiesto. La query continua fino all'ultimo sottodominio (o se un name service intermedio conosce l'indirizzo completo) e ritorna l'indirizzo IP appropriato.

Le **email** o **electronic mail** funzionano attraverso una semplice architettura: l'utente utilizza sul proprio computer un programma detto **User Agent**, questo programma interagisce con un **message transfer agent** per inviare o ricevere le mail alla propria **mailbox**. L'operazione di invio di una mail è detta **mail submission**.

Una email è formata da un messaggio composto da **header** e **body**, racchiuso in un **envelope.** L'header contiene le informazioni del mittente e l'oggetto del messaggio, l'envolope invece indica le informazioni del destinatario.

Inizialmente le mail consistevano in semplici segmenti di testo in ASCII, questo sistema molto velocemente è diventato obsoleto, e con il tempo è stato sviluppato **MIME (Multipurpose Internet Mail Extensions)**, che è costruito sopra allo standard originale (ovvero se MIME non è supportato il messaggio viene inviato come ASCII normale). MIME permette la trasmissioni di dati binari, file multimediali e consente ulteriori metodi di encoding.

Il trasferimento effettivo della mail avviene tramite il protocollo **SMTP (Simple Mail Transfer Protocol),** che funziona con un processo dedicato alla gestione della mail in ascolto sulla porta 25 del computer. SMTP è un semplice protocollo ASCII, e i comandi sono formati da 4 caratteri. Il problema dell'SMTP è che non supporta l'autenticazione e la sicurezza dei messaggi, per questo motivo è stato successivamente sviluppato **ESMTP (Extended SMTP)** che contiene diverse estensioni per consentire nuove funzionalità.

Dopo aver inviato la mail al transfer agent apposito, il transfer agent non utilizza SMTP per consegnare la mail allo user agent, ma utilizza un altro protocollo detto **IMAP (Internet Message Access Protocol)** oppure in alternativa un **sistema webmail**.

ATTENZIONE!!!

Non ho riassunto la parte su streaming multimedia e VoIP, sono le slide 16, 17 e 18.

Peer-to-Peer (P2P)

Il modello Peer-to-Peer si differenzia dal modello Client/Server in quanto ogni Peer ha funzionalità largamente simile.
Il modello P2P è nato a metà anni 2000 con l'intento di far ritornare il contenuto, la scelta e il controllo agli utenti. P2P permette di creare community e scambiare informazioni tra minuscoli endpoint nell'Internet senza nemmeno doversi conoscere.

Il modello P2P si basa su una rete di nodi con capabilità e responsabilità equivalenti, i nodi sono sia server che client **"Servents",** e costituiscono una rete effimera che utilizzano per collaborare. I peer connessi costituiscono una rete sovrapposta al di sopra dell'infrastruttura sottostante.
Le reti sovrapposte sono grandi gruppi di connessioni logiche tra gli end host, e possono essere **strutturate** oppure **non strutturate**. Gli overlays sono tutti presenti nel livello applicazione.
Gli obiettivi del modell P2P sono:

- Riduzione dei **costi** condividendo il costo stesso.
- Grande **scalabilità** e **affidabilità**
- **Interoperabilità** attraverso l'aggregazione di risorse diverse
- **Maggiore autonomia**
- **Anonimità/Privacy**, che è difficile da ottenere quando il server è centrale
- **Dinamismo e comunicazioni ad hoc**. Nel P2P le risorse entrano e lasciano il sistema continuamente, quindi le reti P2P non possono affidarsi a infrastrutture esistenti.

I sistemi P2P possono essere classificati in base al grado di decentralizzazione:

- L'**Hybrid decentralized P2P** sfrutta un server centrale per facilitare l'interazione tra i peer. Il server effettua delle ricerche e identifica i nodi nel network. Gli svantaggi principali sono la presenza di un singolo punto di fallimento e la scalabilità (Esempi: Napster)
- La **Purely decentralized P2P** si basa sui servents che effettuano le medesime mansioni senza una coordinazione centrale. Gli svantaggi principali sono la possibile mancanza di consistenza dei dati, l'overhead di comunicazione e la sicurezza. (Esempi: Gnutella inizialmente, Freenet)

- Il **Partially centralized P2P** utilizza alcuni nodi con ruoli più importanti, che agiscono come indici centrali, detti **supernodi**. (Esempi: Kazaa, Gnutella recentemenete)

oppure al grado di strutturazione P2P:

- **Unstructured P2P**: i dati sono distribuiti casualmente tra i peer e meccanismi di broadcast sono utilizzati per la ricerca. Il piazzamento dei dati non è correlato alla topologie dell'overlay. (Esempi: Napster, Gnutella, Kazaa)
- Nello **Structured P2P** la topologia della rete è strettamente controllata e i file sono piazzati in posti specifiche, provvedendo un mapping tra gli identificatori dei file e le posizioni. (Esempi: Chord, CAN, Past, Tapestry)
- **Loosely Structured P2P**: si piazza a metà tra structured e unstructured, le posizioni dei file sono effettuate attraverso delle *routing hints***,** ma non sono completamente specificate.

Il file sharing è la killer application del P2P, in quanto le aree di scambio sono potenzialmente illimitate e lo scambio è totalmente anonimo.
P2P può essere utilizzato anche per l'instant messaging, conferenze audio/video, applicazioni condivise come per esempio giochi o powerpoint condivisi.
La computazione distribuita è un altro applicativo importante di P2P data la grande scalabilità ottenuta combinando un gran numero di PC.
P2P può anche essere utilizzato per i database distribuendo i nodi che contengono i dati.

La ricerca in una rete P2P deve risolvere diversi problemi, tra cui un possibile bottleneck della grandezza di banda, la necessità autonomia a livello di nodo, la robustezza, la scalabilità, l'espressività e la completezza.
La ricerca può essere effettuata in maniera **blind** oppure **informata**.

La blind search può essere effettuata con diversi metodi. Per esempio è possibile utilizza la **BFS con flooding**, ovvero inoltrando le query a tutti i vicini. Il flooding è ristretto dal TTl dei pacchetti.
E' possibile utilizzare una **BFS modificata**, che utilizza il flooding solo su una porzione dei vicini per ridurre i costi di traffico.
Altri metodi sono la **random walk**, che inoltra la query a un numero casuale di vicini, e **l'iterative deepening** che utilizza BFS consecutive con profondità aumentata.

La ricerca informata può essere effettuata attraverso un local index, utilizzando metodi **proattivi** o **reattivi**.
Tra i metodi proattivi vi sono per esempio il **Neighborhood Signature Search**, dove ogni nodo mantiene un sommario delle keywords mantenute nei peer in un raggio

vicino, oppure la **Associative Search** dove i nodi sono raggrupati in “gruppi di interesse” e le query utilizzano il flooding in uno o più gruppi di interesse.
Tra i metodi reattivi vi sono la **DRLP (Distributed Resource Location Protocol)** che usa una random walk quando non vi sono informazioni disponibili, oppure **l’intelligent BFS.**

La sicurezza nelle reti P2P deve affrontare la presenza di Peer con intenti maliziosi, bisogna quindi provedere a diversi livelli di sicurezza.
La sicurezza dello storage viene affrontata utilizzando dati auto-certificanti, dispersione delle informazioni e SHA.
Il routing sicuro si basa su tre primitive: assegnazione sicura degli ID ai nodi, manutenzione sicura delle tabelle di routing, e inoltro sicuro dei messaggi.

Approcci DHT
Una **DHT (Distributed Hash Table)** è una struttura dati decentralizzata utilizzata per il routing delle query, ogni nodo mantiene del contenuto particolare (o una reference a quel contenuto) e supporta una funziona di routing per aiutare la ricerca.
Esistono diversi modi per realizzare la DHT:

- **Consistent hashing**: la rete sovrapposta è un cerchio, ogni nodo ha un id casuale, il successo è un nodo con il prossimo id più alto, e la chiave è mantenuta nel successo può vicino. Ogni nodo mantiene almeno due successori, quando il successore va via, chiede a quello successivo la lista dei suoi successori. Quando un nuovo nodo entra, trova il suo predecessore e il suo successore opportunamente.
- **Chord** utilizza un overlay circolare, con un ID monodimensionale nello spazio di hash. Il range coperto è]previous_id, own_id]. Un nodo è raggiungibile da un qualsiasi altro nodo in non più di log2(N) hops nell’overlay.
- **CAN** utilizza un hypercubo come geomtria di routing, un valore hash è un punto nel piano cartesiano a D dimensioni, e ogni nodo è responsabile di un cubo D-dimensionale. I vicini sono i nodi che tocccano in più di un punto. L’aggiunta di un nuovo nodo utilizza un procedura ricorsiva per trovare i vicini, rappresentando il cubo come un albero.
- **Tapestry** utilizza una routing mesh dove i nodi adiacenti condividono un prefisso ID.
- **Pastry** è simile a Tapestry con alcune differenze sul processo di routing.
- **Kademlia** distribuisce gli indici tra i nodi che sono trattati come foglie di un albero binario. Il routing avviene esplorando l’albero in base ai prefissi degli ID.

Il file sharing nelle reti P2P utilizza un sistema basato su reputazione, ovvero in base al numero di file condivisi, ma che soprattutto siano genuini.

Parte ∞: Domande

DOMANDE:

- Spiegare il modello ISO/OSI
- Quali sono i protocolli nel livello rete?
- Quali sono le classi di indirizzi?
- Come funziona il protocollo go-back-n?
- Quali sono le altri architetture di rete oltre a client/server?
- Qual è la differenza tra carrier-sense e no-carrier-sense?

—-

- Spiegare il modello TCP/IP
- Quale altro protocollo è di tipo best effort oltre a UDP?
- Come fa il livello data link a capire se un frame è corretto o meno?
- Come funziona la codifica Manchester?
- Cosa sono i bit di parità?
- Come è possibile controllare un burst di errori?
- Quali sono le topologie di rete?
- Spiegare il funzionamento del protocollo Token Ring
- Cos'è il Piggybacking?

—-

- Spiegare il funzionamento del P2P
- Che protocollo trasporto utilizza il P2P?
- Quali sono gli applicativi P2P per lo streaming video/audio? *(risposta: skype)*
- Come funziona Skype? Cos'è l'interlacciamento?
- Cos'è il sottolivello MAC?
- Come funziona il CSMA/CD?
- Spiegare il funzionamento del protocollo Token Ring
- Come funziona la modulazione per ampiezza?
- Cos'è l'FDM?
- Cos'è il CRC? Come funziona?
- Come è possibile controllare un burst di errori?
- Come si dividono i bit in frame al livello data link?
- Come funziona la codifica 4B/5B? E la codifica Manchester?

—

- Cos'è la NAT?

- Quali vantaggi comporta la NAT? *(risposta: risparmia indirizzi IP e permette di implementare un firewall)*
- Come avviene il routing sulla rete?
- Come fa un router a scoprire i router vicini?
- Chi si occupa di far comunicare reti diverse tra di loro?
- Come funziona il protocollo selective-repeat?
- Come funziona il Three-way handshake?
- Cos'è il Constellation Pattern?

—

- Cos'è il TDM?
- Cos'è l'HDLC?
- Come funziona Internet Protocol?
- Cos'è la subnet mask?
- Come fa un router a capire se un pacchetto deve essere inviato dentro o fuori la propria rete?
- Come funziona lo streaming video?
- Cos'è un overlay network?
- Come funziona BitTorrent?
- Spiegare il funzionamento del P2P
- Cos'è il Signal-to-noise ratio?
- Come è possibile rilevare gli errori?
- Cos'è la distanza di Hamming
- E' meglio correggere o rilevare gli errori? Perché?
- Se il RTT è molto alto, quale protocollo data link conviene utilizzare?
- Che protocollo trasporto utilizza il DNS?
- Cosa accade quando una query DNS fallisce?

—

- Di che tipo è una query DNS? *(risposta: ricorsiva)*
- Come funziona il protocollo FTP?
- Cos'è il sottolivello MAC?
- Quali sono i protocolli carrier-sense?
- Come è fatto uno switch?
- Spiegare il funzionamento del P2P
- Qual è la differenza tra commutazione di pacchetto e commutazione di circuito?
- Cos'è il Signal-to-noise ratio?
- Quanta informazione può passare attraverso un canale? Come si calcola il data rate? *(per la risposta basta guardare i teoremi di Nyquist e di Shannon)*
- Cos'è una rete broadcast?
- Quali sono le tipologie di rete?

- Quali sono le topologie di rete?

—

- Spiegare il funzionamento del P2P
- Spiegare TCP e UDP, come funzionano e differenze
- Quali sono i protocolli data link?
- Cos'è ICMP?
- Quali sono le differenze tra i modelli ISO/OSI e TCP/IP?
- Come si fa a stabilire quando un bit è 1 o 0 al livello fisico?
- Cos'è il Signal-to-noise ratio
- Cos'è il jittering in un sistema multimediale?
- Cos'è il bit stuffing
- Cos'è la VLSM (Variable Length Subnet Mask)?
- Come funzionano FDM, TDM e WDM?

—

- Cos'è il flooding?
- Cosa succede quando vi è un problema di congestione? Quali sono e come funzionano gli algoritmi per evitare la congestione?
- Come è possibile controllare un burst di errori?
- Come funziona VoIP?
- I pacchetti UDP arrivano sempre? *(questa domanda è un po' un trabocchetto, la risposta è che i pacchetti UDP potrebbero non arrivare se filtrati dal firewall)*
- Cos'è ICMP?
- Qual è la differenza tra carrier-sense e no-carrier-sense?
- Cos'è il CRC? Come funziona?
- Come funziona la codifica Manchester?
- Cos'è il DNS?

—

- Come funziona il protocollo FTP?
- Quali sono le differenze tra i modelli ISO/OSI e TCP/IP?
- Come si dividono i bit in frame al livello data link?
- Come funziona il protocollo selective-repeat?
- Cos'è il Piggybacking?
- Quanta informazione può passare attraverso un canale? Come si calcola il data rate?
- Come funziona la modulazione per ampiezza?
- Cos'è il Constellation Pattern?
- Come è fatto uno switch?
- Cos'è la VLSM?

- Qual è la differenza tra commutazione di pacchetto e commutazione di circuito?
- Chi si occupa di far comunicare reti diverse tra di loro?
- Cos’è il CRC? Come funziona?
- Cos’è l’interlacciamento? Quali sono i problemi affrontati in un sistema multimediale?