ELEMENTI INTRODUTTIVI

Nonostante questo esame si chiami reti di calcolatori, c'è un piccolo equivoco, perché, al momento, non possiamo parlare di questo concetto. Non più oramai. In un tempo non molto lontano avevamo grossi mainframe che comunicavano tra loro attraverso linee dedicate. Poi, però, le dimensioni dei calcolatori sono andate sempre di più a diminuire, c'era bisogno che essi fossero comunicanti tra loro. Per questo, nacque il concetto di rete, che poi è stato anche esteso non più ai calcolatori, ma anche a degli oggetti concreti, il cosiddetto **IOT**, *Internet of Things*. Oggi abbiamo oltre venti miliardi di dispositivi in rete ed è un numero che andrà sempre a crescere in maniera esponenziale e costante. Le reti hanno radicalmente cambiato le nostre vite, impattando sul nostro modo di comunicare, imparare, lavorare, giocare e hanno influenzato molto la nostra sfera affettiva, abbattendo le distanze e facilitando le relazioni sociali. Molti definiscono una rete come un ponte ideale che serve a congiungere spazio e tempo. La rete è fatta, fondamentalmente, di **connessioni**. Una connessione può viaggiare attraverso l'etere, quindi usando le onde, oppure attraverso la fibra ottica, con grossi cavi che si trovano maggiormente sott'acqua.
Spesso, più reti vengono unite fino a formare un'unica grande rete interconnessa. L'esempio più lampante? Ovvio, **Internet**. Internet è una rete di calcolatori che connette miliardi di dispositivi. Noi ci andremo a concentrare sui vari **componenti** di una rete, per poi soffermarci sui **protocolli**, la **struttura** e i **modelli** di una rete, andando a studiare semplici algoritmi e logiche di instradamento.
Una rete ha diversi scopi. Deve garantire l'**accesso alla informazioni**, la **condivisione delle risorse** e deve, ovviamente, **facilitare la comunicazione**. Quest'ultima deve essere **sicura**, **affidabile**, **efficiente** e **scalabile**, in grado di connettere ambienti applicativi diversi. Una connessione può essere composta da solo due nodi (**punto-a-punto**), oppure da più di due nodi (**multipunto**). Nel primo caso potremo avere problemi di **comunicazione fisica**, quindi malfunzionamenti elettrici, ottici o problemi di onde elettromagnetiche. Nel secondo, invece, nascono molti altri problemi, come l'**indirizzamento** (corretta trasmissione), l'**instradamento** (corretto destinatario), la **commutazione** (inversione dei collegamenti di un circuito), e l'**alta affidabilità**, che

deve essere necessariamente garantita.
Un possibile modello fisico di una rete deve garantire la presenza:

- **hosts** (**stazioni**): dispositivi autonomi connessi ad una rete.
- **links** (**collegamenti trasmissivi**): tipicamente punto-a-punto, come abbiamo già detto, che sono connessi tra loro tramite nodi di commutazione.
- **nodi di commutazione** (**network-switch**): il cui compito è quello di rendere la comunicazione efficiente, riconoscendo le richieste di apertura, i dati e il relativo nodo di destinazione.

All'opposto delle reti che abbiamo appena descritto, si posizionano la **reti broadcast**. Esse hanno un unico canale comunicazione che è condiviso da tutti gli elaboratori. Le reti broadcast si inviano brevi messaggi, che sono spesso detti **pacchetti**. Ogni elaboratore è identificato da un indirizzo. Quando un elaboratore riceve un pacchetto, esamina l'indirizzo di destinazione e, se coincide col proprio indirizzo, il pacchetto viene elaborato. È anche possibile inviare un pacchetto ad ogni elaboratore, secondo una tecnica detta **broadcasting**. Oppure, è possibile inviare il pacchetto ad un sottoinsieme di elaboratori, attraverso la tecnica del **multicasting**.
Possiamo avere diversi flussi di trasmissione, cioè come i dati viaggiano attraverso una linea di comunicazione. Ne vediamo alcuni:

- **simplex**: i dati viaggiano in un'unica direzione.
- **half-duplex**: i dati viaggiano in entrambe le direzioni, ma non possono farlo contemporaneamente. I dispositivi che usano questo meccanismo sono detti **terminali conversazionali**, che prevedono l'invio di una richiesta, la ricezione della risposta, e l'invio di un'ulteriore risposta.
- **full-duplex**: i dati viaggiano contemporaneamente in entrambe le direzioni.

Prima abbiamo accennato il concetto di **commutazion**e. La commutazione è quell'operazione che predispone il percorso che le informazioni del mittente devono seguire per raggiungere il destinatario, attraverso dispositivi intermedi detti commutatori. Abbiamo due tipi di commutazione:

- **commutazione di circuito**: c'è un insieme di commutatori connessi fisicamente e partizionati in più canali logici di comunicazione. Generalmente, è prevista una fase di instaurazione di connessione, dove il nodo A spedisce la richiesta al nodo B. B accetta la richiesta, in modo che A possa spedire i dati correttamente. I dati

viaggiano attraverso la rete di commutatori. Una volta ricevuti, A chiude la connessione, fa richiesta di chiusura e la spedisce a B.

- **commutazione a pacchetto**: il messaggio viene diviso in più unità autonome, che sono dette pacchetti, ciascuna con le proprie informazioni di controllo. In questo meccanismo c'è la prenotazione delle risorse, che vengono divise in base ad un meccanismo FIFO.

Possiamo dire che la commutazione a pacchetto è più efficiente, scalabile e riesce ad ottimizzare l'uso delle risorse. Ovviamente, è inadeguata se bisogna consegnare grandi quantità di informazioni.
Una caratteristica della rete riguarda il modo in cui può avvenire il **colloquio** tra due nodi, con la prima che è tipica della commutazione a circuito e la seconda di quella a pacchetto.

- ***connection-oriented-mode*** (**orientato alla connessione**): i due nodi, prima di effettuare lo scambio, si assicurano di essere presenti sulla stessa linea. Dopodiché, viene instaurata la connessione, che è generalmente gestita da un software comune, che regola i dettagli dello scambio.
- ***connectionless-mode*** (**senza connessione**): un nodo può inviare un messaggio ad un altro anche se esso non è presente sulla linea. Non necessita di servizi di controllo esterni

Per quanto la seconda opzione sia più efficiente e affidabile, non c'è un modo per correggere gli errori. La soluzione a questo problema ritiene che sia necessario affidare il controllo degli errori direttamente alle applicazioni, in modo da alleggerire i protocolli di linea.
Un altro concetto importante è la **topologia di rete**, cioè la configurazione geometrica dei collegamenti tra i vari componenti di rete:

- **rete gerarchica o ad albero**: il traffico dei dati va dal livello più basso verso i nodi intermedi o, semplicemente, verso il nodo più alto. Si tratta di una topologia molto efficiente, perché permette la cooperazione tra i nodi. Si può affidare un compito semplice al nodo più basso, che è anche il meno potente, mentre possiamo affidare al nodo più alto il compito più difficile, data la sua estrema potenza. Un malfunzionamento al nodo più alto, però, tipo un sovraccarico, potrebbero compromettere l'intera struttura, per questo si tende ad appesantirla con operazioni di back-up.

- **rete a stella**: si tratta di una configurazione simile a quella ad albero, soltanto che non esistono differenti livelli. Esiste un nodo centrale che realizza tutte le funzioni degli altri nodi. Ci sono ovviamente, le stesse problematica della rete gerarchica.
- **rete dorsale o bus condiviso**: si tratta della configurazione adottata per le reti ethernet. C'è un unico cavo che collega tutti i nodi. La trasmissione di un nodo viene ricevuta da tutti gli altri. Questa tecnologia è la stessa usata per le periferiche bus, cioè l'insieme dei cavi elettrici che mettono in comunicazione tutti i dispositivi. È necessario implementare un meccanismo di arbitraggio, in modo da evitare che due nodi entrino in conflitto quando vogliono trasmettere contemporaneamente. L'accesso è sicuramente molto semplice, ma ci sono altri svantaggi importanti, per esempio un malfunzionamento di un nodo metterebbe fuori causa l'intera rete e, inoltre, la mancanza di punti di concentrazione rende difficile l'individuazione di un malfunzionamento.
- **rete ad anello**: si tratta della configurazione adottata per le reti LAN (Local Area Network), che poi approfondiremo. La trasmissione è unidirezionale, cioè i dati viaggiano nella stessa direzione ma, essendo un circuito chiuso, si può inviare un messaggio da un qualsiasi nodo all'altro. Anche qui è necessario un meccanismo di arbitraggio.
- **rete a maglia**: ogni nodo è collegato con diversi circuiti. Il traffico è partizionato e questo garantisce efficienza. I costi, però, sono molto elevati e la gestione della struttura è piuttosto complessa.

Un **protocollo** è una serie di norme, convenzioni e tecniche per lo scambio di dati, comandi e informazioni di controllo tra due elementi. Una volta individuato il nodo di destinazione, bisogna stabilire quale strada prendere per connetterlo al nodo sorgente. Questa scelta compete al **protocollo di instradamento**. A questo si aggiunge il **protocollo di linea**, che passa i dati su ciascuna linea. Quest'ultimo, però, garantisce affidabilità solo ad un nodo intermedio, cioè si preoccupa soltanto che la trasmissione avvenga correttamente. Ad occuparsi della consegna è il **protocollo di trasporto**. Per farla breve, il protocollo di linea agisce sulle singole tratte, quindi ***box-to-box***, mentre il protocollo di trasporto è di tipo ***end-to-end***.
In informatica, il termine **standard** serve ad indicare delle specifiche prestabilite da una qualsiasi autorità. Possono essere ***de facto***, cioè imposte nella pratica e implementate all'interno del sistema operativo, oppure ***de jure***, cioè proposti e approvati da qualche

organizzazione conosciuta. Ricordiamo alcuni standard conosciuti, anche se non sarà necessario impararli tutti per l'esame:

- **ISO** (*International Standard Organization*): ente generico di standardizzazione internazionale.
- **IEEE** (*Institute of Electrical and Electronic Engineers*): organizzazione mondiale di ingegneri elettrici ed elettronici che fornisce numerosi standard anche sulle reti.

Le reti possono essere identificate anche dalla loro copertura geografica. Prima di identificarne qualcuna, esse si dividono in due macrogruppi. Il primo comprende le reti che interconnettono processori all'interno dello stesso computer, mentre il secondo interconnette elaboratori diversi:

- **PAN** (*Personal Area Network*): usata di solito quando due dispositivi hanno bisogno di comunicare e scambiarsi messaggi.
- **LAN** (*Local Area Network*): si crea all'interno di aree circoscritte, cioè non più ampie di qualche chilometro, quindi case, uffici o, al massimo, una scuola.
- **MAN** (*Metropolitan Area Network*): questa rete ha un'estensione tipicamente urbana ed è pubblica. Vale a dire, un'azienda qualsiasi, mette a disposizione la rete a chiunque la desideri, a meno che paghi una tariffa prestabilita.
- **WAN** (*Wide Area Network*): si estendono a livello di una nazione, di un continente e, eventualmente, anche del pianeta Terra.

Quando diversi tipi di rete si collegano, parliamo di **interwork**. Qui potremo dire che non c'è bisogno di collegarle, dato che basterebbe usare la WAN ed estendere il collegamento per grandi distanze. Spesso, però, risulta necessario collegare delle reti che sono progettate in maniera diversa. Per questo ricorriamo a specifiche apparecchiature, dette **gateway** che si occupano dell'instradamento e della traduzione delle scelte progettuali.
Le reti non esisterebbero senza il **modello ISO/OSI**, che definisce una struttura alla base delle reti. All'utente finale, spesso, la comunicazione risulta banale, ma deve prima passare attraverso vari livelli, ai quali sono assegnati funzioni specifiche. Vediamoli:

- **Livello 1 - Fisico**: si spostano il bit su un dispositivo analogico che permette la trasmissione (cavo elettrico o fibra). Si specifica come inviare l'informazione, il voltaggio, l'intensità di corrente e il tipo di interfaccia.

- **Livello 2 - Data Link Layer**: i bit vengono trasformati in pacchetti che gestiscono numerose problematiche e permettono un facile accesso.
- **Livello 3 - Network Layer**: si prende carico della destinazione dei pacchetti e si occupa dei meccanismi di routing, scelta ottimale del percorso, e di conversione, cioè la mediazione tra reti di tipi diversi.
- **Livello 4 - Transport Layer**: garantisce il corretto trasporto di informazioni. Si tratta del primo livello di tipo end-to-end, cioè che ha un destinatario e un mittente. Stabilisce, mantiene e termina una connessione.
- **Livello 5 - Session Layer**: fa dialogare la componente utente con i servizi di rete, interfacciando i livelli superiori con quelli specifici della rete.
- **Livello 6 - Presentation Layer**: trasforma i dati forniti dalle applicazioni in un formato standardizzato e offre servizi comuni di comunicazione, tipo crittografia e compressione del testo.
- **Livello 7 - Application Layer**: si trovano servizi applicativi, come il DNS, la posta elettronica e la navigazione sul web.

Ciascuno di questi livelli ha bisogno di implementare specifici protocolli che permettano comunicazione tra essi.

LIVELLO FISICO

Prima di iniziare a parlare di livello fisico, chiariamo un attimo il concetto di **segnale**. I segnali sono variazioni di grandezze fisiche che trasportano determinate informazioni. Essi possono essere acustici, elettrici, magnetici, ecc. Noi ci concentreremo, ovviamente, sui **segnali elettrici**. Una delle principali funzioni del livello fisico, è quella di trasportare i dati in forma di segnali elettrici su un mezzo trasmissivo. Una prima distinzione dei segnali, può essere scritta in questo modo:

- **segnali analogici**: al variare del tempo, possono assumere tutti i valori compresi fra il massimo e il minimo consentito dal canale di comunicazione.
- **segnali digitali**: possono assumere solo un numero discreto di valori, di solito o 0 o 1.

A loro volta, segnali analogici e digitali, possono assumere diverse due forme:

- **segnali periodici**: si ripetono con regolarità nel tempo. Il tempo necessario affinché il segnale si ripeta, è detto **periodo**, mentre la ripetizione del segnale all'interno del periodo, è detta **ciclo**.
- **segnali aperiodici**: cambiano senza esibire regolarità nel tempo.

Adesso, ci tocca scendere ulteriormente nel dettaglio. Ci concentreremo maggiormente sui segnali analogici periodici, che possono essere **semplici** o **composti**. Semplice significa composto da un'**onda sinusoidale**, composto da più onde sinusoidali. L'onda sinusoidale è la forma più importante di un segnale analogico periodico. Si tratta di una curva oscillante, che ricorda molto la funzione seno. Essa è, di solito, rappresentata da tre parametri. Il primo è l'**ampiezza massima**. È il valore assoluto del segnale nella sua intensità massima ed è proporzionale all'energia trasportata dal segnale. Si misura, generalmente, in **volt**. Il secondo parametro è la **frequenza**. È il numero di periodi in un secondo. Generalmente, la frequenza è l'inverso del periodo e viceversa. È espressa in **hertz**. Di solito, la frequenza viene intesa come misura per la velocità con cui il segnale cambia rispetto al tempo. Il terzo parametro è la **fase**, che descrive la posizione dell'onda sinusoidale rispetto al tempo 0. Essa viene misurata in **gradi**.
Una caratteristica importante di un segnale, è la lunghezza d'onda. Essa mette in

relazione la frequenza di un'onda sinusoidale con la velocità di propagazione del mezzo che trasporta il segnale. Nei moderni sistemi di comunicazione, si tende a lasciar perdere i segnali semplici, per favorire i segnali composti. La Serie di Fourier ci dimostra che è possibile rappresentare un segnale composto come la somma di onde sinusoidali semplici, con diverse ampiezze, frequenze e fasi:

$$f(x) = a_0 + \sum_{k=1}^{\infty} (a_k \cos k \cdot x + b_k \sin k \cdot x)$$

I parametri a_0, a_k e b_k sono detti **coefficienti di Fournier** e dipendono dal valore della funzione $f(x)$. Con un'opportuna configurazione di parametri, la serie rappresenta le funzioni che hanno la capacità di essere periodiche. Con questa rappresentazione possiamo dire con certezza che un segnale periodico è equivalente ad una somma di funzioni sinusoidali, dette **armoniche**, aventi una propria ampiezza e una propria frequenza. L'insieme delle frequenze che descrivono il segnale è detto **banda di frequenza** o **spettro**. Ogni canale trasmissivo si comporta come una specie di **filtro passa-banda**, cioè non permette il passaggio di tutte le componenti. Per questo viene definita una **larghezza di banda**, cioè l'insieme delle frequenze che il canale fa passare. Affinché un segnale sia ricevuto, è necessario che la banda passante sia uguale o più ampia della banda di frequenza. Altrimenti, il segnale si deteriorerà. A proposito di ciò, ci tocca definire un attimo come un segnale può deteriorarsi e perché. Un segnale si deteriora a causa del mezzo trasmissivo su cui viaggia. Abbiamo tre tipi di **deterioramento**. Il primo è l'**attenuazione**: è generalmente una perdita di energia, che il segnale perde per superare la resistenza del mezzo trasmissivo. Una parte dell'energia viene, quindi, convertita in calore. Essa è misurata in **decibel**. Il secondo è la **distorsione**: è un cambiamento di forma del segnale. Si verifica, generalmente, con i segnali composti, dove le frequenze sono di più e viaggiano ad una velocità di propagazione diversa. Il terzo tipo di deterioramento è il **rumore**: è una forma di energia indesiderata, che si somma al segnale, degradandone il contenuto informativo. Il rumore, a sua volta, si divide in diversi tipi:

- **rumore bianco**: lo spettro di questo rumore interferisce con l'energia a tutte le frequenze dello spettro elettromagnetico. Esso è equamente distribuito.
- **rumore di intermodulazione**: rumore prodotto dalla non-linearità dei dispositivi, che presenta la presenza di armoniche indesiderate, che non sono state include dal

segnale di ingresso.

- **rumore di modo comune**: rumore presente in ingresso ad un qualsiasi strumento di misurazione.
- **rumore di quantizzazione**: perdita di informazione che si ha quando si trasforma un segnale analogico in digitale.
- **rumore termico**: rumore dovuto all'agitazione termica degli elettroni presenti in una resistenza.

Il rumore è un concetto abbastanza importante. Anche perché, da esso, dipende anche la velocità per il trasferimento dati, oltre che la larghezza di banda e il numero di livelli del segnale. Per calcolare la velocità massima di trasferimenti, sono stati formulati due criteri teorici, uno per un canale senza rumore e un altro con rumore:

- **criterio di Nyquist** (**senza rumore**): possiamo definire la formula come segue. $V = 2 * L * log_2(L)$ dove V indica la velocità in bit per secondo ed è definita da L, che è il numero di livelli del segnale. Come vediamo, aumentando i livelli del segnale, maggiore è la velocità di trasferimento.
- **criterio di Shannon** (**con rumore**): possiamo definire anche qui, la formula come segue. $C = L * log_2(1 + SNR)$. Qui, la velocità viene misurata in termini di capacità del mezzo trasmissivo e, come possiamo vedere, dipende dalla larghezza di banda L e dal rapporto segnale-rumore SNR, cioè il rapporto tra la potenza media del segnale e la potenza media del rumore.

È possibile misurare le prestazioni di una rete valutando diversi criteri:

- **larghezza di banda**: essa può essere misurata in **hertz**, la lunghezza delle frequenze, in **bps**, cioè la velocità con la quale possiamo spedire bit. Collegando entrambe, è possibile ottenere una buona valutazione.
- **throughput**: misura quanto velocemente possiamo spedire i dati su una rete.
- **latenza**: misura quanto tempo occorre a trasferire un intero messaggio, valutando la **trasmissione**, il tempo per emettere dati, la **propagazione**, il tempo per propagarsi lungo il canale trasmissivo, e **attesa** e **inoltro**, cioè l'attesa che avviene nei nodi intermedi per smistare il messaggio.
- **prodotto banda-ritardo**: una misura aggiuntiva che prevede il prodotto della larghezza di banda per la latenza.

- **jitter**: è la variabilità del ritardo, in base al contenuto

I dati devono essere convertiti in segnali analogici o digitali per poter essere convertiti. Per questo motivo, interviene il meccanismo di **codifica**. La codifica può essere divisa in due principali categorie: la **codifica lineare** e la **codifica a blocchi.** Noi ci concentreremo principalmente su quella lineare. È il processo di conversione dei dati digitali in segnali digitali. L'unità di base dei dati è il bit, cioè il più piccolo pezzo di informazione che possiamo rappresentare. Possiamo scegliere diversi tipi di codifica lineare, scegliendoli in base allo spettro del segnale, alla sincronizzazione temporale e alla rilevazione degli errori. La codifica lineare può essere **unipolare** o **polare**. In uno schema unipolare, i valori di tutti i livelli del segnale hanno lo stesso segno, cioè o sono positivi o sono negativi. Uno schema unipolare molto utilizzato è l'**NRZ** (*Non-Return-to-Zero*), che rappresenta il valore 1 con un voltaggio positivo e il valore 0 con un voltaggio negativo. In uno schema polare, invece, i valori dei livelli di un segnale possono assumere sia voltaggi positivi che negativi. La codifica NRZ ha anche una versione polare. In questo caso, però, esistono due varianti. L'**NRZ-I** (*Invert*) e l'**NRZ-L** (*Level*). In NRZ-L, un volta positivo rappresenta 0, mentre uno negativo 1. Invece, in NRZ-I, il cambio di livello del vantaggio indica il valore del bit. Se il cambio di livello non c'è stato, allora è 0, mentre se c'è stato, allora è 1. Nella codifica polare possiamo introdurre anche la codifica **RZ** (*Return-to-Zero*). Qui vengono utilizzati tre livelli del segnale. 0 rappresenta un valore negativo, 1 un valore positivo e poi c'è anche un livello nullo. Un'altra tecnica di codifica molto importante è la **codifica AMI** (*Alternate Mark Inversion*), dove esistono sempre tre livelli. Il bit 0 rappresenta un voltaggio nullo, mentre 1 rappresenta, in maniera alternata, sia voltaggio positivo che negativo. Di particolare importanza è anche la **codifica Manchester**. Ogni bit può avere due livelli di segnale. Per il bit 0, il primo livello ha un voltaggio positivo, mentre il secondo negativo. Viceversa per il bit 1, che ha il primo livello negativo e il secondo positivo.
Per ora ci siamo soffermati sulla codifica digitale-digitale. Ma, spesso, potremo trovarci di fronte una codifica del tipo analogico-digitale. La tecnica principale per effettuare questa codifica è attraverso la **modulazione a impulsi codificati** (**PCM**). Essa presenta tre fasi:

- **campionamento**: il segnale viene misurato a intervalli regolari. Questa procedura può essere effettuata in tre maniere diverse. Esiste il campionamento **ideale**, dove gli impulsi del segnale vengono misurati negli istanti di tempo a distanza T, il campionamento **naturale**, dove la misura del segnale avviene in un breve intervallo di tempo. Infine, avremo il campionamento **a gradini**, dove la misura del segnale è

costante. Rifacendosi al **teorema di Nyquist**, possiamo dire che la frequenza di campionamento deve essere almeno il doppio della frequenza del segnale.

- **quantizzazione**: effettuato il campionamento, avremo la misurazione di una serie di impulsi che spaziano dall'ampiezza minima all'ampiezza massima del segnale. Per poter essere codificati, c'è bisogno di approssimarli ad un insieme finito di valori. Questa è la quantizzazione. In parole povere, ampiezza minima e massima vengono messe in rapporto con un valore arbitrario, che determina la precisione dell'approssimazione.
- **codifica**: i valori presi dalla quantizzazione, vengono trasformati in una sequenza di bit. Per farlo si utilizza la rappresentazione binaria di questi ultimi.

La terza codifica che vedremo è quella che trasforma i dati digitali in un segnale analogico. Per farlo, si ricorre alla **modulazione**. Essa può essere effettuata seguendo alcune caratteristiche:

- **modulazione ASK** (*Amplitude Shift Keying*): il segnale viene modulato in ampiezza, cioè si modifica la sua ampiezza, mantenendo costanti frequenza e fase.
- **modulazione FSK** (*Frequency Shift Keying*): viene modificata la frequenza del segnale, mantenendo costanti fase a ampiezza.
- **modulazione PSK** (*Phase Shift Keying*): viene modificata la fase del segnale, mantenendo costanti ampiezza e frequenza.

Esistono, poi, altre tecniche di modulazione, che derivano da queste. Potremo avere la **BPSK** (PSK Bifase), dove sono presenti due tipi di segnali e 1 e 0 vengono fatti corrispondere con 0° e 180° (la fase si misura in gradi). Poi, possiamo avere anche la **QPSK** (*PSK Quadrature*), dove sono previste quattro fasi. Infine, ci teniamo a citare anche la **QAM** (*Quadrature Amplitude Modulation*). Esso combina sia la modulazione in ampiezza che in fase.
Quando la larghezza di banda di un canale trasmissivo è maggiore della larghezza di banda effettivamente necessaria, il canale può essere condiviso da più trasmissioni simultanee. Per questo, è necessario usare la tecnica del **multiplexing**. In questo caso, c'è un insieme di n linee, che condividono la larghezza di banda di un collegamento fisico. Le linee portano i dati da trasmettere per ognuno dei canali logici del **multiplexer**. Egli crea un singolo segnale che verrà spedito sul canale fisico. Una volta arrivato a destinazione, un dispositivo speculare, il **demultiplexer**, ricostruisce i segnali originali. Possiamo distinguere diversi tipi di multiplexing:

- **FDM** (**a divisione di frequenza**): i segnali dei singoli canali logici vengono modulati utilizzando frequenze portanti diverse. Queste ultime vengono scelte in modo che non interferiscano l'un l'altro. In fase di demultiplexing, poi, vengono utilizzati una serie di filtri per scomporre il segnale composto ricevuto.
- **WDM** (**a divisione di lunghezza d'onda**): si tratta di una tecnica particolarmente utilizzata per i cavi in fibra ottica e la modulazione del segnale viene effettuata in base alla sua lunghezza d'onda.
- **TDM** (**a divisione di tempo**): viene progettato per condividere un canale digitale, piuttosto che la larghezza di banda. La disponibilità del canale è divisa in periodi temporali che si dedicano, a turno, a diversi flussi trasmissivi. Ogni intervallo temporale è detto slot, mentre il flusso di dati viene organizzato in frame. A sua volta, il TDM si divide in due metodologie differenti. Il **TDM deterministico**, dove ogni canale è identificato dalla sua posizione in termini di slot, e il **TDM statistico**, dove non esiste correlazione tra il canale e il relativo slot.
- **CDM** (**a codifica**): si moltiplica l'informazione binaria ricevuta per una parola codice, chiamata chip. La sequenza che ne esce fuori sarà poi modulata e trasmessa.

Un altro componente molto importante dello strato fisico, sono i **mezzi trasmissivi**. Possono essere classificati in due macro-categorie, che sono i **mezzi guidati**, elettrici e ottici, e i **mezzi non guidati**, onde radio e raggi X. Ogni mezzo ha alcune caratteristiche comuni, come la larghezza di banda, il delay, il costo e la facilità di installazione e manutenzione. Parliamo, adesso, dei **mezzi trasmissivi elettrici** in maniera più dettagliata, perché si trattano ancora del mezzo più diffuso. Un mezzo trasmissivo elettrico ideale, che trasporta tutta l'energia senza distorsioni o dissipamento, non esiste. Ma, possiamo realizzarne uno ottimale, con bassa resistenza e induttanza. La principale causa di problemi risultano essere i **disturbi elettromagnetici**, che noi indicheremo con **EMI**, che possono essere contrastati con la schermatura, di cui abbiamo due sistemi principali:

- **foglio**: un foglio di alluminio molto sottile che avvolge il cavo appena sotto la guaina di protezione esterna. L'alluminio ha una forte resistenza elettrica e può essere rinforzato anche con del rame.
- **calza**: treccia di fili di rame che avvolgono il cavo in due direzioni opposte. Sono sicuramente più conducibili rispetto all'alluminio, ma coprono di meno.

Di solito si ottiene una schermatura ottimale con una combinazione dei due sistemi. Un esempio lampante è il **doppino**, usato soprattutto nella telefonia, che consiste in due fili di rame ricoperti da una guaina isolante, che passano sotto un processo di ritorsione, detto **binatura**. Questa binatura serve a ridurre gli EMI, dato che, se fossero appaiati, interferirebbero e provocherebbero il fenomeno della **diafonia**, quando i mezzi trasmissivi vicini non vengono isolati correttamente. Esistono varie versioni di doppino:

- **STP** (*Shielded Twisted Pair*): versione con una schermatura per ogni coppia, più una schermatura globale.
- **FTP** (*Foiled Twisted Pair*): esiste un'unica schermatura per tutto il cavo e, normalmente, è un unico foglio di alluminio.
- **UTP** (*Unshielded Twisted Pair*): versione non schermata.

Poi, possiamo dividere i doppini in altre ulteriori categorie:

- **Categoria 1**: cavi adatti unicamente alla telefonia analogica.
- **Categoria 2**: comprende i cavi della telefonia analogica e digitale e trasmissione di dati a bassa velocità.
- **Categoria 3**: adatta per reti locali fino a 10 mbps.
- **Categoria 4**: cavi per la rete LAN fino a 16 mbps.
- **Categoria 5**: cavi per reti fino a 100 mbps.
- **Categoria 6**: cavi per reti fino a 1000 mbps.
- **Categoria 7**: fino a 10 gbps.

Oltre al doppino, ci teniamo a citare anche il **cavo coassiale**, che è stato per lungo diffuso all'interno delle reti, utilizzato sia per ethernet che per i collegamenti terminali. Il cavo coassiale consiste in un filo di rame rigido circondato da una garza metallica che funge da schermatura. Più lungo è il cavo, più ampia è la banda. Un altro tipo di trasmissione elettrica, può essere la **trasmissione power-line**, cioè una tecnologia che utilizza la rete elettrica per raggiugere il proprio scopo. Si realizza sovrapponendo alla normale corrente elettrica un segnale a frequenza più elevata che, grazie al filtraggio, si differenzia.
La **fibra ottica** è, essenzialmente, un cavo composto da un'anima trasparente in silicio, avvolto da un rivestimento di vetro con indice di rifrazione diverso. La parte vitrea è coperta da una guaina di plastica nera. Le fibre sono, si solito, raggruppate attorno ad

un filo metallico che ne facilita la posa. L'energia si propaga nella fibra in un numero discreto di configurazioni, dette **modi**, e ogni modo ha delle proprie caratteristiche di propagazione. Si tratta di un mezzo di un mezzo trasmissivo ormai molto usato e vantaggioso, che presenta come unico problema una doppia conversione, cioè da elettrico a luce e poi viceversa. Di particolare importanza per la fibra ottica è la **legge di Snell**, cioè la descrizione del fenomeno per cui un raggio luminoso, passando da un mezzo trasparente ad un altro, con densità diversa, cambia direzione nel punto in cui attraversa la superficie di separazione dei due mezzi. Nel punto in cui cambia direzione, si formano degli **angoli di incidenza** che, se superiori ad un certo valore, indicano una rifrazione totale. L'**apertura numerica** indica quanta luce possiamo lanciare all'interno della fibra senza che essa venga perduta. Maggiore sarà l'angolo di incidenza, più alta sarà l'apertura numerica. Parlando in maniera più tecnica, potremo dire che una fibra ottica è composta da un **core**, dove la luce viaggia attraverso, un **cladding**, che è un elemento in vetro che fornisce un diverso indice di rifrazione rispetto al core e un **rivestimento primario plastico**, applicato direttamente sulla fibra per proteggerla. All'interno della fibra, la luce può propagarsi in modi diversi e, in base a questo, esistono fibre diverse:

- **fibre multimodali**: propagazione secondo diversi modi e percorsi simultanei.
- **fibre monomodali**: propagazione in un unico modo

Chiameremo **finestra operativa** come il range di lunghezza d'onda in cui la fibra lavora in maniera più produttiva. Parte dell'energia che viaggia attraverso la fibra, viene assorbita dal materiale e, di fatto, si ha un perdita in termini di segnali trasmessi, in un problema chiamato **attenuazione**. Minore è l'attenuazione, maggiore è la distanza utile per la trasmissione. Le fibre possono coprire distanze molto lunghe e, di solito, possono dividersi in:

- 1300 nm (**II finestra**).
- 1550 nm (**III finestra**, attenuazione minima).

Il **WDM**, acronimo di *Wavelenght Division Multiplexing*, consente di veicolare più lunghezze d'onda all'interno dello stesso dispositivo fisico, indipendentemente dal tipo di fibra che andremo ad usare. Il WDM mette a disposizione tanti canali trasparenti, detti ***lightpath***. Di solito, questo tipo di multiplexer viene realizzato con filtri o reticoli. La trasmissione attraverso la fibra ottica può essere effettuata secondo diverse modalità:

- con **LED** (*Light Emission Diode*) sulle fibre multimodali.

- con **Laser** sulle fibre monomodali.
- con **VCSEL** (*Vertical Cavity, Surface Emitting Laser*), che sono laser semiconduttori e richiedono un sistema ottico molto semplice.

Fino ad ora, abbiamo descritto doppini, cavi coassiali e fibre ottiche. Ma, al giorno d'oggi, molti utenti hanno bisogno di essere sempre connessi e, quindi, questi mezzi trasmissivi non hanno alcuna utilità. Per questo, si ricorre ai **mezzi trasmissivi wireless**.
Prima, però, facciamo alcune precisazioni. Quando gli elettroni si spostano, generano **onde elettromagnetiche**, che si propagano attraverso lo spazio. Il numero di oscillazioni eseguite da un'onda è chiamato **frequenza** ed è misurato in ***hertz***. La distanza tra due massimi (o minimi) è detta **lunghezza d'onda** ed è generalmente indicata con la lettera **λ**. Le onde elettromagnetiche viaggiano alla **velocità della luce**. In base ai tipi di onde elettromagnetiche, esse si posizionano all'interno dello **spettro elettromagnetico**. Le onde radio, microonde, infrarossi e luce visibile si possono utilizzare per trasmettere informazioni modulando l'ampiezza e la frequenza. Invece, la luce UV, i raggi X e i raggi gamma funzionano bene, ma hanno una frequenza elevata e sono dannose per gli esseri viventi. In base allo standard **ITU** (*International Communication Union*), è possibile definire le lunghezze d'onda. Di solito sono indicare con **LF**, **MF** e **HF** ed indicano una frequenza bassa, media o alta (*low*, *medium*, *high*). Esistono anche altri tipi, introdotti successivamente, come **VHF**, **UHF**, **SHF**, **EHF** e **THH** (*very*, *ultra*, *super*, *extremely* e *tremendously high*). In base alla banda possiamo definire diverse tecniche di trasmissione:

- **spettro distribuito a frequenza variabile** (**FHSS**): il trasmettitore salta da una frequenza all'altra centinaia di volte al secondo. Viene utilizzata in ambito militare, dato che risulta difficile da individuare e da disturbare.
- **spettro distribuito a frequenza diretta** (**DSSS**): usa una sequenza codificata per distribuire il segnale su una banda di frequenza molto ampia. Il codice che viene assegnato ad ogni segnale è detto **CDMA**. Si tratta di una tecnica molto utilizzata nelle normali trasmissioni di telefonia 3G e anche nel sistema GPS.

Vediamo, adesso, come funzionano le varie parti dello spettro elettromagnetico:

- **trasmissioni radio**: le onde radio sono semplici da generare, possono viaggiare per lunghe distanze e attraversano facilmente gli edifici. Inoltre, sono **omnidirezionali**, cioè trasmettitore e ricevente non devono per forza essere

allineati. Le proprietà delle onde radio dipendono dalla frequenza. Con delle frequenze basse, le onde radio attraversano bene gli ostacoli, ma soffrono del ***path loss***, cioè di un tipo di attenuazione che avviene quando la sorgente di allontana. Con delle frequenze alte, invece, le onde radio rimbalzano contro gli ostacoli e vengono assorbite più velocemente. Indipendentemente dal tipo, le onde radio sono soggette a interferenze da parte dei dispositivi elettrici.

- **trasmissioni a microonde**: sono onde ad alta frequenza. Sono in grado di attraversare gli ostacoli e si propagano in tutte le direzioni, anche se bisogna usare potenze trasmissive relativamente basse. Le microonde viaggiano in linea retta e, se le antenne sono troppo lontane, c'è un'interferenza da parte della curvatura della terra e, quindi, sono necessari ripetitori.
- **trasmissione ad infrarossi**: i raggi infrarossi sono onde di lunghezza millimetrica, direzionabili e che non passano attraverso i solidi. Facciamo una netta distinzione delle tecnologie, perché potremo avere infrarossi **diretti**, trasmettitore e ricevitore sono allineati, e **a diffusione**, dove la trasmissione è di tipo broadcast. Di solito vengono utilizzati per comunicazioni a piccola distanza, ma non sono utilizzabili all'esterno a causa dei disturbi che causerebbero le emissioni solari. Questa tecnologia presenta una variante, detta ***lightwave trasmission***, dove c'è una larghezza di banda maggiore e, quindi, è possibile coprire distanze più ampie, ricordando sempre gli svantaggi degli infrarossi standard.
- **trasmissione satellitare**: usano un **satellite**, che altro non è che un grande ripetitore, che si trova in cielo. Esso contiene diversi **trasponder**, cioè ricetrasmittenti satellitari. Ognuno riceve il segnale in ingresso e lo amplifica in uscita su un'altra frequenza, per evitare interferenze. Possiamo individuare vari tipi di satellite. Abbiamo i **satelliti geostazionari**, che si trovano a 36000 chilometri di quota e appaiono fissi in cielo. Essi sono particolarmente adatti per la trasmissione dei dati a causa del loro puntamento continuo. Poi, potremo avere **satelliti su orbite medie**, che si trovano a 18000 chilometri di quota, con sei ore di periodo nell'orbita. Sono particolarmente adatti per la trasmissione GPS. Infine, abbiamo **satelliti su orbite basse**, situati tra 750 e 1500 chilometri di quota, che sono molto veloci, richiedono poca potenza, ma sono estremamente vicini e vengono usati perlopiù per la telefonia e la navigazione.

DATA LINK LAYER

Il secondo livello del modello ISO/OSI è detto **livello di collegamento dati**. Oltre a soffermarci sulle caratteristiche strutturali di questo livello, faremo anche uno studio abbastanza dettagliato dei protocolli, esaminando la natura degli errori, le loro cause e come possono essere rilevati o corretti. Il DLL fa uso dei servizi messi a disposizione del livello fisico per inviare e ricevere bit lungo il canale di comunicazione. Le sue principali funzioni sono:

- fornire un'interfaccia ben definita per il livello superiore, quello di rete.
- gestire gli errori di trasmissione.
- regolare il flusso dati, in modo che ci sia sincronia tra dispositivi lenti e veloci.

Il DLL prende i pacchetti provenienti dal livello di rete e li incapsula in **frame**, prima di trasmetterli. Ogni frame ha un **intestazione** (*header*), una sezione che contiene il **pacchetto** (*payload field*) e una **sequenza di chiusura** (*trailer*). Vedremo le varie tecniche di framing più nel dettaglio. Prima, però, ci tocca parlare dei servizi che il DLL fornisce al livello di rete:

- **servizio senza conferma e senza connessione**: una macchina sorgente invia frame indipendenti ad una macchina destinazione, senza che avvenga una notifica di avvenuta ricezione. L'uso di questo servizio è appropriato per trasmissioni con una frequenza di errori molto bassa.
- **servizio con conferma e senza connessione**: simile al precedente, solo che è prevista una notifica di avvenuta ricezione. Se il frame non arriva durante un intervallo di tempo definito da un qualche protocollo, allora può essere spedito. L'uso di questo servizio è limitato a canali non affidabili, quindi le reti wireless.
- **servizio con conferma e con connessione**: la macchina sorgente e quella di destinazione stabiliscono una connessione prima di iniziare la comunicazione. Ogni frame viene numero e, quindi, ci assicuriamo che sia consegnato una sola volta e in modo corretto. L'uso di questo servizio è consigliato per collegamenti lunghi, come nelle reti satellitari o telefoniche.

Quando il DLL usa il servizio fornito dal livello fisico, quest'ultimo non gli garantisce che il canale sia privo di errori. Quindi, il DLL avrà il compito di rilevare questi errori e, eventualmente, correggerli. Un primo passo è quello di suddividere i bit in una serie discreta di frame. Esistono quattro tecniche principali di framing:

- **conteggio dei byte**: viene inserito un campo nell'header per specificare il numero di byte del frame. Il DLL legge questo numero, sa quanti byte ci sono e raggiungerà la fine del frame. Il problema di questa tecnica è che il conteggio potrebbe essere facilmente alterato. In caso di errore, il livello fisico aggiungerebbe ridondanza tra i bit e, ovviamente, il numero non sarebbe più veritiero.
- **flag byte con byte stuffing**: vengono inseriti, appunto, dei flag byte nell'header e nel trailer di ogni frame. Due flag byte consecutivi indicano che il frame è finito e ne è iniziato un altro. Però, specialmente nei dati binari, è possibile che questo flag byte appaia anche all'interno della sezione centrale del frame, quindi all'interno dei dati. Per questo motivo, utilizziamo la tecnica del byte stuffing, cioè l'aggiunta di un **byte di escape** subito prima di ogni flag byte all'interno del pacchetto. Così facendo, avremo disambiguazione e non ci saranno errori durante il framing.
- **flag bit con bit stuffing**: si lavora a livello di gruppi più omogenei di bit, non più in byte (8 bit). Viene seguito un protocollo ben specifico, dove ogni frame comincia a finisce con una speciale sequenza di bit, `01111110`, equivalente di un flag byte. Ogni volta che si incontrano cinque bit a `1` consecutivi, automaticamente si inserisce uno `0`. Così, quando il destinatario riceve la trasmissione e incontra cinque `1` consecutivi, seguiti da uno `0`, automaticamente lo elimina.
- **violazione della codifica**: si tratta di una scorciatoia del livello fisico. Abbiamo visto che, per codificare i bit in segnali, spesso viene inserita una ridondanza nei dati. Così, alcuni segnali, non risultano trasparenti al destinatario. Questi segnali verranno poi usati per delimitare il frame, dato che non hanno un'utilità particolare.

Spesso, il DLL utilizza una combinazione di questi metodi.
Come abbiamo già detto, il DLL ha il compito di gestire gli errori di trasmissione. Esistono due principali strategie. La prima usa **codici a correzione di errori**, mentre la seconda **codici a rilevazione di errori**. A seconda dei dispositivi di trasmissione, usiamo una o l'altra tecnica. Vediamo prima i meccanismi di correzione degli errori. Prima di continuare diremo che un frame è composto da m bit di dati (messaggio) ed r bit di ridondanza (controllo). Quindi, la lunghezza totale di un blocco n, sarà uguale ad $m + r$. Questo tipo di blocco è detto **parola del codice** (*codeword*). In due sequenze

di bit, possiamo calcolare un numero che indica quanti bit differiscono. Questo numero è detto **distanza di Hamming** ed è di particolare importanza, perché se due codeword sono a distanza di Hamming d una dall'altra, saranno necessari d errori per convertire una sequenza nell'altra.
Per progettare un buon codice a correzione di errori, ogni codeword valida di $m + r$ bit ha bisogno di n combinazioni errate dedicate, in modo che, durante l'intercetto, esse siano riconducibili alla codeword valida. Un approccio che utilizza questo meccanismo è la **codifica di Hamming**, che è anche il più utilizzato, sia per **errori singoli**, che per **errori burst** (errori su gruppi di bit).
La correzione degli errori risulta poco ottimale per trasmissioni in fibra ottica o doppino, dato che c'è un tasso basso di errori e, correggerli, richiederebbe troppo tempo e troppe risorse che non sono assolutamente necessari. La tecnica più usata è sicuramente la **codifica polinomiale**, detta anche **controllo ciclico di ridondanza** (**CRC**), in cui ogni sequenza di bit viene vista come un polinomio. Ad ogni frame viene aggiunta una nuova sequenza di bit, detta **checksum**, in modo che il polinomio $frame + checksum$ risulti divisibile per un polinomio $G(x)$, chiamato **polinomio generatore**, concordato tra sorgente e destinazione. Il destinatario divide il polinomio ricevuto per $G(x)$ e, se ottiene reste non nullo, sa che tale polinomio ha subito un errore. Questa tecnica polinomiale è molto utilizzata soprattutto nelle connessioni LAN, ma anche punto-a-punto.
Un'altra problematica da affrontare, quando si parla di DLL, è il controllo del flusso, cioè quando una sorgente vuole trasmettere i frame molto più velocemente di quanto la destinazione sia in grado di accettare. Esistono due principali approcci:

- **controllo di flusso tramite feedback**: la destinazione manda alla sorgente delle informazioni per informarla del suo stato e permettere lo scambio di dati.
- **controllo di flusso tramite limitazione del tasso di invio**: c'è un meccanismo che limita il tasso al quale la sorgente può trasmettere i dati.

Un altro meccanismo potrebbe essere quello di valutare i tempi di risposta del destinatario e inserire dei ritardi per farlo adattare alla capacità di trasmissione. Il tempo di processamento, però, non è una costante e non può in alcun modo rappresentare una soluzione. La presenza di un controllo dipende soprattutto dal **protocollo di comunicazione** che viene usato nel DLL e, ora, ne vedremo alcuni. Indicheremo con **ACK** una notifica positiva, cioè che il pacchetto è arrivato correttamente senza errori, e con **NAK** una notifica negativa, che ci sono stati problemi di trasmissione. Nel caso sia

stato ricevuto un NAK, allora il mittente ritrasmette il pacchetto. Vediamo ora alcuni protocolli:

- **protocollo stop-and-wait**: prevede che A, dopo aver inviato il frame, si fermi per attendere un riscontro. B, una volta ricevuto il frame, invierà ad A un frame di controllo, privo di dati, per dire ad A che può ricominciare a trasmettere. Il traffico di questo esempio è **simplex**, cioè dalla sorgente alla destinazione. Per ora abbiamo previsto una comunicazione senza errori, ma che succede se i pacchetti vengono danneggiati? Mittente e destinatario non sono sincronizzati. Per questo, è necessario aggiungere la checksum alla notifica, che sia ACK o NAK. Così, il mittente rinvia i pacchetti quando riceve una notifica alterata. Questo, però, provoca la duplicazione e il destinatario non saprà se il pacchetto è stato duplicato. La soluzione sta nell'aggiungere un **numero di frequenza**. Il destinatario lo controllerà e saprà se è nuovo o duplicato. Protocolli di questo tipo vengono detti **PAR** (*Positive Acknowledgement with Retransmission*) oppure **ARQ** (*Automatic Repeat reQuest*).
- **piggybacking**: fino ad ora abbiamo supposto un protocollo che permettesse la trasmissione solamente in una direzione, ma nelle situazioni reali, quasi mai è così. Il piggybacking è una tecnica molto utilizzata. Di solito, un protocollo richiede sempre un messaggio di conferma tra mittente e destinatario attraverso l'invio di un ACK. Esso può essere inviato immediatamente all'avvenuta ricezione ma, tuttavia, se il protocollo prevede altri scambi successivi, il destinatario può rimandare nuovamente l'ACK e includerlo nel messaggio successivo. Quindi, il frame inviato ha **in groppa** l'ACK relativo all'ultima ricezione. Questa tecnica è molto adatta per le reti LAN, ma poco efficiente, invece, per quelle molto grandi.
- **protocolli a finestra scorrevole**: sono caratterizzati dalla dimensione di una finestra, che stabilisce il numero di frame che possono essere inviati prima di fermarsi e attendere l'ACK. Poiché è possibile inviare più frame consecutivi, è necessario che essi vengano numerati. La numerazione è in modulo 2^n. La dimensione della finestra si può decidere a priori o può cambiare dinamicamente in base alla rete o in base ai trasmittenti. Di solito, avremo comunque una **finestra di trasmissione** e una **finestra di ricezione**. La prima deve tener conto dei frame di cui ancora non c'è riscontro, attraverso un buffer, mentre la seconda dei frame ricevuti, che devono essere conservati dato che ancora non è stato inviato nessun riscontro. Questo protocollo permette di ottimizzare al meglio la trasmissione, ma

risulta poco efficiente quando si tratta della gestione degli errori. Il mittente, prima di accorgersi dell'errore, avrà già inviato molti frame. Per questo, usiamo due varianti di questo protocollo. Il primo è detto **go-back-n**. Nel momento in cui troviamo un errore, tutti i frame successivi vengono scartati, senza inviare nessun ACK. Se la frequenza di errori è alta, allora perderemo molta banda. L'altra variante è detta **selective-repeat**. Qui, un frame errato viene scartato, mentre i successivi corretti vengono inseriti all'interno di un buffer. Quando la trasmissione va in timeout, il frame corretto più vecchio viene trasmesso. Se tutto va a buon fine, allora vengono trasmetti tutti gli altri frame del buffer. Questo approccio richiede molta memoria se la finestra è molto grande.

Adesso, tocca concentrarci sui protocolli DLL per le **reti LAN**, che saranno oggetto di esame. All'interno di esse, esistono due tipi di collegamento. Il primo è quello **punto-a-punto**, impiegato per le connessioni a lunga distanza, mentre il secondo è **broadcast**, basato sulla presenza di un canale condiviso. Quest'ultimo è quello più utilizzato nelle reti LAN ma, se fatto in modo improprio, può causare **collisioni**, cioè quando due nodi ricevono due o più frame contemporaneamente. Per questo, avremo bisogno di protocolli ad accesso multiplo, che fissano le modalità con cui i nodi regolano le trasmissioni sul canale condiviso. Essi si possono classificare in tre macro-categorie:

- **protocolli a suddivisione del canale**: il canale viene partizionato.
- **protocolli ad accesso casuale**: non c'è nessun partizionamento e i nodi coinvolti trasmettono ripetutamente i pacchetti.
- **protocolli a rotazione**: ciascun nodo ha il suo turno di trasmissione, con quelli più coinvolti che hanno turni più lunghi.

Per quanto riguarda la Categoria 1, essa si divide a sua volta in diversi tipi di protocolli:

- **TDMA**: acronimo tradotto di accesso multiplo a divisione di tempo. Il canale è diviso per intervalli di tempo.
- **FDMA**: acronimo tradotto di accesso multiplo a divisione di frequenza. Il canale è suddiviso in bande di frequenza. Ogni nodo ha assegnata una banda di frequenza prefissata.

Invece, per ciò che concerne la Categoria 2, essa descrive come bisogna comportarsi per evitare le collisioni, adottando protocolli ad accesso casuale:

- **Slotted ALOHA**: assumendo il tempo diviso in slot, i pacchetti con la stessa dimensione e i nodi tutti sincronizzati, quando ad un nodo arriva un nuovo pacchetto, il nodo attende fino all'inizio dello slot successivo. Nel caso non si verifichi una collisione, il nodo trasmette senza problemi. In caso contrario, il nodo la rivela prima della fine dello slot e ritrasmette con probabilità p durante gli slot successivi. Si tratta di un protocollo semplice, decentralizzato e veloce, ma non garantisce la completa immunità dalle collisioni, quindi ci sarà spreco di tempo.
- **ALOHA puro**: i nodi non sono sincronizzati. Quando arriva un pacchetto, viene direttamente ritrasmesso nel canale, quindi non si adotta alcuna contromisura per le collisioni. Ad aumentare il traffico, ovviamente, aumentano le collisioni e, quindi, il protocollo risulta poco efficiente.
- **CSMA**: acronimo tradotto di accesso multiplo a rilevazione della portante. Esistono due tipi di implementazione. Quella **persistente**, quando il nodo si mette in ascolto all'interno del canale, se è occupato, attende, altrimenti inizia la trasmissione, oppure quella **non persistente**, che è più o meno la stessa cosa, solo che il nodo attende un tempo fissato prima di rimettersi in ascolto. Questo protocollo, però, non risolve le collisioni, perché a causa del **ritardo di propagazione**, non è previsto che due nodi rilevino la reciproca trasmissione. Per questo motivo, andiamo ad introdurre due varianti di questo protocollo. Il primo è il **CSMA/CD**, acronimo di CSMA Collison Detection. Esso è provvisto di tre fasi. La prima è la rilevazione della trasmissione (***carrier sense***), quando un nodo si mette in ascolto e decide di trasmettere solo se questo è libero. La seconda è l'accesso multiplo (***multiple access***) dove, vista la presenza del ritardo di propagazione, il nodo può credere che il canale sia libero anche se un nodo sta ancora trasmettendo. La terza è la rilevazione della collisione (***collision detection***) quando, per rilevare una collisione, un nodo, mentre sta trasmettendo, si mette in ascolto per rilevare segnali sul mezzo trasmissivo, confrontandoli con quelli generati da lui stesso. Alla fine di una trasmissione, potremo avere il periodo di contesa, cioè se due o più nodi cominciano a trasmettere. Questo periodo è detto **slot di contesa**. A seguito di un'avvenuta collisione, si intraprendono diverse azioni. Il nodo trasmittente interrompe la comunicazione e trasmette una sequenza di **jamming** (interferenza trasmissiva), per comunicare a tutti l'avvenuta collisione. I nodi in ascolto, quindi, scartano i bit ricevuti. Il nodo trasmittente ripete la trasmissione dopo un lasso di tempo casuale ma, raggiunta la **sedicesima** volta, non lo farà più.

Per la Categoria 3, quindi per i protocolli a rotazione o a prenotazione:

- **mappa di bit elementare**: sulla rete avremo N nodi, numerati da 0 a $N-1$. Alla fine della trasmissione di un frame, inizia uno slot di contesa, in cui ogni nodo trasmette un bit `1`, se ha bisogno di comunicare, o `0` altrimenti. Al termine della contesa, tutti i nodi sanno chi deve trasmettere e chi no e, così, procedono un frame per volta. Se N è molto grande, questo protocollo risulta poco efficiente.
- **token ring**: non viene utilizzata una trasmissione broadcast, ma un insieme di collegamenti punto-a-punto, associati in successione, per creare un anello. Sull'anello circola un piccolo frame, detto token, che i nodi ricevono da una parte e ritrasmettono dall'altra. Un nodo è autorizzato a trasmettere solo quando è in possesso del token. Andando nello specifico, il token è una sequenza di bit, che circola sull'anello quando tutti i nodi sono inattivi. L'anello deve avere un ritardo sufficiente per contenere il token circolante. In condizioni di basso carico, però, questo protocollo è poco efficiente.

Lo **standard IEEE 802** definisce un insieme di standard per le reti LAN e MAN, relativamente al DLL e al livello fisico. L'idea alla base è che le reti LAN e MAN forniscano un'interfaccia unificata verso il livello di rete. Per questo motivo, il DLL venne diviso in due sottolivelli: **LLC** (*Logical Link Control*) e **MAC** (*Media Access Control*). LLC è un'interfaccia comune a tutte le LAN, mentre il MAC è peculiare di ciascuna LAN. Il MAC è sicuramente indispensabile, in quando le LAN implementano sempre una rete di tipo broadcast (utilizzo solo di un canale trasmissivo). Per favorire questo tipo di comunicazione, vengono introdotti gli **indirizzi MAC**, formati da 48 bit, che servono a identificare il destinatario. Per determinare l'indirizzo MAC di una determinata macchina, partendo solo dal suo indirizzo IP, si utilizzano le cosiddette **tabelle ARP**, che contiene le corrispondenze e il tempo di vita di ciascuna voce. A detenere una posizione di rilievo nel mercato delle LAN cablate è sicuramente **Ethernet**, una LAN ad alta velocità, più semplice e meno costosa di un token ring. Per Ethernet, la topologia più usata, è quella a stella. Ethernet è composto dai seguenti elementi:

- **preambolo**: composto da otto byte, serve ad attivare gli adattatori del ricevente e a sincronizzare l'orologio con quello del trasmittente.
- **indirizzo di destinazione**: composto da sei byte, serve ad attivare la trasmissione, dato che, quando un adattatore riceve un pacchetto, legge l'indirizzo di destinazione e lo trasferisce al livello di rete.

- **campo tipo**: consente ad Ethernet di supportare i vari protocolli di rete.
- **controllo CRC**: consente all'adattatore del destinatario di rilevare la presenza di un errore.

Essendo Ethernet un CSMA/CD persistente, quando il carico è basso, la rete si comporta in maniera ottimale, riducendo drasticamente i ritardi. La rete, però, può adeguarsi anche ad un carico più alto, utilizzando l'**algoritmo del back-off**, che riduce i nodi della rete, peggiorando le prestazioni, ma continuando a funzionare correttamente. Ethernet domina, ormai, il mercato delle LAN. All'inizio, viaggiava sul cavo coassiale a circa 10 mbit/s. Ora, conservando le stesse caratteristiche in termini protocollari, viaggia anche attraverso fibra e doppino e a oltre 100 gbit/s. Vediamo, nel dettaglio, alcune tecnologie di Ethernet note:

- **10Base5**: si tratta della prima versione di Ethernet, che usava un cavo giallo di grandi dimensioni, chiamato **thick Ethernet**. Questo cavo aveva tacche specifiche ogni due metri e mezzo, che serviva al dispositivo **transceiver** di rilevare le collisioni.
- **10Base2**: l'uso del transceiver rischiava, però, di rompere il cavo, quindi venne introdotto un cavo più flessibile, con un adattatore che lo interrompeva e si inseriva. Il problema era che, nel caso avessimo voluto inserire un nuovo nodo, avremo dovuto aprire il cavo completamente, inserire l'adattatore e chiuderlo di nuovo.
- **10BaseT** e **100BaseT**: la lettera T equivale a **twisted pair**, doppino intrecciato. Ogni nodo ha una diretta connessione con quello centrale e la massima distanza tra i due e, al massimo, di cento metri. Abbiamo anche due varianti di queste versioni, la prima è **100Base-TX**, che utilizza due doppini, mentre **100Base-FX** utilizza due cavi in fibra multimodale.
- **Gigabit Ethernet**: utilizza sempre il protocollo CSMA/CD, solo che il rilevamento delle collisioni è esteso soltanto a due nodi, trasformando il tutto in una connessione punto-a-punto. Funziona sia in half-duplex che in full-duplex e ha una compatibilità a ritroso, cioè può essere efficiente anche a 100 e 10 mbps.

Per costruire reti più grandi, il livello fisico ci mette a disposizione i **ripetitori**, che amplificano il segnale e lo ritrasmettono alla stessa frequenza, in modo bidirezionale. Possiamo anche usare più di un ripetitore, creando i cosiddetti **hub**. A livello fisico, tutti i dispositivi sono nello stesso dominio di collisione, di broadcast e condividono la banda. Aumentando la dimensione della rete, andremo incontro alle collisioni, quindi ci

toccherà adottare il protocollo CSMA/CD per rilevarle. Ma, sappiamo anche questo protocollo non è efficiente per reti molto grandi. Introduciamo il **bridge**. Il bridge è un dispositivo multiporta che lavora al livello di DLL e permette di connettere reti LAN che usano diversi DLL, separando i domini di collisione. La funzione principale del bridge, è quella di partizionare la rete e ridurre il carico, ma può anche assicurarsi che tutti i frame siano stati inviati in maniera corretta, individuando eventuali errori. Il bridge, per adempiere alle sue funzioni, ha bisogno di conoscere la topologia della rete per poter raggiungere i suoi obiettivi. Per questo motivo, utilizzeremo il cosiddetto ***transparent bridging***. Funziona con una logica *plug-and-play*, che vuol dire assenza di configurazione, ed impara da solo la topologia di rete, osservando il traffico che la attraversa. Così facendo, costruisce una tabella hash interna (***MAC Address Table***), che associa gli indirizzi sorgente all'interfaccia del bridge. Quest'operazione è detta ***backward learning***. Usando la tabella, il bridge decide se scartare il frame o ritrasmetterlo.
Dal punto di vista funzionale, uno **switch** è un bridge multiporta, anche se molto cambia a livello di prestazioni. Questo perché un bridge è fondamentalmente software-based, mentre lo switch è hardware-based. Nello switch, il dominio di collisione è confinato alla singola porta. In caso di trasmissione full-duplex, non avremo collisioni perché switch e dispositivo possono inviare e ricevere allo stesso tempo. In realtà, è anche possibile collegare bridge e switch per interconnettere più reti LAN, creando topologie più complesse. Per sapere in quale porta debba essere trasmesso il frame, lo switch deve creare e mantenere aggiornata una tabella relativa all'associazione tra indirizzo sorgente e porta di destinazione. Anche quest'operazione viene effettuata tramite plug-and-play. In base alle loro caratteristiche, i dispositivi di switch possono essere divisi in diverse categorie:

- ***cut-through***: ogni volta che viene letto l'header del frame, viene immediatamente stabilita una cross-connection, che è una connessione ad alta velocità, tra mittente e destinatario. I bit cominciano ad essere trasmessi prima della fine del frame. Inoltre, anche i frame in cui sono stati rilevati errori, vengono trasmessi.
- ***store-and-forward***: il frame, prima di essere trasmesso, è letto e bufferizzato dallo switch. Tramite CRC si controllano gli errori e, nel caso vengano rilevati, il frame viene scartato. Questa tecnica permette di filtrare il frame, ma richiede una grande capacità di memorizzazione.
- ***port-based***: ad ogni porta corrisponde un indirizzo MAC.

- ***segment-based***: ad ogni porta corrispondono più indirizzi MAC.

Gli switch possono anche essere divisi in base alla tecnologie che utilizzano:

- ***shared-memory***: memorizza i pacchetti in una memoria comune a tutte le porte. Sarà poi la memoria ad occuparsi di incanalare i pacchetti alla destinazione corretta. Ha n input ed m output e tutte le porte di ingresso sono anche di uscita. L'obiettivo è quello di massimizzare la produttività. Questa tecnologia ha limitata scalabilità, ma è poco costosa.
- ***switching-matrix*** o ***fabric***: utilizza una matrice di commutazione. In base all'indirizzo e al contenuto della matrice, viene attivata la connessione necessaria.
- ***bus-architecture***: ha un bus interno condiviso a media velocità. La comunicazione interna usa un TDMA.
- ***crossbar switch***: ogni input è connesso ad ogni possibile output. Qui si possono presentare problemi di contesa e, aumentando il numero delle porte, aumenta anche la complessità strutturale. Un crossbar switch può essere **perfetto** se la sua logica è centralizzata, cioè ha tutte le linee connesse e gode della proprietà di *non-blocking*, cioè è sempre disponibile. La complessità, ovviamente, cresce in maniera esponenziale. È possibile ridurla, utilizzando la **MIN** (*Multistage Interconnection Network*). Lo switch viene partizionato in diversi stadi, con capacità limitata, ma interconnessi tra di loro. L'unico difetto è che, in presenza di una richiesta simultanea, avremo un blocco.

A proposito di blocco, se c'è una richiesta simultanea, è necessario che una di loro si blocchi e attenda il passaggio dell'altra. La **contesa**, però, può ridurre la produttività. Per questo motivo, è indispensabile l'uso della **bufferizzazione**. Ogni porta di output ha un buffer dedicato. Essi vengono riempiti da un **shifter**, utilizzando un algoritmo di tipo **RR** (Round-Robin), anche se l'ordine di arrivo è comunque preservato. Ci sono, ovviamente, alcuni sistemi che non tollerano assolutamente situazioni di blocco o contesa. Per questo motivo, è necessario introdurre la **ridondanza**, eliminando i cosiddetti *single points of failure*. In compenso, però, potremo avere la cosiddetta **tempesta di broadcast** (*broadcast storm*). Mettiamo caso che un host X invii un frame. Lo switch A replica il frame su tutte le sue porte, quindi anche la B, mentre lo switch B replica nuovamente il frame in tutte le sue porte, e quindi anche in A. Il frame ritorna ad A all'infinito, creando quindi la tempesta. Un altro problema importante è la **replicazione di frame**. Mettiamo caso che un host X invii un frame unicast (destinato

ad una sola macchina) ad un router Y. L'indirizzo del router non è conosciuto né dallo switch A e né dallo switch B. Entrambi gli switch, quindi, inviano il messaggio su tutte le porte. Y, quindi, riceve lo stesso frame due volte. Il terzo problema che andremo ad affrontare, sarà l'**instabilità del MAC Address Database**. Mettiamo caso che l'host X invii un frame unicast al router Y. Gli switch A e B sanno che l'indirizzo MAC di X sta sulla porta 0. Non conosce Y, gli switch fanno un inoltro su tutte le porte, quindi l'indirizzo MAC vede che la trama una volta arriva alla porta 0 e una volta alla porta 1, creando instabilità. Un altro problema comune è la presenza di **loop multipli**, molto comuni all'interno di topologie complesse. La soluzione a questo problema è l'utilizzo del protocollo ***Spanning Tree***. In parole povere, rende una topologia ridondante *loop-free*, mettendo in stato di blocco alcune porte. Questo protocollo consiste, essenzialmente, di tre elementi. Abbiamo un ***root bridge***, che è il punto da dove poi si dirama lo ST. Poi, avremo una ***root port***, per ogni switch che non è un root bridge, che serve a indirizzare il root bridge. E, infine, avremo una ***designated port***, che serve per raggiungere un collegamento che non posso spegnere. Di particolare importanza è il **BPDU**, acronimo di *Bridge Protocol Data Unit*, che viene inviata tra gli switch ogni due secondi e normalmente trasporta le informazioni necessarie per eleggere il root bridge, localizzare i cicli e monitorare lo stato dello ST. Per eleggere il root bridge, in partenza, tutti gli switch mettono il proprio bridge ID come Root ID e poi, man mano che si scambiano le informazioni con la BPDU, si individua sia il root bridge, che il cammino ottimale per lo ST, vedendo quello con i cammini più corti e che ha un costo minore. Il **costo path**, così come viene chiamato, dipende dal percorso che ogni collegamento fa fino a raggiungere la propria destinazione. Quando si usa uno ST, le porte passano sotto quattro differenti stati:

- ***blocking***: non passano frame, ma solo la BPDU.
- ***listening***: la porta si mette in ascolto e i frame passano attraverso di essa.
- ***learning***: avvia il backward learning, cioè associando gli indirizzi all'interno della MAC Address Table.
- ***forwarding***: aggiorna il MAC database e i frame possono passare.

Introduciamo, adesso, un altro concetto molto importante per il DLL, cioè le reti **VLAN**, acronimo di Virtual LAN. In una rete flat, ogni dispositivo collegato in rete vede ogni pacchetto trasmesso. Invece, in una rete VLAN, c'è una serie di switch che sono confinati all'interno dello stesso dominio di broadcast, in modo tale che se un pacchetto viene trasmesso ad un gruppo di switch, soltanto quelle macchine riceveranno il

pacchetto desiderato. Ogni VLAN è logicamente equivalente ad un bridge fisico. Se abbiamo uno switch con 12 porte e abbiamo 3 VLAN, lo switch sarà diviso in tre, con quattro porte ciascuno. Le reti VLAN non si vedono all'interno del DLL, ma solo nel NL. Per garantire che una VLAN passi correttamente da uno switch all'altro, si dedica una porte speciale, detta **trunk**. Si può identificare a quale VLAN appartenga un frame in arrivo:

- **in base alla linea di arrivo** (**associazione statica**): ogni porta fisica l'associo a determinate VLAN.
- **in base al MAC Address di provenienza** (**associazione dinamica**): uno specifico MAC Address deve andare su una VLAN.

Questa modalità di associazione influisce sulle prestazioni e, per questo, si utilizza quella statica, che è più efficiente. Quando un frame deve uscire da uno switch e passare in un altro, bisogna aggiungere un campo, detto **tag**, che specifica la VLAN di appartenenza. Come abbiamo già specificato, le informazioni sui tag viaggiano attraverso porte speciali, le trunks. Per individuare una porta trunk utilizziamo il protocollo di negoziazione, detto **DTP**, acronimo di *Dynamic Trunk Protocol*. Quando uno switch riconosce un altro switch, prova a negoziare una trunk. Quando due switch si riconoscono, non si occupano soltanto di configurare le trunk, ma possono anche diffondere informazioni sulla configurazione delle VLAN. Lo fanno attraverso il protocollo **VTP** (*VLAN Trunking Protocol*), che garantisce la consistenza delle VLAN all'interno di un singolo dominio amministrativo. Il VTP ha tre modalità operative:

- **Server Mode**: per creare e rimuovere reti VLAN.
- **Client Mode**: non può modificare la configurazione delle VLAN.
- **Transparent**: per creare e rimuovere LAN localmente e lanciare annunci. Gli annunci sono inviati dal server come dei frame multicast. Possono essere inviati periodicamente o su richiesta.

Abbiamo parlato prima di problemi di scalabilità. Quando ci riferiamo allo switching tradizionale, non avremo problemi del genere. Se, invece, allarghiamo il concetto alle VLAN, avremo problemi di scalabilità. Il **trunking** permette di risolvere parzialmente questo inconveniente. Piuttosto che avere un'interfaccia associata ad ogni VLAN, avremo una sola interfaccia nello switch, che trasporta informazioni lungo tutte le VLAN aprendo una trunk. Saranno interfacce virtuali sotto quella reale. Per migliorare questo sistema, utilizziamo la tecnica del **routing on a stick**. Permette di avere infinite

interfacce virtuali per quella reale. Per migliorare ulteriormente questa tecnica, si è passato ad un aiuto hardware, detto **Layer 3 Switching**. Si collega direttamente la parte interna del router alla fabric, così le VLAN comunicherebbero ad altissime prestazioni. Questo meccanismo si chiama così perché integra un **router**, quindi un dispositivo del Livello 3, quello di rete. Il router permette di gestire la qualità del servizio e integra meccanismi di sicurezza. Un ulteriore migliore è stata fatta aggiungendo funzionalità a livello di trasporto, dette **Layer 4 Switching**.
Un altro sistema molto usato è il modello gerarchico a tre livelli. I livelli sono questi:

- livello di accesso: eroga la connettività in rete agli utenti finali.
- livello di distribuzione: separa accesso e l'ultimo livello e garantisce funzioni di raccolta e controllo all'ultimo livello.
- il livello core commuta i frame a grande velocità, senza però effettuare nessuna elaborazione.

Un altro meccanismo è detto switch block ed è composto solo da un livello di accesso, che connette gli utenti finali alla rete, garantendo una banda dedicata per ogni porta; e da un livello di distribuzione, che eroga servizi di broadcast, sicurezza e connessione esterna.
Un protocollo DLL molto importante per le reti ad accesso WAN è l'**ADSL**, acronimo di *Asymmetric Digital Subscriber Line*, che è lo standard per fornire un accesso digitale a banda più elevata di quanto sia disponibile con il modem. La linea telefonica è composta da un doppino con un filtro a 4 kHz, attraverso cui viene trasmessa la voce. Il doppino, però, ha capacità tale da raggiungere il mHz. Per questo, si divide lo spettro disponibile in **256 canali** da 4 kHz. Il canale 0 viene riservato alla telefonia, mentre i successivi quattro non vengono utilizzati. I restanti, invece, vengono impiegati per la trasmissione dati. Non ci sarà interferenza proprio grazie ai quattro canali non utilizzati. Il modem ADSL riceve i dati e li divide in **flussi paralleli**, generando un segnale analogico in base al flusso. I servizi di connettività dati e telefonia vengono erogati da un **central office**, da cui partono i collegamenti in rame di *ultimo miglio* verso gli utenti finali. Uno **splitter** separa i dati dalla voce, mentre un **DSLAM** (*DLS Access Multiplexer*) ha il compito di multiplare le migliaia di accessi ADSL in un'unica interfaccia ad alta densità. Quello che abbiamo definito ultimo miglio, se in rame, limita le prestazioni di ADSL. Quindi, di recenti, si sta sostituendo il rame con le fibre ottiche. In generale si usa la dicitura ***Fiber to the***, per indicare il luogo, che può essere la casa, l'ufficio o altre abitazioni. Un'architettura molto comune per le fibre è detta **PON**, acronimo di *Passive*

Optical Network. Spesso, le fibre si uniscono a formare un'unica grande fibra che può collegare anche diverse centinaia di case. Usando l'architettura PON e, uno splitter, è possibile portare il segnale su diramazioni diverse.
Le reti WLAN appartengono alla categoria delle reti LAN, ma sono **wireless**, quindi utilizzate quando è difficoltoso adoperare i cavi. Le reti wireless hanno molti vantaggi, tra cui i costi ridotti, meno problemi legati alle distanze e la mobilità delle prestazioni.
L'IEEE ha definito diversi standard per le trasmissioni wireless, tra cui il più comune è certamente l'**802.11**, i cui dispositivi appartenenti a questo standard si identificano sotto il nome di **Wi-Fi**. Come detto, le WLAN utilizzano una tecnica a radio frequenza per la trasmissione e la ricezione dei dati, minimizzando il cablaggio e favorendo la mobilità.
All'interno di 802.11 abbiamo due tipi di tecniche trasmissive:

- **Frequency Hopping Spread Spectrum** (**FHSS**): divisione di spettro ad alta frequenza. Qui, la trasmissione salta ad intervalli di tempo definiti, di solito minori di 400ms, da una frequenza ad un'altra, secondo uno schema pseudocasuale noto a tutti. Grazie a questo sistema, viene garantita sicuramente sicurezza e solidità contro il ***multipath fading***, quando il segnala arriva distorto sotto forma di un certo numero di repliche.
- **Direct Sequence Spread Spectrum** (**DSSS**): per far fronte al rumore, utilizziamo la tecnica del ***chipping***, cioè che ogni bit è convertito in una serie di bit ridondanti. Si tratta di un metodo ottimale per la trasmissione e la ricezione di segnali deboli. Viene utilizzato un sistema a dispersione di banda, dove il dato viene modulato prima di essere trasmesso e ogni bit viene disperso su una sequenza di 11 bit (**codifica di Barker**).

Il DSSS è sicuramente più immune ai rumori, ma spreca molta più banda. L'FHSS è più sicuro, ma ha una banda limitata. Le reti WLAN tendono spesso ad avere diverse velocità di trasmissione, quindi tramite un protocollo di ***Dynamic Rate Shifting***, la trasmissione si adatta automaticamente alla natura del canale.
Vediamo, adesso, due dei problemi più comuni all'interno delle reti WLAN.
Consideriamo tre stazioni, A, B e C. B è a portata di A e di C, ma A e C non possono rilevare le rispettive trasmissioni. Se C sta trasmettendo dati a B, A non potrà rilevare l'occupazione del canale, in quanto fuori portata. Per questo, A comincerà a trasmettere, interferendo con la comunicazione tra B e C. Questo problema è chiamato **stazione nascosta**. Supponiamo nuovamente che A stia trasmettendo verso un'altra comunicazione e B e C desiderano comunicare. B, però, trova il canale occupato,

dato che A è alla sua portata, ma potrebbe comunque trasmettere a C senza creare interferenze. Qui, invece, ci troviamo di fronte al problema della **stazione esposta**. Per risolvere questi problemi, usiamo il protocollo **MACA**, acronimo di *Multiple Access with Collision Avoidance*. A invia un piccolo frame, detto **RTS** (*Request to Send*), a B. B, a sua volta, trasmette un altro piccolo frame, detto **CTS** (*Clear to Send*), che autorizza A a trasmettere. Tutte le altre stazioni, sapranno che B ha inviato un CTS ad A e che quindi dovranno aspettare prima di trasmettere. Le interferenze, però, potrebbero comunque verificarsi, nell'ipotesi che vengano inviati due CTS contemporaneamente. Per ovviare anche a questo problema, utilizziamo l'**algoritmo di backoff esponenziale binario**. Ogni stazione riceve un numero casuale n, compreso tra 0 ed m. In caso di interferenza, la stazione attenderà $n * slotditempo$ prima di riprovare. Ad ogni interferenza, m aumenta in maniera esponenziale. Invece, la lunghezza dello slot di tempo deve essere definita in maniera tale che ogni stazione debba possa determinare se un'altra ha avuto accesso al canale nello slot precedente. Il protocollo MACA ha anche una variante, detta **MACAW** (MACA per Wireless), che introduce migliorie specifiche proprio per le reti WLAN. Viene introdotto un frame di ACK con meccanismo *stop-and-wait*, quindi una stazione si ferma e aspetta finché non riesce a trasmettere. L'algoritmo di backoff viene modificato, in modo da essere trasmesso separatamente ai vari canali trasmissivi.
Lo standard 802.11 prevede due modalità operative per la trasmissione dei dati:

- **DCF** (***Distributed Coordination Function***): conosciuta anche come rete ad hoc, prevede la comunicazione tra stazioni senza un arbitraggio centralizzato, ma meccanismi di contesa e di gestione delle collisioni. Le collisioni vengono gestite da un protocollo **CSMA/CA** (*Carriere Sense Multiple Access Collision Avoidance*). Il protocollo controlla il canale, se è libero trasmette, se è occupato aspetta. In caso di collisione, utilizza l'algoritmo di backoff esponenziale binario. Le collisioni possono anche essere gestite dal protocollo MACAW.
- **PCF** (***Point Coordination Function***): prevede che ci sia una stazione centrale che coordina la trasmissione in tutte le altre. Qui non avremo collisioni, perché l'ordine delle trasmissioni è controllato. Il protocollo prevede che le stazioni si registrino con la stazione base, prima di cominciare a comunicare. La stazione base si occupa anche di regolare le temporizzazioni che avvengono durante il salto di frequenza.

La modalità DCF deve essere supportata da ogni rete wireless, mentre quella PCF è diffusa, ma opzionale. 802.11 prevede un meccanismo di attesa a tempi differenziati,

che prevede la coesistenza tra le due modalità operative.
All'interno di 802.11 esistono tre tipi di frame:

- **dati**: dedicati al trasferimento dei dati dei protocolli superiori. Questo frame è costituito da diversi campi. Abbiamo un **frame control**, che definisce la versione del protocollo, la **durata**, che specifica per quanto tempo il frame resterà nel canale. Poi, abbiamo **quattro indirizzi**. Uno specifica la destinazione, un altro la sorgente, un altro ancora la stazione base di partenza e l'ultimo la stazione di arrivo del frame. Successivamente, avremo un **campo sequenza**, che numera i frame e, infine, il campo dati e il checksum con CRC.
- **gestione**: dedicati alle funzioni di gestione della cella, quali associazione, autenticazione ed interrogazione. Questo frame ha solo due campi **address**, dato che il traffico è confinato all'interno della stazione stessa.
- **controllo**: sono i frame di ACK, RTS, CTS, le cui informazioni sono contenute nel campo **subtype**.

Le modalità operative delle reti WLAN possono dividersi ulteriormente in altre due categorie:

- **modalità ad hoc** (*infrastructureless*): i computer comunicano uno con l'altro solo grazie alla propria interfaccia di rete wireless. Si tratta di una modalità semplice da configurare, ma ideale solo per piccole reti.
- **modalità AP** (*infrastructured*): la comunicazione in rete avviene tramite un **Access Point** hardware o software, parte integrante della rete WLAN. Le reti wireless sono connesse alla rete cablata tramite una stazione che ha funzioni di bridge e di controllo. È una modalità molto più complessa da configurare, ideale per reti più grandi.

In base a queste modalità, possiamo descrivere l'802.11 come un sistema diviso in **celle**, dette **BSS** (*Basic Service Set*). Ogni cella ha il suo AP. Ogni **WT** (*Wireless Terminal*) è dentro una cella e agganciato ad un AP. Gli AP sono collegati ad un *Distribution System* (**DS**). L'insieme delle celle è detto **ESS** (*Extended Service Set*). Un WT può passare da un AP ad un altro in maniera del tutto trasparente, in un meccanismo detto **roaming**. Ogni AP invia periodicamente un ***frame beacon***, per notificare la propria presenza e altre informazioni di configurazione. Il client che riceve il frame, invia periodicamente un ***probe-request frame***, attendendosi un ***probe-response frame*** dagli AP vicini, con lo scopo di individuare destinatari di roaming.

In 802.11, l’autenticazione serve per gestire il controllo degli accessi alla rete, basato su un controllo di porta che utilizza le **PAE** (*Port Access Entity*). Viene definito un protocollo di autenticazione. Ne individuiamo due:

- *Extensible Authentication Protocol* (**EAP**): protocollo di trasporto di generici meccanismi di autenticazione tra due peer (nodi paritari).
- *Remote Authentication Dial-In User Service* (**RADIUS**): protocollo basato sul modello client/server. Anch’esso trasporta generici meccanismi di autenticazione, solo che può includere anche diversi attributi per scopi specifici.

RADIUS è senza dubbio il protocollo più utilizzato all’interno di 802.11.

NETWORK LAYER

Il **livello di rete** ha il compito di fornire allo strato superiore, quello di trasporto, un servizio per la consegna dati, in modo da mascherare l'infrastruttura della rete. Introduciamo, adesso, nuovi termini che adopereremo all'interno di questo capitolo. L'**host** sarà la stazione su cui opera il livello di trasporto, che deve trasmettere o ricevere i dati utilizzando il servizio del livello di rete. Un **pacchetto** è una tupla, composta da un header, dei dati e un trailer, che il livello di rete costruisce dal DLL. Il **router** è una stazione intermedia, che riceve i pacchetti e li inoltra. In generale, due host sono separati da un certo numero di nodi, interconnessi da svariate linee e sono possibili più tragitti tra due nodi. Lo strato di rete avrà, quindi, i seguenti compiti:

- **routing**: instradamento a livello di rete, cioè determinare quale tragitto, tra quelli disponibili, i dati dovranno seguire.
- **congestione**: evitare di sovraccaricare le linee quando sono disponibili percorsi alternativi.
- **internetworking**: risolvere i problemi legati al transito attraverso reti differenti.

Inizialmente, era previsto un modello di NL solamente connection oriented ma poi, a causa anche delle esigenze dalla rete, che richiedeva un meccanismo più flessibile, si è passati anche ad un modello connectionless. Il servizio ***connectionless*** prevede che i pacchetti siano instradati indipendentemente l'uno dall'altro. Un router dispone di una tabella, che definisce su quale linea debba essere messo il pacchetto per raggiungere la sua destinazione finale. Per quanto concerne, invece, il servizio ***connection oriented***, l'idea è quella di avere un circuito virtuale nella rete. Viene definita una sequenza di router che i pacchetti dovranno seguire. Tutti i pacchetti appartenenti alla stessa connessione, seguiranno la stessa strada. Il pacchetto, quindi, conterrà solo l'identificativo della connessione.
Parliamo, adesso, di routing, in maniera più dettagliata, essendo la funzione principale del NL. Questo processo, come abbiamo detto, permette ad un router di scegliere, tramite un algoritmo, la linea di uscita su cui instradare i dati. Sono previste due operazioni:

- **inoltro** (*forwarding*): in base all'indirizzo di destinazione o al circuito virtuale, viene scelta la linea di uscita.
- **instradamento**: creazione e aggiornamento della **tabella di routing**, che associa alla destinazione, la linea di uscita.

Per creare un buon **algoritmo di routing**, esso deve rispettare alcuni criteri. Deve essere corretto, semplice, robusto, stabile, imparziale ed ottimizzato. L'unico protocollo utilizzato al NL è il cosiddetto ***Internet Protocol*** (**IP**). Esso ha la funzione di recapitare un insieme di bit, detto **internet datagram**, dalla sorgente alla destinazione, attraverso una serie di linee interconnesse. Il recapito viene effettuato direttamente attraverso un router. Il datagram può anche essere partizionato. Un pacchetto IP è costituito da un `header` di lunghezza fissa 20 byte, più una parte opzionale. Poi, avremo un campo `version`, che contiene la versione dell'IP (IPv4, IPv6, ecc). Poi, avremo un campo `IHL`, che contiene la lunghezza dell'header in parole di 32 bit. Il campo `type-of-service` serve ad indicare diverse classi di servizio (precedenza, latenza), ignorato dal router. Il campo `total-lenght` indica la lunghezza totale del pacchetto in byte (max 65535). I campi `identification`, `DF`, `MF` e `fragment-offset` sono dedicati alla **frammentazione**. Abbiamo detto che il datagram può essere frammentato in base alle esigenze della rete. Il campo `identification` è il numero identificativo del datagram più piccolo diviso. Il campo `fragment-offset` contiene la posizione del primo byte. Questi due campi servono a ricostruire il datagram. Infine, il bit `MF` (*More Fragments*) viene impostato a `0` nell'ultimo frammento e il bit `DF` (*Don't Fragment*) viene impostato ad `1` se il datagramma non deve essere frammentato. Il campo `time-to-live` è un contatore che viene decrementato di volta in volta che un pacchetto viaggia in rete. Quindi, quando il pacchetto viene buttato via, `time-to-live` diventa `0`. Il campo `protocol` indica il protocollo di livello superiore a cui sono riservati i dati del pacchetto (UDP, TCP). Il campo `checksum` contiene un codice CRC per il controllo degli errori. I campi `source` e `destination-address` contengono informazioni relative, rispettivamente, a sorgente e destinazione. Nell'header possono anche essere inserite opzioni aggiuntive relative alla sicurezza e alle istruzioni per il router.
Per poter identificare il destinatario, ogni host e router devono avere un proprio **indirizzo IP** univoco, che distingue la rete di appartenenza. In realtà, risulta più corretto dire che ogni interfaccia di rete deve avere un proprio indirizzo IP, dato che un router può averne più di una. Per motivi di diagnostica, ogni indirizzo IP ha un'ulteriore indirizzo, detto `loopback`, che identifica sé stesso. Un indirizzo IP è costituito generalmente da **32 bit**, rappresentati come **4 numeri decimali**, con un valore

compreso tra 0 e 255, separati da un **punto** (`10.103.0.21`). È generalmente diviso in due parti: un **prefisso**, che identifica la rete, ed un **suffisso**, che identifica l'host o l'interfaccia. Possiamo raggruppare gli indirizzi IP in classi di categorie:

- **indirizzi di classe A**: tutti quelli che hanno il primo byte di valore compreso tra 0 e 127, tipo `20.9.0.200`. Solo il primo byte è destinato all'indirizzo di rete, i restanti indicano l'host.
- **indirizzi di classe B**: tutti quelli che hanno il primo byte di valore compreso tra 128 e 191, tipo `131.154.10.21`. In questo caso, i primi due byte identificano la rete, il resto l'host.
- **indirizzi di classe C**: tutti quelli che hanno il primo byte di valore compreso tra 192 e 223, tipo `193.206.144.1`. In questo caso, i primi tre byte identificano la rete, il resto l'host.
- **indirizzi di classe D**: tutti quelli che hanno il primo byte compreso tra 224 e 239. Sono tipicamente riservati per i servizi multicast.
- **indirizzi di classe E**: tutti quelli che hanno il primo byte compreso tra 240 e 255. Si tratta di una serie di indirizzi che non vengo mai effettivamente usati, ma adoperati soltanto in fase di sperimentazione.

Ovviamente, avremo anche **indirizzi speciali**. L'indirizzo che contiene tutti `0` nel campo dell'host indica quella rete (`10.0.0.0` indica la rete `10`). Potremo avere anche tutti `0` nel campo della rete, indica quella rete (se l'host `193.206.144.10` vuole inviare un pacchetto all'host `193.206.144.20`, potrà anche indicare come indirizzo `0.0.0.20`). Potremo avere anche l'indirizzo `255.255.255.255`, che rappresenta l'indirizzo broadcast della rete locale direttamente connessa.
Gli indirizzi hanno bisogno di essere assegnati ad un'unità centrale, che garantisca l'unicità. Questa unità è detta **ICANN** (*Internet Corporation for Assigned Names and Numbers*). Nonostante lo spazio di indirizzamento potesse contenere oltre due miliardi di indirizzi, è sorto comunque il problema della **carenza di indirizzi**. Purtroppo, lo spazio di indirizzamento delle classi A e B è molto vasto e nessuna rete può contenere sedici milioni di nodi diversi. Un metodo molto utile per risolvere il problema della carenza di indirizzi è il ***subnetting***. Una rete a cui è stata assegnata una classe A di indirizzi, può suddividere il suo spazio di indirizzamento in gruppi più piccoli, trattando ogni gruppo come una rete a sé stante. Per implementare queste sottoreti, è necessario introdurre un'informazione aggiuntiva, che specifichi quali bit definiscono la sottorete e

quali l'host. Per fare ciò su utilizza un bit detto **maschera**. Se vale `1`, allora il bit dell'indirizzo fa parte della rete, mentre se vale `0` fa parte dell'host. Quindi, una sottorete, dovrà avere almeno **4 indirizzi**: uno identificativo per la sottorete stessa, uno per il indicare il broadcast ed almeno due per indicare l'host.
L'indirizzamento a classi ha portato anche al problema opposto, ad un'**eccedenza di indirizzi**, cioè quando viene aumentato il numero di host connessi. Utilizzando sempre la maschera, si cerca di accorpare classi con indirizzi contigui opportuni, accorpando diverse sottoreti, in un'unica rete.
Per gestire questo schema di indirizzamento, è necessario che il router cambi radicalmente il suo modo di utilizzare la tabella di routing. Per questo è stato introdotto un nuovo standard, chiamato **CIDR** (*Classless InterDomain Routing*). Ogni record della tabella specifica l'indirizzo di destinazione con la sua maschera, in modo da superare la definizione delle classi. Ovviamente, avendo tante reti indirizzabili, le dimensioni della tabella andrebbero letteralmente ad esplodere. Per questo motivo, gli indirizzi vengono assegnati **a blocchi** alle varie organizzazioni regionali o locali, che annunciano all'esterno una sola rete (aggregazione). Abbiamo visto che un indirizzo di destinazione può corrispondere a più di una voce nella tabella e i pacchetti IP non conoscono le maschere, quindi come verrà fatto l'instradamento? Esiste la regola del ***longest-prefix-match*** quando, nel momento in cui esistono più match nella tabella, viene scelto quello con la maschera più lunga.
Col tempo, si cominciò a cercare un successore della versione IPv4. C'era la necessità di ampliare lo spazio indirizzi, in modo da potersi adattare al progresso tecnologico che si espandeva in maniera esponenziale. Per questo motivo, venne introdotto l'**IPv6**. Gli obiettivi erano sicuramente la semplicità, la sicurezza, il supporto di pacchetti di grandi dimensioni, la previsione di evoluzioni future e, soprattutto, il supporto di protocolli di livello superiore. IPv6 prevedeva l'utilizzo di **16 byte**, mostrati come otto gruppi di numeri decimali formati da quattro cifre e separati da `:`. Anche IPv6 contiene informazioni di rete ed informazioni di host. Per verificare quale parte è dedicata alla rete, si specifica la lunghezza in bit dell'indirizzo di rete dopo un `/` in coda all'indirizzo. Anche il pacchetto IPv6 è composto da un `header` di lunghezza fissa, un campo `dati` e un campo `version`. Poi, però, abbiamo anche dei campi nuovi. Il campo `traffic-class` serve ad identificare i pacchetti che necessitano di un instradamento particolare. Il campo `flow-label` identifica i pacchetti appartenenti allo stesso flusso trasmissivo e serve a supportare il traffico veloce. Il campo `payload-lenght` indica la lunghezza del pacchetto in byte. Il campo `next-header` indica il protocollo del livello di trasporto. Il campo `hop-limit` ha la stessa funzione del time-to-live e serve ad evitare che un

pacchetto resti troppo a lungo in rete se non utilizzato. Con l'eliminazione di molti campi, è stata aggiunta l'**intestazione estesa**, cioè header aggiuntivi che specificano alcune opzioni, come di destinazione, sorgente, routing, frammentazione, autenticazione, ecc.
Abbiamo, però, un problema. IPv6 può spedire, instradare e ricevere datagrammi da IPv4, ma non il contrario. Un'opzione può essere l'introduzione dei **nodi dual stack**. I nodi IPv6 posseggono anche un'implementazione IPv4, in modo da possedere entrambi gli indirizzi e scegliere quale usare. Questo può comunque comportare che due nodi IPv6 si scambino comunque datagrammi IPv4, dato che, durante il percorso, incontrando nodi IPv4, molti campi andrebbero persi. Un modo potrebbe essere mappare indirizzi IPv6 su indirizzi IPv4, in modo da saltare su host che supportino solo quelle modalità ed evitare perdite di dati.
Il protocollo **ICMP** (*Internet Control Message Protocol*) è il protocollo utilizzato per monitorare il funzionamento del livello di rete. Esistono diversi messaggi ICMP destinati ad avvisare i router di qualche specifico evento che si è verificato nella rete. Ne vediamo alcuni:

- **destinazione irraggiungibile**.
- ***time exceeded***: raggiunta la scadenza del time-to-live.
- **problema di parametri**: i parametri sono inconsistenti.
- ***source quench***: serve a controllare il flusso e a rallentare la sorgente.
- ***redirect***: c'è stato un instradamento errato.
- ***echo*** e ***echo reply***: verificare la raggiungibilità di un host.
- ***timestamp*** e ***timestamp response***: trasportano informazioni temporali per valutare il ritardo della connessione.

Il routing IP è composto da due momenti:

- ***routing control***: scambio di informazioni di routing tra i nodi della rete per recepire informazioni riguardo la tabella di routing.
- ***packet forwarding***: decisione di instradamento di ogni pacchetto, basata sul solo indirizzo di destinazione.

Il routing IP segue la politica del ***best-effort***, dove i pacchetti possono andare ritardati, duplicati, persi o distribuiti fuori ordine. Ogni router ha le proprie tabelle e scambia

informazioni con i vicini sulle strade (routes). Ci sono essenzialmente due tipi di routing IP:

- **instradamento diretto**: la trasmissione di un datagramma IP avviene su due macchine connesse alla stessa sottorete. Non sono coinvolti router intermedi. Il trasmettitore IP risolve l'indirizzo fisico dell'host del destinatario, tramite il protocollo ARP, che poi vedremo, incapsula il datagramma nell'unità dati della rete fisica e lo invia alla destinazione. Per inviare il datagramma, quindi, vengono usati i meccanismi propri della rete fisica.
- **instradamento indiretto**: l'host della destinazione non è sulla stessa rete del mittente. Il mittente deve, quindi, identificare un router a cui inviare il datagramma. Il router deve inviare il datagramma attraverso la sottorete di destinazione. Il router esamina il datagramma ricevuto e, se l'host di destinazione non si trova nella sottorete in cui il router è direttamente connesso, passa il controllo al router successivo, che lo instraderà.

Per instradare un pacchetto IP verso una destinazione appartenente alla stessa rete del mittente, esso viene incapsulato in un nuovo pacchetto, ma appartenente allo strato sottostante, il DLL. L'host, però, conosce soltanto il suo indirizzo IP e la sua rete di appartenenza. Per questo, utilizzeremo il protocollo **ARP** (*Address Resolution Protocol*). Mettendo caso che ci sia un host con indirizzo $IP1$ e indirizzo hardware $HW1$. $IP1$ vuole inviare ad $IP2$ un pacchetto IP sulla stessa rete. ARP cosa fa?

1. viene costruito un pacchetto DLL contenente $IP1$, $IP2$ e $HW1$, con un campo dedicato ad $HW2$ riempito con tutti `0`.
2. questo pacchetto viene inviato broadcast su tutta la rete.
3. tutti ricevono il pacchetto ARP, ma viene processato solo da $IP2$.
4. $IP2$ costruisce un altro pacchetto, contenente l'informazione mancante e lo invia direttamente ad $HW1$.
5. ARP lo acquisisce e $IP1$ può instradare correttamente il proprio pacchetto.

Per aumentare le prestazioni di ARP, si può introdurre una cache veloce, che memorizza le associazioni temporanee tra IP e hardware. Una versione di ARP può essere Reverse ARP, che permette di effettuare questa operazione a partire dall'indirizzo fisico.
Una volta parlato del NL in generale, ci tocca concentrarci maggiormente sul routing, di

cui abbiamo discusso soltanto in maniera blanda. Un router può interconnettere reti che usano diverse tecnologie, inclusi mezzi fisici differenti. Il collegamento di più reti sotto un unico dominio amministrativo prende il nome di **AS** (*Autonomous System*). I router che instradano messaggi all'interno dello stesso AS e non hanno diretta connessione con le altre reti sono detti ***Interior Routers*** e scambiano informazioni tramite un protocollo **IGP** (*Interior Gateway Protocol*). I router che instradano messaggi tra AS diversi sono detti ***Exterior Routers*** e si scambiano informazioni tramite un protocollo **EGP** (*Exterior Gateway Protocol*). I routers che fungono da ponte di collegamento tra AS diversi vengono detti ***Border Routers***. Prima abbiamo parlato di tabella di routing, ma ora vediamo nel dettaglio i record di cui essa è composta:

- **indirizzo IP di destinazione**: quando un router riceve un pacchetto dati, controlla nella propria tabella di routing se esiste una entry per tale destinazione e, in caso affermativo, inoltra il flusso dati.
- **metrica**: definisce l'algoritmo di routing, che poi vedremo meglio, per selezionare il percorso migliore.
- **indirizzo del router di next hop**: l'indirizzo del router successivo, utile a raggiungere la rete di destinazione.
- **interface**: interfaccia del router attraverso cui deve essere instradato il pacchetto verso il next-hop.
- **timer**: scandisce temporalmente ogni quanto inviare gli updates ai router vicini.

Esiste un router particolare, detto **gateway**, dove è inviato il traffico diretto ad una destinazione non presente nella tabella di routing. Ci sono due principali strategie di instradamento:

- **routing statico**: prevede il calcolo dei percorsi offline quando la rete non è ancora attiva e la loro configurazione è manuale, a carico dell'operatore. Viene utilizzato in reti poco complesse, in cui c'è al massimo un'unica connessione
- **routing dinamico**: i percorsi cambiano dinamicamente in base alle situazioni di traffico e ad altre informazioni locali come congestione, guasti.

Di particolare importanza per gli algoritmi di routing risulta essere il **principio di ottimalità**. Se un router J si trova su un percorso ottimale che collega i router I e K, allora il percorso ottimale da J a K è incluso in quello da I e K. Se non lo fosse, il percorso da I a K includerebbe una tratta, da J a K, non ottimale e, quindi, non

sarebbe ottimale nel complesso. Il principio porta alla definizione del ***sink tree***. Si tratta di un albero che collega tutti i possibili router sorgenti ad uno dato come destinazione. I sink tree sono determinati ed utilizzati da tutti gli algoritmi di routing. Il principio di ottimalità e il sink tree permettono di valutare le prestazioni di ogni singolo algoritmo di routing. Vediamo, adesso, alcuni algoritmi di routing più utilizzati:

- ***Shortest Path Routing***: si costruisce un grafo della sottorete, con i nodi che sono i router e gli archi i collegamenti, instradando lungo il percorso più breve. Il concetto di percorso più breve si determina secondo diversi criteri, come la distanza geografica, la banda disponibile, il traffico medio, ecc. Si tratta di un algoritmo statico. SPR definisce soltanto una classe di algoritmi, di cui l'algoritmo più noto è quello di **Dijkstra**. Qui, per trovare il percorso più breve fra un nodo A e un nodo D , si parte da uno dei due, definito come `working-node`, e i suoi adiacenti si etichettano con `<distanza cumulativa>` e `<nodo precedente>`. Si esaminano tutti i nodi etichettati, selezionando quello con distanza minore, che diventerà il nuovo `working-node`.
- ***Flooding***: ogni pacchetto in arrivo è inviato su tutte le linee del router corrente, esclusa quella in arrivo. Con il *flooding* si genera molto traffico e, quindi, ci sono alcune strategia per limitarlo. Potremo inserire un **numero di salti** all'interno del pacchetto, che viene decrementato ad ogni transito da un router e, quando arriva a zero, il pacchetto viene eliminato. Un'altra strategia potrebbe essere quella di inserire un **numero di sequenza** all'interno del pacchetto. Ogni router mantiene una lista di tutti gli altri router, memorizzando i numeri di sequenza già trasmessi. Quando un router Y, riceve un pacchetto da un router X, controlla nella propria tabella l'entry X e, se il numero di sequenza corrisponde, allora lo scarta. Si può anche adottare una variante di questo algoritmo, detto **flooding selettivo** che, sfruttando la conoscenza della topologia di rete, inoltra i pacchetti non in tutte le direzioni, ma solo in quelle presumibilmente corrette. Anche questo algoritmo è un algoritmo statico. È primitivo, ma molto robusto e viene usato in sistemi che richiedono un'alta tolleranza agli errori.
- ***Distance Vector Routing***: si tratta di un algoritmo dinamico, in cui ogni router X possiede un vettore delle distanze stimate verso ogni destinazione. La destinazione può essere un router **adiacente**, per il quale si suppone che X conosca la distanza e un router **lontano** Z. In questo caso, X riceve N vettori di distanze dai suoi N router adiacenti. X calcola N distanze, ognuna pari alla somma della distanza

$X - adiacente$ con la distanza $Z - adiacente$. La minore fra le N è la distanza inserita da X nel vettore. Questo algoritmo è un algoritmo di facile implementazione, ma è sconsigliato per le reti di grandi dimensioni, perché lento a convergere (scambiare informazioni tra i router lontani). Reagisce bene ai miglioramenti, ma moto lentamente ai peggioramenti. Un problema molto comune di DVR è il **routing loop**. In caso di rottura del link tra due router, il problema potrebbe non venir subito diagnosticato e, quindi i router che comunicano aggiornerebbero comunque la propria tabella, riportando, però, un costo fasullo ed entrando in loop. Una prima tecnica per risolvere questo problema può essere la ***Split Horizon***, che evita di inviare informazioni di costo, quindi evita che si formi il loop. Per questo, è possibile aggiungere l'estensione di ***Reverse Poison***, cioè il router invia le informazioni di costo, ma con una metrica infinita, comunicando che, quindi, quella destinazione è irraggiungibile e, quindi, il loop non si verificherà.

- ***Link State Routing***: Il DVR che abbiamo visto non tiene conto della banda e presenta il problema del conteggio all'infinito. Per questo motivo, venne introdotto questo nuovo algoritmo, che si presenta in cinque punti. Il primo è la **scoperta dei router vicini** ed occorre che ogni router abbia un identificativo univoco. Il secondo è la **misurazione della distanza**, dove si invia un pacchetto speciale `ECHO`, che richiede risposta al router vicino e, poi, si divide per due il tempo totale impiegato, tenendo in considerazione anche il carico. Il terzo è la **costruzione di pacchetti informativi**, dove viene generato un pacchetto ad ogni router con lo stato dei link dei suoi vicini. Il quarto è l'**invio di pacchetti informativi** dove, tramite *flooding*, si limita il traffico con i numeri di sequenza. Il quinto e ultimo è l'**elaborazione dei percorsi** dove, ricevuti i pacchetti informativi, tramite Dijkstra si trova localmente il percorso più breve.
- **Routing Gerarchico**: la dimensione delle tabelle di routing può essere eccessiva. In tal caso, la rete viene organizzata in **regioni gerarchiche**. La tabella di routing centrale aveva un'entry per ogni destinazione, mentre la entry della tabella gerarchica prevede informazioni per i router locali, più altre relative alle regioni gerarchiche. Il principale svantaggio di questo algoritmo è che, perdendo la visibilità del singolo nodo, alcuni percorsi potrebbero non essere ottimali.

Il primo protocollo di routing che andremo a vedere è **RIP** (*Routing Information Protocol*), basato sull'algoritmo DV. È un protocollo adatto a reti di piccole dimensioni e, quindi, quasi in disuso. Per questo motivo, venne sviluppato un successore, detto

OSPF (*Open Short Path First*). Esso doveva rispettare diversi requisiti. Non ci doveva essere nessun vincolo di brevetto, doveva supportare diversi metriche per la distanza, doveva essere capace di reagire dinamicamente ai cambiamenti della topologia di rete, doveva poter bilanciare il carico e doveva essere sicuro. OSPF supporta tre tipi di reti: punto-a-punto tra due router, multiaccesso con trasmissione broadcast (LAN) e multiaccesso senza trasmissione broadcast (WAN). OSPF traduce la rete in un grafo pesato orientato, da dove poi si calcola il percorso più breve. La rete viene divisa in **aree** che non si sovrappongono. L'area 0 di ogni rete è chiamata **dorsale** (*backbone*), a cui tutte le altre sono collegate. L'area, inoltre, permette di classificare quattro tipi di router:

- **router interni**, dislocati in un'area.
- **router di confine**, se collegano due o più area.
- **router di dorsale**, se sono sulla dorsale.
- **router di confine della rete**, se collegano due o più reti.

Le quattro classi possono sovrapporsi.
Prima abbiamo visto anche l'EGP, protocollo per lo scambio di informazioni tra AS. A sua volta, EGP contiene il protocollo **BGP** (*Border Gateway Protocol*), in cui due AS si scambiano informazioni di instradamento designando due router che stabiliscono una sessione di ***peering***. Il BGP ha un funzionamento simile al DV. Ci sono alcuni attributi obbligatori da includere all'interno della comunicazione:

- `ORIGIN`: un protocollo di tipo IGP, da cui proviene l'informazione che usa l'AS sorgente.
- `AS_PATH`: lista di AS attraversati.
- `NEXT_HOP`: prossimo router da usare lungo il percorso.

Ogni AS ha un id univoco. Inizialmente viene aperta una connessione tra peers, di tipo TCP, poi vengono forniti degli annunci, che contengono informazioni sulla raggiungibilità e, infine, si verifica il corretto funzionamento.
BGP ha alcuni svantaggi. Sceglie i due percorsi possibili in base alla metrica di costo e non può discriminare sulla base della distanza o della congestione e poi, è molto difficile far configurare a tutti i router un meccanismo di annuncio sulla raggiungibilità.

TRANSPORT LAYER

Il livello di trasporto fornisce la **comunicazione logica** tra processi applicativi ed host differenti. I protocolli di trasporto vengono eseguiti nei sistemi terminali. Il TL, in invio, scinde i messaggi in **segmenti** e li passa al livello di rete mentre, in ricezione, riassembla i segmenti in messaggi e li passa al livello superiore, quello applicativo. Generalmente, esistono due protocolli principali di trasporto, che sono **TCP** (*Transmission Control Protocol*) e **UDP** (*User Datagram Protocol*), che però vedremo meglio nel dettaglio. NL e TL sono strettamente correlati. Il NL fornisce comunicazione logica tra host, mentre il TL tra processi, basandosi sui servizi forniti dal livello di rete. Si potrebbe fare un'analogia con un moderno sistema postale. Ci sono 12 ragazzi che inviano lettere a 12 ragazzi. I processi sono i ragazzi, i messaggi delle applicazioni sono le buste, l'host sono le case, Anna e Andrea sono i protocolli di trasporto e l'ufficio postale è il protocollo del livello di rete. Abbiamo nominato prima TCP e UDP. Diremo che TCP è il protocollo più affidabile, che garantisce consegne nell'ordine originario, con annesso controllo di congestione, di flusso e setup della connessione. Al contrario, UDP effettua consegne senz'ordine e non è particolarmente affidabile. Entrambi i protocolli non garantiscono i ritardi e l'ampiezza di banda.
Un servizio molto importante del TL è il **multiplexing** e il **demultiplexing**. Il TL riceve segmenti dal NL sottostante. Abbiamo detto che il TL ha il compito di consegnare questi segmenti al livello applicativo. Ricordiamo che un processo può presentare una o più **socket**, che sono delle porte attraverso cui fluiscono i dati dalla rete al processo. Il compito di trasportare i dati dei segmenti a livello di trasporto verso la giusta socket, viene detto demultiplexing. Il compito di raggruppare le porzioni di dati presso l'host di origine a partire da diverse socket, incapsularle con informazioni di intestazione per formare i segmenti e passarle al livello di rete, viene detto multiplexing.
Nel demultiplexing, l'host riceve il datagramma IP, che è composto da un indirizzo IP di origine e un indirizzo IP di destinazione. Ogni datagramma porta un segmento a livello di trasporto. Ogni segmento ha un numero di porta di origine e un numero di porta di destinazione. L'host, poi, usa gli indirizzi IP e i numeri di porta per inviare il segmento alla socket appropriata. Il demultiplexing può essere effettuato sia senza connessione (UDP) che con connessione (TCP).

Parliamo, ora, più nel dettaglio, del protocollo senza connessione UDP. Si tratta di un protocollo di trasporto **senza fronzoli**, con un servizio di consegna a **massimo sforzo**. I pacchetti di questo protocollo sono detti datagrammi utente e, a causa dell'inaffidabilità, possono essere perduti o consegnati fuori sequenza. Senza connessione, significa anche nessun handshaking per favorire la connessione e quindi consegna indipendente. UDP è ormai in disuso, ma spesso veniva preferito per la sua semplicità, per il fatto che non fosse necessario stabilire una connessione e che non fosse necessario controllare la congestione. UDP viene utilizzato nelle **applicazioni multimediali**, che tollerano piccole perdite. La struttura di un datagramma utente è molto semplice. C'è la `source port` e la `destination port`, `lenght`, che indica la lunghezza in byte, una `checksum` per rilevare gli errori e, infine, la sezione dei dati. I protocolli più comuni di UDP sono:

- **BOOTP** (*Bootstrap Protocol*): processo di avvio dei computer o del sistema operativo.
- **DHCP** (*Dynamic Host Configuration Protocol*): ricevere automaticamente la configurazione IP per instaurare la connessione.
- **DNS** (*Domain Name Server*): assegnare nomi ai nodi della rete.
- **TFTP** (*Trivial File Transfer Protocol*): trasferimento file di livello applicativo molto semplice.

Adesso, ci tocca parlare nel dettaglio del protocollo TCP, che è sicuramente quello più usato. TCP ha un'architettura punto-a-punto, cioè un mittente ed un destinatario. Il flusso di byte arriva in sequenza e garantisce affidabilità. Il controllo di flusso e di congestione definiscono la dimensione della finestra. È una trasmissione full-duplex, cioè bidirezionale. TCP è orientato alla connessione, cioè avviene l'handshaking che inizializza lo stato del mittente e del destinatario prima di scambiare i dati. I pacchetti in TCP vengono detti segmenti e hanno molti più attributi. Avremo sempre la `source port` e la `destination port`. Poi avremo il `sequence number`, il valore del primo byte, e l'`acknowledge number`, cioè il valore dell'ultimo byte. Il campo `URG` indicherà i dati urgenti, `ACK` il numero di riscontro valido, `PSH` che indica che i dati devono essere inviati immediatamente, mentre `RST`, `FIN` e `SYN` servono a impostare la chiusura della connessione. Poi avremo `window size`, che indica il numero di byte che il destinatario intende accettare, il `checksum` per il controllo degli errori e poi altri **campi opzionali** per la lunghezza, il tipo e i dati relativi.
TCP supporta il **controllo del flusso**. Il mittente non vuole sovraccaricare il buffer del

destinatario trasferendo troppi dati troppo velocemente. Quindi, la frequenza d'invio deve corrispondere alla frequenza di lettura dell'applicazione ricevente. Il destinatario comunica lo spazio disponibile includendo il campo `RcvWindow` nei segmenti. Il mittente limita i dati non riscontrati a `RcvWindow`, garantendo che il buffer in ricezione non vada in overflow.
Con TPC abbiamo detto che c'è bisogno che mittente e destinatario instaurino una **connessione**. Le variabili TCP relative ai numeri di sequenza, ai buffer e alle informazioni per il controllo del flusso. Si utilizza il processo dell'**handshake a tre vie**:

1. Il client invia un segmento `SYN` al server, specificando solo il numero di sequenza iniziale.
2. Il server riceve il `SYN` e risponde con un segmento `SYN+ACK`. Il server alloca i buffer e specifica il numero di sequenza iniziale.
3. Il client riceve `SYN+ACK` e risponde con un segmento `ACK`, che contiene i dati.

Esiste un meccanismo anche per la **disconnessione**. Diciamo che la chiusura deve **simmetrica**, cioè concordata da entrambe le parti, altrimenti non verrebbe effettuata nel modo corretto. Vediamo i vari punti:

- Il client invia al server un segmento `FIN`.
- Il server riceve il segmento `FIN` e risponde con un `ACK`. Chiude la connessione e invia un `FIN`.
- Il client riceve il `FIN` e risponde con un `ACK`.
- A questo punto, il server riceve l'`ACK`, i due sono sincronizzati e la connessione viene chiusa.

Un altro importante compito del TCP è la gestione delle **congestioni**. Si verifica una connessione quando troppe sorgenti trasmettono troppi dati, ad una velocità talmente elevata che la rete non è in grado di gestirli e, tutto ciò porta a smarrire i pacchetti e a lunghi ritardi. La congestione si può verificare in tre scenari. Quando abbiamo due mittenti, due destinatari, un router e un buffer illimitato, dove non si verificherà nessuna ritrasmissione. Oppure potremo avere un router e buffer finiti. In questo caso, se il buffer è pieno, i pacchetti verranno scartati e il mittente lo ritrasmetterà solo se la connessione è affidabile. Infine, potremo avere un terzo scenario, dove ci sono quattro mittenti, percorsi multihop a timeout e i link dei router hanno alta capacità. In questo caso, ci

troveremo di fronte a del traffico basso.
Esistono due principali approcci per il controllo della congestione:

- **punto-a-punto**: nessun supporto specifico alla rete. La congestione è dedotta osservando le perdite e i ritardi nei sistemi terminali. TCP di solito usa questo metodo.
- **assistito dalla rete**: i router forniscono un feedback ai sistemi terminali, attraverso un singolo bit

Si potrebbe aggiungere un altro campo, detto **finestra di congestione** (`CongWin`), che impone una limitazione addizionale alla quantità di traffico che un host può inviare in una connessione.
TCP implementa diversi protocolli noti:

- **HTTP** (*Hypertext Transfer Protocol*): usato come principale sistema per la trasmissione d'informazioni sul web.
- **HTTPS** (*Hypertext Transfer Protocol over Secure Socket Layer*): come il precedente, ma con una trasmissione più sicura.
- **FTP** (*File Transfer Protocol*): per il trasferimento di file.
- **SMTP** (*Simple Mail Transfer Protocol*): per la trasmissione di email.

APPLICAZIONE

Vediamo, adesso, i successivi tre livelli di cui si compone il modello ISO/OSI, ma ne parleremo più nello generico, soffermandoci, invece, su come si crea un'applicazione di rete vera e propria.
Il **livello di sessione** consente di stabilire una sessione tra applicazioni, ovvero uno scambio di informazioni temporaneo e interattivo tra due o più dispositivi di comunicazione. Offre funzionalità di **controllo del dialogo**, **gestione dei token** e **sincronizzazione**. Una sessione è uno stato, dove almeno una delle parti comunicanti deve conservare informazioni su di esso e sulla cronologia per poter continuare. È possibile avere sessioni anche in altri livelli del modello.
Il **livello di presentazione** si occupa di trasformare i dati forniti dal livello di applicazione in un formato standard e di offrire servizi di comunicazione comuni come la **crittografia** o la **compressione**. Può avere una sintassi **astratta**, definizione formale dei dati che gli applicativi si scambiano, **concreta locale**, come i dati sono rappresentati localmente, o di **trasferimento** (come i dati sono codificati durante il trasferimento).
Il **livello di applicazione** deve interfacciarsi e fornire servizi per i processi delle applicazioni, interagendo con uno dei protocolli di livello di trasporto per ricevere dati o inviarli passandoli nella
forma richiesta.
Creare un'applicazione di rete significa creare software in grado di funzionare su più macchine (posta elettronica, web). Un'applicazione di rete può avere diverse architetture:

- **client/server**: il server è un host sempre attivo, con un indirizzo IP fisso, mentre il client comunica con il server e può contattarlo in qualsiasi momento. Può avere un indirizzo IP dinamico e non comunica direttamente con gli altri client.
- **P2P**: non c'è un server sempre attivo, ma coppie arbitrarie di host che comunicano direttamente tra loro. I peer non devono necessariamente essere sempre attivi.
- **ibridi** (client/server e P2P): es. messaggistica istantanea.

Prima di andare avanti, chiariamo ancora alcuni concetti che già abbiamo espresso. I **processi comunicanti** sono processi su host differenti che comunicano scambiandosi

messaggi. Un processo riceve e invia messaggi attraverso la sua **socket**, una porta. Affinché un processo su un host invii un messaggio a un processo su un altro host, il mittente deve identificare il processo destinatario.
A proposito di ciò, introduciamo un argomento che abbiamo solo accennato, cioè il **server DNS** (*Domain Name System*). Il server DNS è quella parte dell'applicazione che ha il compito di stabilire la corrispondenza tra un indirizzo logico e l'indirizzo IP di un host. Si tratta propriamente di un database distribuito che permette di convertire i nomi simbolici degli host negli indirizzi IP numerici. È più facile ricordare **www.google.com**, piuttosto che `64.233.167.99`. Per il client, il server DNS è come una **scatola nera**. Viene inviato un messaggio di richiesta al DNS, specificando l'hostname che deve essere trasformato in indirizzo IP. Dopo un ritardo variabile, il client riceve un messaggio di risposta DNS che fornisce la correlazione desiderata. Oltre alla traduzione degli hostname in indirizzi IP, il DNS offre anche servizi di ***host aliasing***, cioè quando un host con un nome complicato può avere nomi più semplici. Possiamo distinguere tre tipi di DNS:

- **DNS locale**: ciascun provider ha un DNS locale. La richiesta inviata da un host viene prima mandata al server dei nomi locali.
- **DNS root**: quando il DNS locale non può soddisfare immediatamente la richiesta di un host, si comporta come un client DNS ed invia la richiesta al DNS root. Il root server potrebbe aver registrato l'hostname richiesto e potrebbe inviarlo al DNS locale che poi gira la risposta all'host richiedente. Oppure, potrebbe anche non aver registrato l'hostname, ma conoscerebbe il suo indirizzo IP e, quindi, la sua correlazione.
- **DNS assoluto**: ha sempre un record DNS che traduce l'hostname in IP per quel host. Nel caso precedente, quando abbiamo parlato che il DNS potrebbe non aver registrato l'hostname, in questo caso conoscerebbe l'indirizzo IP e la correlazione andando a consultare proprio il DNS assoluto.

Il database distribuito del DNS può essere interrogato in due modi:

- **query ricorsive**: il client propaga la richiesta al DNS di livello superiore che gestisce la stessa negoziando con gli altri DNS nella gerarchia. La risposta, quindi, viene propagata all'indietro.
- **query iterative**: la negoziazione con i vari DNS viene gestita direttamente dal client. Quando un DNS A fa una richiesta ad un DNS B, se esso non ha la

correlazione, invia la propria risposta ad A, che indica l'indirizzo IP del DNS successivo nella catena, un ipotetico DNS C. E così via.

Possiamo distinguere anche altri due tipi di DNS particolari:

- **Server TLD** (*top-level-domain*): si occupano dei domini `.com`, `.org`, `.net`, `.edu` e di tutti i nomi locali di alto livello `.it`, `.uk`, `.fr`.
- **Server di competenza** (*authoritative server*): ogni organizzazione dotata di host Internet pubblicamente accessibili deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP.

Una figura particolarmente importante all'interno del mondo del DNS è il **resolver** che, generalmente, è il client per l'interrogazione al DNS. In maniera più tecnica, resolver è una procedura che viene chiamata nel momento in cui si ha intenzione di effettuare un'interrogazione al DNS. Il protocollo di comunicazione fra resolver e DNS è del livello applicativo ed è basato su scambio di pacchetti UDP e TCP. Un'altra componente principale è lo **spazio dei nomi**, cioè come sono organizzati i record all'interno del DNS. In linea di massima, vengono scritti in **modo gerarchico**. Il **dominio di primo livello** identifica la nazione di appartenenza oppure la categoria a cui appartiene la società proprietaria dell'host. Il **dominio di secondo livello** indica, di solito, una singola organizzazione. I segmenti successivi indicano domini di livelli inferiori che servono ad individuare lo **specifico host**. I nomi sono stringhe, lunghe fino a 63 caratteri. Il nome può indicare un dominio assoluto (**FQDN**, *fully qualified domain name*) oppure un dominio parziale **PQDN** (*partially qualified domain name*). Dal punto di vista sintattico, si tratta di una sequenza di segmenti alfanumerici separati da un punto. Parlando in maniera più tecnica, i nomi vengono salvati all'interno del DNS secondo lo schema **RR** (*Resource Record*), con un formato `RR = (name, value, type, ttl)`. Anche un messaggio DNS ha un formato particolare. Esso prevede un **header** iniziale che è composto da vari campi. Il campo `identification` permette al client di accoppiare la risposta ricevuta alla richiesta inviata. Il campo `flag` indica il tipo di messaggio (domanda, risposta, ricorsione disponibile, ecc). Poi, avremo altri quattro campi `number`, che indicano il numero di presenze di quattro tipi di sezioni (domande, risposte, ecc). Il campo `question` indica il nome richiesto e il tipo di domanda. Il campo `answer` contiene la risposta alla domanda in formato RR. Il campo `competence` indica i record per il server di competenza. E poi, avremo un ulteriore campo per informazioni aggiuntive.
Di particolare importanza per un'applicazione di rete, risulta essere la **posta elettronica**. Essa ha varie componenti principali. La prima è un **agente utente**, detto

anche *mail-reader*, che sono programmi per leggere e gestire la posta e le **mailboxes** (Outlook, Messenger). I messaggi in uscita o in arrivo vengono memorizzati sul **server di posta**. Esso include una casella di posta, messaggi in arrivo, e una coda di messaggi, in attesa di essere trasmessi. Utilizza il protocollo **SMTP** (*Simple Mail Transfer Protocol*). SMTP, a sua volta, utilizza il protocollo TCP per trasferire i messaggi in modo affidabile dal client al server. Ci sono tre tipi di trasferimento: l'**handshake**, il **trasferimento** e la **chiusura**. Anche se i server di posta utilizzano SMTP per inviare e ricevere mail, i client mail a livello utente utilizzano SMTP solo per inviare il messaggio. Per recuperare i messaggi, le applicazioni client usano solitamente protocolli come **IMAP** o **POP**. Il server di riferimento per l'invio della posta è detto ***relay agent*** e tutti i client inviano la posta al relay che la invia al destinatario. Gli indirizzi di posta elettronica hanno il formato `utente@host.dominio`. Gli indirizzi sono risolti dal DNS che individua il server a cui inviare il messaggio. Il server di posta riceve i messaggi e li accoda nella mailbox dell'utente, che di solito è un file di testo in una directory specifica del server. Il formato dei messaggi è molto semplice. C'è un'**intestazione** (da, a, oggetto), una riga vuota, e un **corpo del messaggio**. È possibile anche aggiungere altri campi di intestazione, usando estensioni provenienti dallo standard **MIME** (*Multipurpose Internet Mail Extensions*).
Passiamo, adesso, ad un concetto davvero fondamentale all'interno di una rete: il **web**. Il web è una collezione di documenti ipertestuali, distribuita su vari server e collegata ad Internet. Si tratta di un grafo i cui nodi sono i singoli documenti collegati fra loro da puntatori (***hyperlink***). La maggior parte dei documenti sono in formato **HTML** (*HyperText Markup Language*). Nel 1991 il CERN definì il protocollo **HTTP** (*HyperText Transfer Protocol*), che permette la lettura ipertestuale e non sequenziale dei documenti. Il protocollo HTTP ha visto nascere il **WWW** (*World Wide Web*), la grande ragnatela mondiale. Una **pagina web** è il modo in cui vengono rese disponibili all'utente finale le informazioni del WWW della rete Internet. Un insieme di pagine web relazionate tra loro è un **sito web**. Una pagina web ha uno o più **URL** (*Universal Resource Locator*), una sequenza di caratteri, che ne permette l'individuazione. Ogni URL ha generalmente sei parti:

- **protocollo**: descrivere il protocollo di comunicazione (HTTP, HTTPS).
- **username** o **password** (opzionali): è possibile, in alcuni casi, specificare l'autenticazione di accesso alla risorsa.
- **hostname**: indirizzo fisico del server.

- **porta** (opzionale): quando il processo server è in ascolto su una porta non conforme allo standard, si specifica questo parametro.
- **percorso** (opzionale): pathname nel file system del server che identifica la risorsa.
- **querystring** (opzionale): si separa dall'URL con ? e consente di passare al server più parametri.

Le pagine web vengono lette da un **browser**, che è in grado di interpretare l'HTML e far visualizzare la pagine così com'è stata progettata da uno sviluppatore (Internet Explorer, Chrome, Safari).
Parliamo, adesso, più nel dettaglio, di HTTP. Esso definisce il protocollo di comunicazione tra client e server e può essere usato per trasferire qualsiasi tipo di risorsa. Il messaggio è composto da una **linea iniziale**, uno o più **header**, una **linea vuota** e un **corpo del messaggio**.
La linea iniziale può contenere un **metodo**, un **percorso locale** e la **versione**. I metodi più importanti sono:

- `GET`: chiede il trasferimento di una risorsa.
- `HEAD`: chiede il trasferimento delle intestazioni della richiesta.
- `POST`: invia i dati da elaborare al server.

La linea iniziale può anche contenere un **codice di stato**, per esempio `200`, che indica che tutto è andato correttamente o `404`, che indica la risorsa non esistente. Di particolare importanza è un **proxy HTTP**, che agisce da intermediario tra client e server.
Le connessioni HTTP possono essere di due modi:

- **non persistenti**: almeno un oggetto viene trasmetto su una connessione TCP.
- **persistenti**: più oggetti possono essere trasmessi su una singola connessione TCP.