

PISSIR  
Progettazione e Implementazione di Sistemi Software in Rete

Alessandro Zappatore

Università del Piemonte Orientale  
Anno accademico 2024/2025, 1° semestre

## Indice

<b>1</b>	<b>Il livello Data Link e le LAN</b>	<b>4</b>
1.1	Il data link layer . . . . .	4
1.1.1	I servizi forniti . . . . .	4
1.1.2	Implementazioni . . . . .	4
1.2	Individuazione e correzione degli errori . . . . .	4
1.2.1	Controllo di parità . . . . .	4
1.2.2	CRC (Cyclic Redundancy Check) . . . . .	5
1.3	Protocolli ad accesso multiplo . . . . .	6
1.3.1	Protocolli a partizionamento del canale . . . . .	7
1.3.2	Protocolli ad accesso casuale . . . . .	8
1.3.3	Protocolli a turni . . . . .	11
1.4	Switched Local Area Networks . . . . .	11
1.4.1	Indirizzo MAC . . . . .	11
1.5	Protocollo ARP (Address Resolution Protocol) . . . . .	12
1.6	Ethernet . . . . .	14
1.7	Switch Ethernet . . . . .	14
1.7.1	Interconnessione tra switch . . . . .	15
1.8	Switch vs Router . . . . .	16
1.9	VLAN (Virtual LAN) . . . . .	16
1.10	VLAN spanning (trunk port) . . . . .	17
1.11	Networking nei data center . . . . .	18
1.11.1	Architetture dei data center . . . . .	18
1.11.2	Load balancing . . . . .	19
1.12	Il processamento di una richiesta . . . . .	19
1.12.1	Connessione a Internet . . . . .	19
1.12.2	Richiesta HTTP . . . . .	20
<b>2</b>	<b>Livello fisico</b>	<b>22</b>
2.1	Segnali a larghezza di banda limitata . . . . .	22
2.1.1	Data rate massimo per un canale . . . . .	22
2.2	Modulazione digitale . . . . .	23
2.2.1	Trasmissione baseband . . . . .	23
2.2.2	Trasmissione passband . . . . .	24
<b>3</b>	<b>Reti wireless e mobili</b>	<b>28</b>
3.1	Elementi di una rete wireless . . . . .	28
3.1.1	Modalità di funzionamento . . . . .	28
3.1.2	Tipi di connessioni e infrastruttura . . . . .	28
3.2	Caratteristiche dei collegamenti wireless . . . . .	29
3.2.1	CDMA (Code Division Multiple Access) . . . . .	30
3.3	Reti WLAN 802.11 . . . . .	31
3.3.1	Architettura . . . . .	31
3.3.2	Canali e associazioni . . . . .	32

3.3.3	Accesso multiplo al canale . . . . .	32
3.3.4	Soluzione alternativa (RTS-CTS) . . . . .	33
3.3.5	Frame 802.11 . . . . .	33
3.3.6	Mobilità nella stessa sotto-rete IP . . . . .	34
3.3.7	Caratteristiche avanzate di 802.11 . . . . .	34
3.4	Bluetooth 802.15 . . . . .	34
3.4.1	Topologia (Piconet) . . . . .	34
3.4.2	Scatternet . . . . .	35
3.4.3	Comunicazioni Bluetooth . . . . .	35
3.4.4	Medium Access Control . . . . .	35
3.4.5	Modalità . . . . .	35
3.5	ZigBee 802.15.4 . . . . .	36
3.5.1	Topologia . . . . .	36
3.5.2	Architettura . . . . .	36
3.5.3	Medium access control . . . . .	37
3.6	Z-Wave . . . . .	38
3.7	Reti cellulari 4G e 5G . . . . .	39
3.7.1	Elementi di una rete 4G . . . . .	39
3.7.2	Protocollo LTE . . . . .	40
3.7.3	Separazione piano controllo e dati . . . . .	40
3.7.4	Stack data plane . . . . .	40
3.7.5	Reti 5G . . . . .	42
3.8	Gestione della mobilità . . . . .	42
3.8.1	Routing indiretto . . . . .	43
3.8.2	Routing diretto . . . . .	43
3.9	Mobilità nelle reti 4G . . . . .	44
3.9.1	Configurazione degli elementi del <i>control-plane</i> LTE . . . . .	44
3.9.2	Configurazione dei tunnel del <i>data-plane</i> . . . . .	44
3.9.3	Handover all'interno della stessa rete . . . . .	45
3.9.4	Mobile IP . . . . .	46
3.9.5	Impatto sui protocolli di livello superiore . . . . .	46
<b>4</b>	<b>Architettura a microservizi</b>	<b>47</b>
4.1	Cos'è un'architettura software . . . . .	47
4.2	Modello 4+1 viste . . . . .	47
4.3	Stile architetturale a strati (Layers/Tiers) . . . . .	50
4.3.1	Differenza tra layer e tier . . . . .	50
4.4	Architettura a esagoni . . . . .	50
4.5	Svantaggi dell'architettura monolitica . . . . .	51
4.5.1	Sviluppo di un architettura monolitica . . . . .	52
4.6	Traduzione all'architettura a microservizi . . . . .	52
4.6.1	Sviluppo di un architettura a microservizi . . . . .	53
4.6.2	Scalabilità . . . . .	54
4.6.3	Quando scegliere un'architettura a microservizi . . . . .	55
4.7	Da request driven a event driven . . . . .	55
4.7.1	Evento . . . . .	56
4.7.2	Benefici di event-driven . . . . .	56
4.8	Positività dei microservizi . . . . .	56
4.8.1	Negatività dei microservizi . . . . .	56
4.9	Definire un'architettura a microservizi . . . . .	56
4.9.1	Cos'è un servizio . . . . .	56
4.9.2	Come fare . . . . .	57
4.9.3	Identificazione delle operazioni di sistema . . . . .	57
4.9.4	Definizione delle operazioni di sistema . . . . .	59
4.9.5	Definire le query . . . . .	60
4.9.6	Decomposizione delle business capabilities . . . . .	61

4.9.7	Decomposizione da sotto domini (DDD) . . . . .	61
4.9.8	Principi della decomposizione . . . . .	62
4.9.9	Mappatura delle operazioni e servizi . . . . .	62
4.9.10	Identificazione delle collaborazioni . . . . .	63
4.9.11	Scelta del paradigma di comunicazione . . . . .	63

# 1 Il livello Data Link e le LAN

## 1.1 Il data link layer

I dispositivi (host, router, server ecc.) che supportano il data link layer sono detti **nodi**. La comunicazione avviene tramite **canali** che connettono nodi adiacenti, questi canali vengono chiamati **link** (via cavo, wireless); un'intera rete LAN viene vista come un unico link. I pacchetti che vengono trasmessi a livello data link si chiamano **frame**.

### Richiami dell'esame di reti

A livello IP il pacchetto trasmesso è chiamato **datagramma**, a livello trasporto TCP il pacchetto è chiamato **segmento**.

### 1.1.1 I servizi forniti

- **Incapsulamento (framing)**: Nella maggior parte dei casi la parte contenente i dati del frame contiene il datagramma IP. Il livello data link fornisce come servizio quello di trasferire un datagramma da un nodo a un altro attraverso i link.
- **Accesso al link**: Un protocollo di medium access control (MAC) specifica come il frame deve essere trasmesso sul link.
- **Trasporto affidabile**: Garantisce che ogni frame raggiunga la sua destinazione senza errori. Questo evita di rimandare tutto il pacchetto TCP e consente di ritrasmettere solo i frame danneggiati.
- **Individuazione e correzione degli errori**: Il nodo mittente fornisce un meccanismo per rilevare e correggere eventuali errori nei frame.
- **Controllo di flusso**: Sincronizza la velocità di trasmissione tra nodo mittente e nodo ricevente.
- **Half-duplex e full-duplex**: In modalità half-duplex, solo un nodo può trasmettere alla volta. In modalità full-duplex, entrambi i nodi possono trasmettere e ricevere contemporaneamente.

### 1.1.2 Implementazioni

Il livello data link deve essere implementato in ogni nodo connesso alla rete. Le funzionalità Ethernet sono integrate nella scheda madre o in un chip Ethernet. Il livello collegamento è implementato su un chip detto *network adapter* o NIC.

## 1.2 Individuazione e correzione degli errori

L'individuazione degli errori prevede l'aggiunta di bit o byte aggiuntivi ai dati trasmessi per rilevare o correggere eventuali errori. Potrebbero verificarsi errori sia nei dati sia nei bit aggiuntivi, poiché il canale potrebbe non essere affidabile.

### 1.2.1 Controllo di parità

Un singolo bit di parità viene aggiunto a una sequenza di bit secondo regole condivise tra mittente e ricevente. Se il numero di bit corrotti è pari, il test passa con successo anche se è presente un errore.

### Esempio

Se il numero di 1 è dispari, aggiungo un 1; se è pari, aggiungo uno 0. Attraverso matrici di parità, calcolo i bit di parità sia per le righe sia per le colonne, confrontando se coincidono. Posso correggere un errore singolo, ma posso solo rilevare errori su due bit.

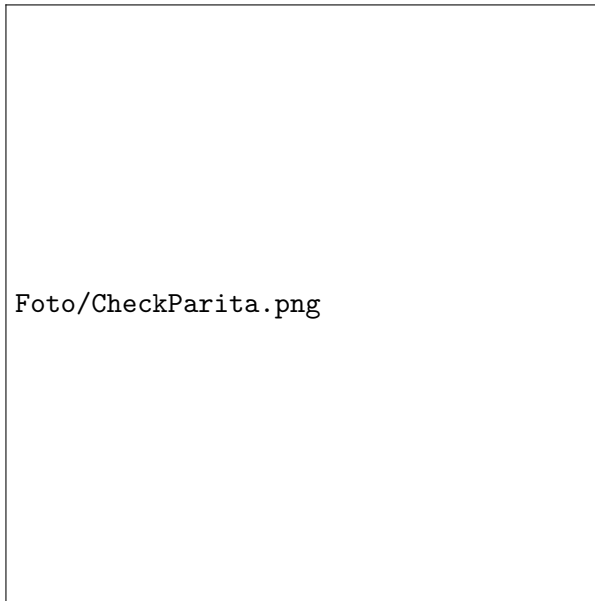


Figura 1: Esempio di controllo di parità

#### 1.2.2 CRC (Cyclic Redundancy Check)

Il CRC è uno dei più potenti metodi di rilevamento errori, ma non può correggerli. I codici CRC sono detti anche *codici polinomiali*, poiché la stringa da inviare può essere rappresentata come un polinomio con coefficienti binari.

- $D$ : numero di bit per i dati.
- $G$ : generatore di bit con caratteristiche specifiche, di lunghezza  $r + 1$  bit.
- $R$ : resto calcolato tramite divisione polinomiale modulo 2.

## Svolgimento

Per calcolare il CRC:

1. Aggiungo in coda una sequenza di CRC lunga  $r$  bit (un bit in meno rispetto al generatore).

$$\langle D, R \rangle = D \cdot 2^r \text{ XOR } R$$

2. Shift dei dati:  $D \cdot 2^r$  (shifto il dato di  $r$  bit).
3.  $\text{XOR } R$ , metto come ultimi bit significativi quelli di  $R$ .
4. Il mittente prende gli  $R$  bit dei CRC e prende  $\langle D, R \rangle$  e lo divide per  $G$  e fa il modulo 2.
5. Il destinatario verifica la correttezza dividendo  $\langle D, R \rangle$  per  $G$ : se il resto è 0, il test è passato.

$R$  deve essere tale per cui:

$$D \cdot 2^r \text{ XOR } R = nG$$

AB	A XOR B
00	0
01	1
10	1
11	0

Posso rilevare un numero di bit errati  $< G$ . Inoltre posso rilevare un numero di errori dispari.

## Esempio CRC

$D = 101110$      $G = 1001$      $r = 3$

```
10111000 :1001
1001 XOR 101011
  101
 000 XOR
 1010
1001 XOR
 110
 000 XOR
 1100
1001 XOR
 1010
 1001 XOR
   011 = R
```

Il mittente trasmette 101110**011**, e il destinatario verifica il resto.

**Generatori CRC standard** Esistono generatori  $G$  standard, utilizzati in base alla lunghezza del dato, ad esempio: 8, 12, 16 o 32 bit.

### 1.3 Protocolli ad accesso multiplo

Esistono due tipologie di link:

1. **Punto-Punto**: un nodo è collegato direttamente ad un altro attraverso un link (canale) dedicato;
  - Esempio: collegamento Ethernet tra un host e uno switch.

2. **Broadcast**: il link è condiviso, potenzialmente diversi nodi possono trasmettere e/o ricevere contemporaneamente;

- Esempio: prime versioni di Ethernet (tipologia a bus), comunicazione wireless.

I protocolli ad accesso multiplo stabiliscono un insieme di regole che ogni nodo deve seguire affinché la comunicazione avvenga con successo. In assenza di protocolli si verifica il problema delle **collisioni**, ossia due o più nodi che trasmettono contemporaneamente.

Consideriamo algoritmi di tipo distribuito che determinano come il canale viene condiviso; per esempio, questi algoritmi stabiliscono quando un determinato nodo deve trasmettere o cosa accade in caso di collisione. La comunicazione avverrà sia per trasmettere dati sia per regolare la trasmissione stessa, utilizzando lo stesso canale.

**Protocollo ideale** Consideriamo un canale ad accesso condiviso che è in grado di trasmettere ad un rate di  $R$  bps (bit per secondo).

Si desidera che:

1. quando un singolo nodo vuole trasmettere, può trasmettere ad una velocità di trasmissione pari ad  $R$ ;
2. quando ci sono  $M$  nodi che vogliono trasmettere contemporaneamente, ogni nodo può trasmettere ad una velocità media di  $\frac{R}{M}$ ;
3. completamente decentralizzato:
  - non voglio nodi specializzati che coordinino la trasmissione (single point of failure);
  - non voglio la sincronizzazione dei clock dei diversi nodi o degli slot.
4. semplice.

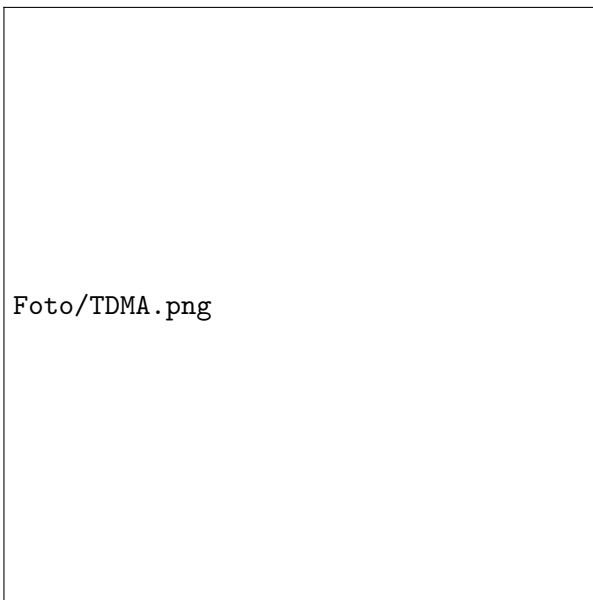
### 1.3.1 Protocolli a partizionamento del canale

Divido la capacità complessiva del canale in diverse parti che assegno ai diversi nodi.

**TDMA** Ogni nodo può accedere al canale solo in determinati intervalli di tempo. Ad ogni stazione viene assegnato un intervallo di tempo fisso in cui può trasmettere. Gli slot inutilizzati vengono persi, NON riutilizzati da altri nodi.

#### Esempio

Il time-frame viene suddiviso in un numero di slot pari al numero di nodi complessivi della rete.



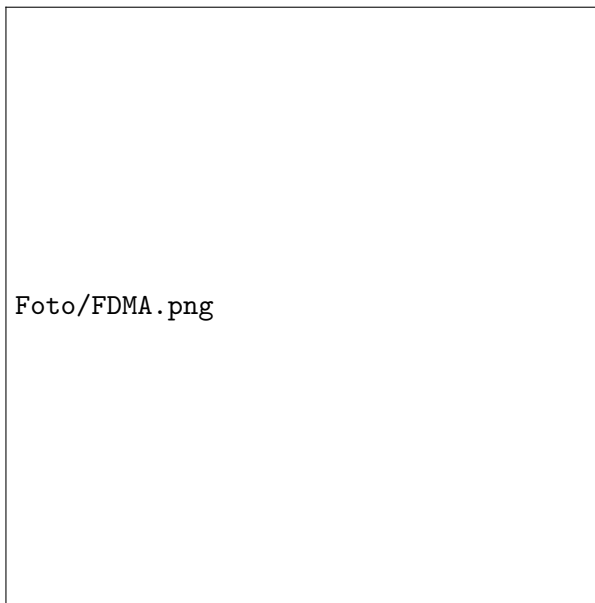
Proprietà ideali:

1. Soddisfatto;
2. Soddisfatto;
3. Non soddisfatto, è necessario mantenere gli stessi clock per sapere quando inizia uno slot;
4. Soddisfatto.

**FDMA** Viene partizionato il dominio delle frequenze. Ad ogni stazione viene assegnata una frequenza stabilita. Il tempo non utilizzato in frequenza viene sprecato. Posso non usare certe frequenze tra un intervallo ed un altro per evitare le sovrapposizioni di banda.

#### Esempio

Vengono partizionate le frequenze in base al numero di nodi presenti.



Proprietà ideali:

1. Non soddisfatto;
2. Soddisfatto;
3. Soddisfatto;
4. Soddisfatto;

#### 1.3.2 Protocolli ad accesso casuale

Il canale non è diviso e si permette l'occorrenza di collisioni. In caso di collisioni, devo risolverle. Quando un nodo ha un pacchetto da inviare, lo trasmette alla massima velocità. A priori non c'è coordinamento tra i nodi.

**Slotted ALOHA** Assumiamo che tutti i frame abbiano la stessa dimensione, il tempo è diviso in slot di uguale dimensione necessaria per trasmettere un intero frame. I nodi iniziano a trasmettere solo all'inizio dello slot. I nodi sono sincronizzati. Se 2 o più nodi trasmettono simultaneamente, tutti i nodi della rete rilevano la collisione.

Quando un nodo ottiene un nuovo frame da inviare, lo trasmette nel successivo slot. Dopo la trasmissione, il nodo si mette in ascolto; se rileva un picco di energia, identifica la collisione e ritrasmette nel successivo slot con probabilità  $p$ .



## Esempio

All'inizio tutti i nodi vogliono trasmettere e avviene una collisione. Successivamente, dopo uno slot vuoto, i nodi 1 e 2 ritrasmettono e vanno in collisione. Allo slot successivo solo 2 trasmette ed ha successo. E così via finché tutti non hanno trasmesso.



Proprietà ideali:

1. Soddisfatta;
2. Non soddisfatta;
3. Non soddisfatta, perché è richiesta sincronizzazione;
4. Soddisfatta.

Durante le collisioni vengono sprecati degli slot. Un nodo è in grado di accorgersi subito della collisione, ma deve aspettare un nuovo slot per trasmettere.

## Efficienza

Un indicatore dell'efficienza è la proporzione di slot che hanno una comunicazione con successo rispetto a tutti gli slot disponibili su un intervallo di tempo lungo.

Supponiamo che  $N$  nodi vogliano continuare a trasmettere dei frame ed ognuno trasmetta con probabilità  $p$ .

- La probabilità che un dato nodo abbia successo nella trasmissione è  $p(1 - p)^{N-1}$ ;
- La probabilità che un qualsiasi nodo abbia successo è  $Np(1 - p)^{N-1}$ ;
- Per la massima efficienza, trovo  $p^*$  che massimizza  $Np(1 - p)^{N-1}$ ;
- Per molti nodi, prendo il limite di  $Np^*(1 - p^*)^{N-1}$  quando  $N$  tende a infinito;
- Efficienza massima =  $\frac{1}{e} \approx 0.37$ .

Nella migliore delle ipotesi, utilizzo il canale il 37% del tempo.

**CSMA (Carrier Sense Multiple Access)** Un nodo ascolta il canale prima di trasmettere (*carrier sense*). Se il canale è libero, trasmette; altrimenti, aspetta.

### Esempio

Possono capitare delle collisioni anche ascoltando prima di trasmettere, poiché il segnale non si propaga immediatamente ovunque (*propagation delay*).

Foto/CSMA.png

**CSMA/CD** Protocollo utilizzato in Ethernet. CSMA con rilevamento della collisione (*collision detection*). Durante la trasmissione, se rilevo una collisione, interrompo la trasmissione per ridurre lo spreco del canale. È facile da implementare su canali via cavo, ma difficile in wireless.

### Esempio

Per velocizzare il rilevamento delle collisioni, appena qualcuno la rileva, invia un segnale agli altri nodi, affinché tutti rilevino la collisione.

Foto/CSMACD.png

### Algoritmo di Ethernet

1. Ethernet riceve il datagramma dal livello rete sovrastante e crea il frame;
2. La scheda di rete ascolta il canale:
  - se è libero: inizia la trasmissione del frame;
  - se è occupato: aspetta finché il canale non si libera e poi trasmette.
3. Se trasmetto l'intero frame senza collisioni, ho finito;
4. Se rilevo una collisione durante l'invio, interrompo la trasmissione e mando un segnale di *abort* (*jamming signal*) agli altri nodi;
5. Dopo l'abort, ritrasmetto utilizzando un algoritmo di **binary (exponential) backoff**:
  - dopo l' $m$ -esima collisione, scelgo un numero casuale  $K$  nell'intervallo  $\{0, 1, 2, \dots, 2^{m-1}\}$ . Ethernet aspetta  $K \cdot 512$  bit times (tempo necessario per trasmettere un bit) e poi ritorna al punto 2;
  - Più collisioni ci sono, maggiore sarà l'attesa per ritrasmettere.

### 1.3.3 Protocolli a turni

Ogni nodo può trasmettere in maniera dedicata, ma solo in determinati turni.

**Polling** Un controllore centralizzato richiede ciclicamente ad ogni nodo se ha bisogno di trasmettere. Se un nodo deve trasmettere, gli viene dedicato un intervallo di tempo; altrimenti, il controllore passa al nodo successivo.

**Contro:**

- **Overhead del polling:** il controllore invia richieste di controllo (pochi byte);
- **Latenza:** i nodi devono attendere che il controllore completi il ciclo;
- **Single point of failure:** se il controllore si guasta, l'intero sistema si blocca.

Questo protocollo è utilizzato in **Bluetooth**.

**Token-passing** In questo protocollo non c'è un master. Al suo posto, si utilizza un *token*. I nodi sono disposti in un anello logico e il token passa ciclicamente tra i nodi. Il nodo che possiede il token può trasmettere per un intervallo di tempo fisso, costante e limitato.

**Contro:**

- **Overhead del token:** il token deve essere generato e trasferito;
- **Latenza:** i nodi devono attendere il proprio turno per ricevere il token;
- **Single point of failure:** se un nodo si guasta mentre ha il token, il sistema si blocca.

## 1.4 Switched Local Area Networks

Gli switch utilizzano indirizzi propri a livello di collegamento per inoltrare i frame.

### 1.4.1 Indirizzo MAC

**Indirizzi IP** Gli indirizzi IP, definiti a livello di rete, sono associati alle interfacce dei dispositivi e sono lunghi **32 bit**. Vengono utilizzati per il forwarding dei pacchetti.

**MAC** Le interfacce di rete degli host e dei router possiedono indirizzi a livello di collegamento, chiamati **indirizzi MAC**, mentre gli switch non li utilizzano.

Gli indirizzi MAC:

- Sono lunghi **6 byte (48 bit)**;
- Vengono espressi in notazione esadecimale;
- Sono unici per ogni interfaccia all'interno di una LAN, poiché controllati dalla IEEE.

Tipicamente, i **primi 24 bit** più significativi sono associati a uno specifico produttore di dispositivi.

**Funzionamento** Quando un'interfaccia vuole inviare un frame:

- Inserisce l'indirizzo MAC di destinazione nel frame;
- Inoltra il frame sulla LAN.

Se il mittente vuole inviare un frame a tutte le interfacce presenti sulla LAN, utilizza un indirizzo speciale, chiamato **indirizzo broadcast**, nel frame.

## 1.5 Protocollo ARP (Address Resolution Protocol)

Il protocollo ARP, tipicamente implementato su tutti gli host, serve per fare il mapping tra l'indirizzo IP e l'indirizzo MAC corrispondente all'interno di una sotto-rete.

Ciascun nodo possiede una **tabella ARP** che contiene:

- La corrispondenza tra indirizzo IP e indirizzo MAC;
- **TTL (Time To Live)**: intervallo di tempo dopo il quale il mapping non è più valido.

### Esempio protocollo ARP nella stessa LAN

Supponiamo che *A* voglia inviare un datagramma a *B* e che l'indirizzo MAC di *B* non sia presente nella tabella ARP di *A*.

*A* esegue un **broadcast ARP** sull'intera rete fisica utilizzando l'indirizzo MAC *FF-FF-FF-FF-FF-FF*, specificando l'indirizzo IP di *B* e il proprio indirizzo MAC sorgente. Tutti i nodi ricevono questa richiesta e la processano; solo *B* risponderà direttamente in modalità unicast, specificando il proprio indirizzo MAC associato all'indirizzo IP.

*A* salva l'informazione nella sua tabella ARP. Questa entry può essere:

- **Cancellata** quando il suo TTL scade;
- **Rinnovata** in caso di una nuova richiesta con la stessa associazione IP-MAC.


## Esempio protocollo ARP: invio in sotto-reti diverse

Supponiamo che  $A$  debba inviare un datagramma a  $B$ .  
Assumiamo che  $A$  conosca:

- L'indirizzo IP di  $B$ ;
- L'indirizzo IP del primo router (attraverso il **DHCP**);
- L'indirizzo MAC dell'interfaccia del router.

1.  $A$  crea il datagramma IP con:

- IP sorgente: indirizzo IP di  $A$ ;
- IP destinazione: indirizzo IP di  $B$ .



Foto/ARPDatagram.png

2.  $A$  incapsula il datagramma in un frame con:

- MAC sorgente: indirizzo MAC di  $A$ ;
- MAC destinazione: indirizzo MAC dell'interfaccia del router.



Foto/ARPMac.png

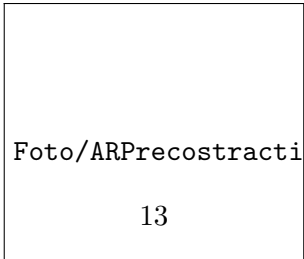
3. Il frame viene inviato da  $A$  al router.

4. Il router processa il pacchetto:

- Rimuove l'intestazione del frame e passa il datagramma al livello IP;
- Determina la prossima interfaccia di uscita.

5. Il router crea un nuovo frame:

- Gli indirizzi IP rimangono gli stessi ( $A$  e  $B$ );
- L'indirizzo MAC sorgente diventa l'indirizzo dell'interfaccia del router;
- L'indirizzo MAC destinazione diventa quello di  $B$ .



Foto/ARPreconstruction.png

## 1.6 Ethernet

La LAN Ethernet originariamente utilizzava un bus coassiale per connettere i nodi, e i frame erano trasmessi in broadcast. Successivamente, il bus è stato rimpiazzato con uno switch, che, a differenza dei router, opera esclusivamente a livello collegamento.

### Struttura di un frame Ethernet

Preambolo	Indirizzo destinazione	Indirizzo sorgente	Tipo	Campo dati	CRC
-----------	------------------------	--------------------	------	------------	-----

- **Preambolo:** utilizzato per sincronizzare mittente e destinatario e per regolare la velocità di trasmissione dei bit. Consiste in 7 byte con il pattern *10101010* seguiti da 1 byte con il pattern *10101011*. Questo pattern segnala che le informazioni iniziano subito dopo il preambolo;
- **Indirizzo destinazione:** indirizzo MAC del destinatario, lungo 6 byte;
- **Indirizzo sorgente:** indirizzo MAC del mittente, lungo 6 byte;
- **Tipo:** indica il protocollo di livello superiore del payload (tipicamente IP), rappresentato da 4 bit;
- **Campo dati:** contiene il datagramma IP, con una dimensione massima di 1500 byte e minima di 46 byte;
- **CRC (Cyclic Redundancy Check):** utilizzato per verificare la correttezza del frame.

In Ethernet:

- Non c'è un handshake tra mittente e destinatario, rendendolo un protocollo **connectionless**;
- Gli errori vengono rilevati ma il frame corrotto è scartato, quindi è un protocollo **unreliable**.

Il protocollo MAC utilizzato in Ethernet è **CSMA/CD (Carrier Sense Multiple Access with Collision Detection)** con **binary backoff**.

**Ethernet 802.3** È uno standard che definisce sia gli aspetti di livello **data-link** che di livello **fisico**.

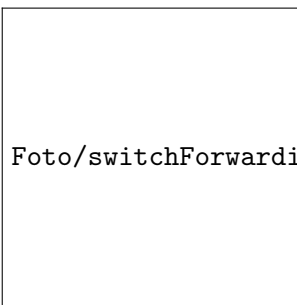
## 1.7 Switch Ethernet

Gli switch Ethernet sono dispositivi a livello collegamento con un ruolo attivo: memorizzano e inoltrano i frame Ethernet, esaminano gli indirizzi MAC, e decidono in modo selettivo se e a quale interfaccia (porta) inoltrare il frame. Questi dispositivi operano su segmenti Ethernet dove si utilizza il protocollo CSMA/CD per l'accesso.

Gli switch sono dispositivi **trasparenti** agli host e supportano soluzioni **plug and play** (nessuna necessità di configurazione) e **self learning**. Essi consentono la trasmissione simultanea di più frame, offrendo connessioni dedicate (tipicamente full-duplex) per ogni host. Inoltre, bufferizzano i pacchetti, eliminando così le collisioni, e utilizzano il protocollo Ethernet su ogni connessione.

## Switch forwarding table

**Switch forwarding table** Come fanno gli switch a sapere che  $A'$  è raggiungibile attraverso l'interfaccia 5?



Foto/switchForwardingTable.png

Ogni switch mantiene una **tabella di inoltramento** che contiene:

- **Indirizzo MAC**;
- **Interfaccia** per raggiungere l'host;
- **Time stamp (TTL)** = intervallo di tempo dopo il quale l'entry scade.

Le entry di questa tabella sono riempite in modo diverso rispetto alle tabelle di routing. Lo switch impara dinamicamente la corrispondenza tra indirizzo MAC e porta a cui inoltrare il frame.

**Self learning** Ipotizziamo che  $A$  voglia inoltrare un frame a  $A'$ .  $A$  invia il frame sul cavo fino a raggiungere lo switch. Lo switch memorizza che l'host  $A$  è raggiungibile dalla porta 1 e aggiunge questa informazione alla sua tabella.

### Frame filtering/forwarding

- **Filtering**: determina se un frame deve essere inoltrato o scartato.
- **Forwarding**: determina le interfacce a cui il frame deve essere inoltrato.

Quando il frame raggiunge lo switch:

1. Memorizza l'indirizzo MAC dell'host mittente;
2. Indicizza la tabella dello switch utilizzando l'indirizzo MAC di destinazione;
3. Se viene trovata una entry per il destinatario:
  - Se la destinazione è sullo stesso segmento da cui il frame è arrivato, lo scarta;
  - Altrimenti lo inoltra all'interfaccia corretta.
4. Se non viene trovata una entry:
  - Esegue il **flooding**, inoltrando il frame su tutte le interfacce eccetto quella del mittente.

### 1.7.1 Interconnessione tra switch

La tipologia a stella non è sufficiente per LAN di dimensioni importanti. Gli switch possono quindi essere connessi tra loro in modo gerarchico. Durante il flooding, l'informazione si propaga fino a raggiungere tutti gli host.



## 1.8 Switch vs Router

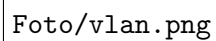
- Entrambi implementano un meccanismo di **store-and-forward**:
  - *Router*: dispositivi di **livello rete** (esaminano le informazioni contenute nelle intestazioni a livello di rete, come gli indirizzi IP);
  - *Switch*: dispositivi di **livello collegamento** (esaminano solo le intestazioni di livello 2, come gli indirizzi MAC).
- Entrambi utilizzano **tabelle di inoltro**:
  - *Router*: le tabelle sono calcolate attraverso **algoritmi di routing**, basandosi sugli **indirizzi IP**.
  - *Switch*: le tabelle sono costruite attraverso il **self-learning**, utilizzando tecniche come il flooding per apprendere dinamicamente gli **indirizzi MAC**.

## 1.9 VLAN (Virtual LAN)

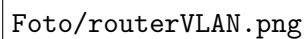
Le reti LAN tradizionali non sono facilmente scalabili o portabili, ma le VLAN lo sono. Uno switch che supporta VLAN permette di definire LAN virtuali su una singola LAN fisica. Gli host su una VLAN possono comunicare tra loro come se fossero solo loro connessi a quello switch.

**VLAN basata sulle porte** Lo switch può essere configurato per fare in modo che le porte vengano etichettate e dedicate a una particolare VLAN (solitamente caratterizzate da un colore). Le porte in ogni VLAN formano un **dominio di broadcast**. Per far comunicare dispositivi appartenenti a VLAN diverse è necessario utilizzare un router esterno.





Foto/vlan.png



Foto/routerVLAN.png

### Vantaggi

- **Configurazione dinamica:** è possibile cambiare la VLAN di un dispositivo senza spostarlo fisicamente.
- Possibilità di definire la VLAN basandosi sugli **indirizzi MAC** degli endpoint anziché sulle porte fisiche dello switch.

### 1.10 VLAN spanning (trunk port)

In presenza di più switch distinti, è possibile fare in modo che gli host di uno switch secondario possano appartenere alle VLAN dello switch principale.

Foto/spanning.png

Ciò è possibile utilizzando una particolare **porta trunk** dello switch che permette di interconnettere LAN differenti. Ad esempio, se si manda un messaggio in broadcast sulla VLAN 1, esso viene inoltrato anche sulla porta trunk (es. porta 16), connessa fisicamente all'altro switch, e arriva alla porta corrispondente (es. porta 1) del secondo switch.

Sulla porta trunk passano i frame di una qualunque VLAN. Per identificare a quale VLAN appartiene un frame, si utilizza il **protocollo 802.1q**, che aggiunge un campo aggiuntivo nell'header del protocollo Ethernet. Questo campo è gestito solo dagli switch. Quando uno switch riceve il frame, riconosce la VLAN di appartenenza, lo inoltra al destinatario corretto e rimuove le informazioni aggiuntive per tornare al formato standard del protocollo Ethernet.

Preambolo	Destinazione	Sorgente	Tipo di protocollo	Informazioni di controllo	Tipo	Dati	CRC
-----------	--------------	----------	--------------------	---------------------------	------	------	-----

- **Tipo di protocollo:** 2 byte (valore 81-00).
- **Informazioni di controllo:**
  - 12 bit per identificare la VLAN;
  - 3 bit per definire la priorità.

## 1.11 Networking nei data center

### 1.11.1 Architetture dei data center

I data center non sono solamente connessi a Internet, ma dispongono di propri network interni che connettono gli host tra loro. Gli host nei data center sono chiamati **blades**, e sono impilati in rack. In cima ad ogni rack c'è uno switch, detto **Top of Rack (TOR)**, che connette gli host nel rack tra di loro e con gli altri switch nel data center. Ogni host nel rack è dotato di un'interfaccia di rete che si connette al proprio TOR, mentre ogni TOR dispone di porte che possono essere collegate ad altri switch.

I data center supportano due tipi di traffico:

1. **Traffico tra client esterni e host interni;**
2. **Traffico tra host interni.**

Per gestire il primo tipo di traffico, i data center utilizzano dei **border router**, che li connettono direttamente a Internet.

### 1.11.2 Load balancing

Le richieste ricevute da un data center sono inizialmente dirette a un load balancer, che si occupa di distribuire le richieste agli host. I grandi data center dispongono di numerosi load balancer, ciascuno dedicato a specifiche applicazioni cloud. Un tale load balancer è spesso chiamato “*layer-4 switch*”, in quanto prende le sue decisioni in base al numero di porta e all’indirizzo IP del pacchetto.

Al ricevimento di una richiesta, il load balancer la inoltra a uno degli host che gestisce l’applicazione. Il load balancer opera anche come NAT, traducendo l’IP pubblico della richiesta nell’IP interno e viceversa.

## 1.12 Il processamento di una richiesta

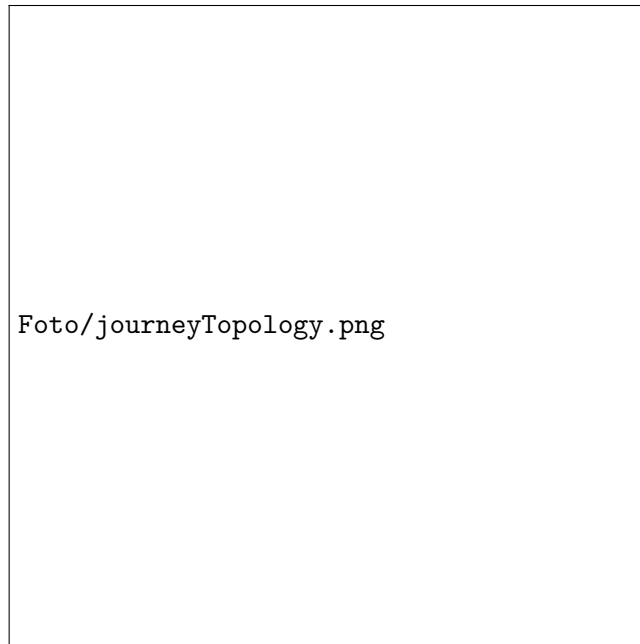


Figura 2: Topologia del percorso della richiesta.

### 1.12.1 Connessione a Internet

Il client si connette e, come prima cosa, deve ottenere:

- Il proprio indirizzo IP;
- L’indirizzo IP del first-hop router;
- L’indirizzo del server DNS.

Per ottenere queste informazioni utilizza il protocollo **DHCP**.

Il client invia una richiesta DHCP, che viene incapsulata in UDP, successivamente in IP e infine in 802.3 Ethernet. Questa richiesta viene inoltrata in broadcast.

Il server DHCP risponde con un **ACK**, specificando:

- L’indirizzo IP del client;
- Le altre informazioni richieste (indirizzo del router e del server DNS).

La risposta viene incapsulata e inoltrata in unicast al nodo.

Ora il client ha un indirizzo IP, sa il nome e l’indirizzo del server DNS, e conosce l’indirizzo IP del first-hop router.

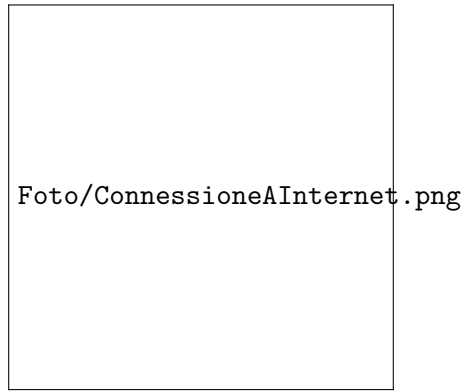


Figura 3: Processo di connessione a Internet tramite DHCP.

### 1.12.2 Richiesta HTTP

Prima di inviare una richiesta HTTP, il client necessita dell'indirizzo IP del dominio *www.google.com*. Il client crea una query DNS, che viene incapsulata in UDP, successivamente in IP e infine in Ethernet. Per inviare il frame al router, il client necessita dell'indirizzo MAC del router, che ottiene utilizzando il protocollo **ARP**.

Il protocollo ARP invia una richiesta in broadcast, ricevuta dal router, il quale risponde con l'indirizzo MAC dell'interfaccia. Ora il client è a conoscenza dell'indirizzo MAC del router e può inviare la richiesta DNS.

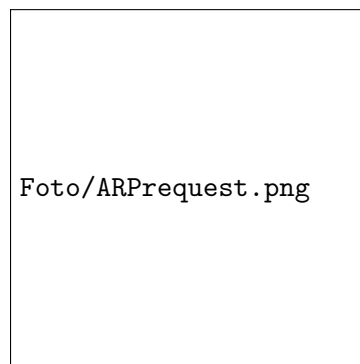


Figura 4: Richiesta ARP per ottenere l'indirizzo MAC del router.

Il datagramma IP contenente la query DNS viene inoltrato dal client al first-hop router attraverso lo switch LAN. Successivamente, il datagramma IP attraversa il *campus network* e il *comcast network* fino al server DNS.

Il server DNS risponde con l'indirizzo IP di *www.google.com*.



Figura 5: Comunicazione con il server DNS.

Per la richiesta HTTP, il client apre una connessione TCP eseguendo il **3-way handshake**.

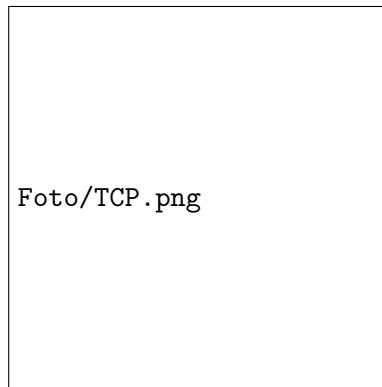


Figura 6: Processo di 3-way handshake per aprire una connessione TCP.

Una volta stabilita la connessione, la richiesta HTTP viene inoltrata attraverso il socket TCP. Il datagramma IP contenente la richiesta HTTP attraversa la rete fino a raggiungere il server *www.google.com*. Il web server risponde con una **HTTP reply**, contenente la pagina web richiesta. Il datagramma IP contenente la risposta HTTP viene inoltrato fino a raggiungere il client.



Figura 7: Risposta finale del server HTTP al client.

## 2 Livello fisico

### 2.1 Segnali a larghezza di banda limitata

Qualsiasi forma di segnale, incluse quelle discrete (digitali, come le sequenze di 1 e 0), può essere scomposta tramite la serie di Fourier:

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

Questa scomposizione rappresenta il segnale come una somma di armoniche, con ampiezze ( $a_n$  e  $b_n$ ) che ne determinano la potenza. L'inclusione di più armoniche consente di approssimare il segnale originale con maggiore precisione.



Figura 8: Esempio di approssimazione di un segnale mediante armoniche.

**Larghezza di banda** La larghezza di banda (*bandwidth*) è una caratteristica del mezzo trasmissivo, definita come l'intervallo di frequenze che il mezzo può trasmettere senza degradare eccessivamente il segnale.

- I segnali che occupano un intervallo di frequenze da 0 a una frequenza massima sono detti **baseband**.
- I segnali traslati per occupare un intervallo più alto di frequenze sono detti **passband**.

#### 2.1.1 Data rate massimo per un canale

La velocità massima di trasmissione dei dati è limitata dalla larghezza di banda, indipendentemente dalla tecnica di codifica utilizzata.

**Canale ideale (senza rumore)** L'equazione di Nyquist fornisce il data rate massimo per un canale con banda finita e privo di rumore:

$$\text{Maximum data rate} = 2B \log_2 V \left[ \frac{\text{bit}}{\text{sec}} \right]$$

dove:

- $B$  è la larghezza di banda (in Hz),
- $V$  è il numero di livelli discreti differenti del segnale.

Se un segnale attraversa un filtro passa-basso (*low-pass filter*) con banda  $B$ , può essere ricostruito con  $2B$  campioni al secondo.

**Canale con rumore** In presenza di rumore, il segnale degrada e il data rate massimo è determinato dall'equazione di Shannon:

$$\text{Maximum data rate} = B \log_2 \left( 1 + \frac{S}{N} \right)$$

dove:

- $\frac{S}{N}$  è il rapporto segnale-rumore (*Signal-to-Noise Ratio*, SNR), spesso espresso in decibel (dB).

#### Definizioni

##### Parametri principali:

- $V$ : Numero di livelli discreti differenti del segnale.
- $\frac{S}{N}$ : Rapporto segnale-rumore (SNR), calcolato in decibel come:

$$\text{SNR(dB)} = 10 \cdot \log_{10} \left( \frac{S}{N} \right)$$

## 2.2 Modulazione digitale

La modulazione digitale è il processo di conversione dei bit in segnali trasmissibili.

### 2.2.1 Trasmissione baseband

**NRZ (Non-Return-to-Zero)** La modulazione più semplice prevede l'uso di una tensione positiva per rappresentare 1 e una negativa per rappresentare 0. Il destinatario decodifica i segnali mappandoli ai simboli più coerenti. Con NRZ, il segnale si alterna tra due livelli, richiedendo una banda minima di  $\frac{B}{2}$  per un bit rate  $B$  bit/s. Utilizzando quattro livelli di tensione, è possibile inviare 2 bit per simbolo.

**Manchester** Per facilitare la sincronizzazione in presenza di lunghe sequenze di 0 o 1 consecutivi, il segnale dati è combinato con il segnale di clock mediante un'operazione XOR. Una transizione alta-bassa rappresenta 0, mentre una transizione bassa-alta rappresenta 1. Questo metodo raddoppia la banda necessaria rispetto a NRZ.

**NRZI (Non-Return-to-Zero Inverted)** Codifica 1 come una transizione e 0 come l'assenza di transizione.

**Mapping a blocchi** Per mantenere un numero sufficiente di transizioni e garantire la sincronizzazione, i dati sono codificati in sequenze più lunghe. Ad esempio, ogni 4 bit di dati possono essere mappati in una sequenza di 5 bit, con un overhead del 25%.

**Scrambling** I dati sono combinati (XOR) con una sequenza pseudocasuale per migliorare le proprietà del segnale.

**Segnali bilanciati** I segnali bilanciati, come il **bipolar encoding**, alternano tensioni positive e negative per rappresentare 1, mentre 0 è rappresentato da 0 volt. Questo elimina la componente DC.



Figura 9: Esempi di trasmissione baseband.

### 2.2.2 Trasmissione passband

La trasmissione passband codifica i dati digitali utilizzando segnali analogici, modificando parametri come ampiezza, frequenza o fase.

#### Parametri delle onde

Un'onda sinusoidale è descritta da:

$$A \sin(2\pi ft + \phi)$$

- $A$ : ampiezza (intensità del segnale).
- $f$ : frequenza (numero di cicli al secondo, inverso del periodo).
- $\phi$ : fase (punto di inizio del segnale).

L'idea è quella di codificare i bit in base alla modulazione dei parametri soprastanti.

Il segnale baseband è aumentato per occupare una banda da  $S$  a  $S + B$  Hz senza cambiare la quantità di dati trasmessi.

**ASK (Amplitude Shift Keying)** Modifica l'ampiezza dell'onda. Lo 0 corrisponde ad ampiezza  $A = 0$ , mentre l'1 ha un'ampiezza concordata.





Figura 10: Esempio di ASK.

**FSK (Frequency Shift Keying)** Utilizza frequenze differenti: una frequenza  $f$  per lo 0 e una  $2f$  per l'1.



Figura 11: Esempio di FSK.

**BPSK (Binary Phase Shift Keying)** Codifica i bit modificando la fase della forma d'onda. Lo 0 e l'1 hanno fasi opposte.



Figura 12: Esempio di BPSK.

**Soluzioni ibride** Combinano più parametri per aumentare l'efficienza.

**QPSK (Quadrature Phase Shift Keying)** Utilizza quattro fasi distinte ( $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ ,  $270^\circ$ ) per rappresentare due bit per simbolo (00, 01, 10, 11).

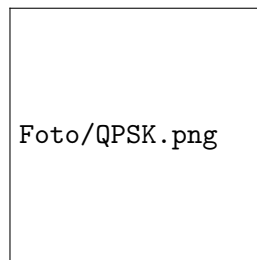


Figura 13: Esempio di QPSK.

**QAM (Quadrature Amplitude Modulation)** Combina modulazione di fase e ampiezza:

- **QAM-16:** Utilizza 16 punti distinti per rappresentare 4 bit per simbolo,  $2^4 = 16$ .
- **QAM-64:** Utilizza 64 punti per rappresentare 6 bit per simbolo,  $2^6 = 64$ .

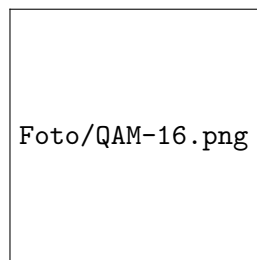


Figura 14: Costellazione QAM-16.

**FDM (Frequency Division Multiplexing)** Divide lo spettro in bande di frequenza, assegnando ciascuna banda a una sorgente distinta. Le bande sono separate da *guard bands* per ridurre interferenze.

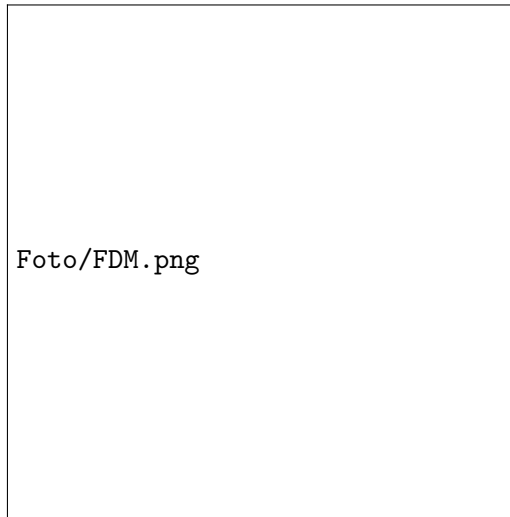


Figura 15: Schema di FDM.

**OFDM (Orthogonal Frequency Division Multiplexing)** Divide la banda in subcarrier ortogonali, che trasmettono dati indipendentemente senza interferenza reciproca.

## 3 Reti wireless e mobili

### 3.1 Elementi di una rete wireless

In una rete wireless si possono identificare i seguenti elementi fondamentali:

- **Host wireless:** dispositivi come laptop, smartphone e dispositivi IoT.
  - *Stazionari:* dispositivi che si connettono a reti diverse in base alla posizione, ad esempio un laptop che utilizza il Wi-Fi in università e poi si connette al Wi-Fi di casa.
  - *Mobili:* dispositivi in movimento, come uno smartphone connesso alla rete mobile.
- **Base station:** dispositivi generalmente connessi alla rete cablata (es. access point o antenne degli operatori mobili). Questi fungono da intermediari tra i dispositivi mobili e la rete cablata.
- **Collegamenti wireless:** connessioni che permettono a un host di comunicare con una base station o direttamente con un altro host wireless.

#### 3.1.1 Modalità di funzionamento

Le reti wireless possono operare in due modalità principali:

**Modalità con infrastruttura** In questa modalità, una base station gestisce la connessione dei dispositivi alla rete. I dispositivi si connettono a una base station in modo wireless, e la comunicazione tra dispositivi può passare attraverso l'infrastruttura.

Un caso particolare è l'**handoff**, che si verifica quando un dispositivo si sposta da una base station a un'altra. In tal caso, l'infrastruttura deve gestire il trasferimento, inoltrando le richieste e le risposte tra la vecchia e la nuova base station.

**Modalità Ad Hoc** In modalità Ad Hoc, non esiste una base station. I dispositivi comunicano direttamente tra loro. Se un host non può comunicare direttamente con un altro, i dati vengono inoltrati tramite altri nodi intermedi (approccio "passa parola"). La copertura della rete è dinamica e si adatta alla posizione dei nodi.

Un esempio comune è l'hotspot di un cellulare, dove il nodo connesso alla rete cablata funge da tramite per altri dispositivi.

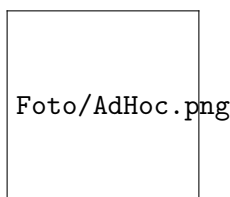


Figura 16: Esempio di rete Ad Hoc.

#### 3.1.2 Tipi di connessioni e infrastruttura

Le reti wireless possono essere classificate in base alla presenza o meno di infrastruttura e al tipo di connessione:

	Single hop	Multiple hop
Con infrastruttura	Dispositivi connessi a una base station tramite una <b>connessione diretta</b> .	Se non direttamente coperto, il dispositivo utilizza altri nodi come tramite per raggiungere la base station. Questo approccio è tipico di reti <b>meshed</b> .
Senza infrastruttura	Non è presente una base station, e la connessione non è necessariamente legata a Internet (es. Bluetooth).	Nessuna base station e nessuna connessione a Internet. I nodi possono inoltrare dati ad altri per raggiungere un dispositivo remoto. Tipico di reti MANET, VANET.

Tabella 1: Classificazione delle reti wireless in base alla presenza di infrastruttura e al tipo di connessione.

### 3.2 Caratteristiche dei collegamenti wireless

I collegamenti wireless presentano alcune differenze fondamentali rispetto ai collegamenti cablati:

- **Decremento della potenza del segnale:** il segnale radio si attenua progressivamente mentre si propaga attraverso il mezzo (path loss).
- **Interferenza dovuta ad altre sorgenti:** la trasmissione può essere disturbata da segnali provenienti da altre sorgenti che utilizzano la stessa frequenza o bande sovrapponibili. Anche dispositivi non comunicativi, come motori, possono generare interferenze.
- **Multipath propagation:** il segnale radio può riflettersi su ostacoli, percorrendo vie diverse rispetto a quella in linea d'aria. Questo può introdurre ritardi nel tempo di propagazione e degradare la qualità del segnale.

**SNR (Signal-to-Noise Ratio)** L'SNR è una misura del rapporto tra la potenza del segnale ricevuto e il rumore di fondo, espressa in decibel (dB). Maggiore è l'SNR, più facile è estrarre il segnale dal rumore.

- Un SNR elevato permette di ridurre il **BER** (Bit Error Rate), migliorando la qualità della comunicazione.
- Per un dato SNR, è possibile scegliere un livello di codifica del segnale che soddisfi il BER richiesto, garantendo la massima velocità di trasmissione compatibile con le condizioni del canale.
- La codifica può essere regolata per ridurre la velocità di trasmissione, mantenendo il BER sotto una soglia accettabile.

**Problema del terminale nascosto** Il problema del terminale nascosto si verifica quando più nodi trasmettono dati a un nodo ricevente, causando collisioni che impediscono al destinatario di decifrare i messaggi.

- In un contesto wireless, i nodi trasmettenti potrebbero non rilevare l'attività degli altri a causa di ostacoli o distanza eccessiva, portando a trasmissioni simultanee non rilevate.
- Questo problema rende inutilizzabile il protocollo 802.3, che si basa sulla rilevazione delle collisioni (*collision detection*).

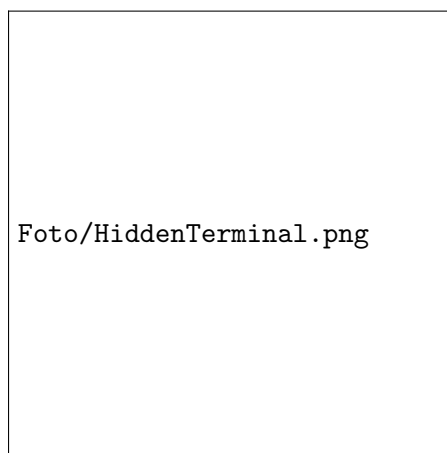


Figura 17: Esempio del problema del terminale nascosto.

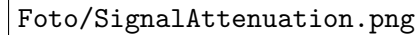


Figura 18: Problema di attenuazione del segnale, distanza eccessiva.

### 3.2.1 CDMA (Code Division Multiple Access)

Il CDMA è una tecnica di accesso multiplo in cui ogni nodo è associato a un codice univoco (chipping sequence). Questo consente a tutti i nodi di trasmettere simultaneamente sulla stessa frequenza, ma con codifiche distinte che il ricevente può decifrare.

**Codifica** La codifica viene effettuata moltiplicando i dati originali del nodo per la sua chipping sequence.

**Decodifica** Il ricevente decodifica il messaggio moltiplicando i dati ricevuti per la chipping sequence corrispondente al nodo desiderato. La somma risultante viene divisa per la lunghezza della chipping sequence ( $M$ ), ottenendo così il dato originale:

$$D_i = \frac{\sum_{m=1}^M Z_{i,m} \cdot c_m}{M}$$

#### Esempio con 1 mittente



Figura 19: Esempio di CDMA con 1 mittente

Un mittente trasmette un bit "1". La trasmissione consiste in una sequenza di bit, moltiplicata per la chipping sequence. Il ricevente decodifica il messaggio applicando lo stesso codice e calcolando il valore iniziale.

**Esempio con più mittenti** Con più mittenti, ciascuno trasmette utilizzando una chipping sequence differente. Il ricevente somma i segnali ricevuti e utilizza la chipping sequence del mittente desiderato per isolare il messaggio.

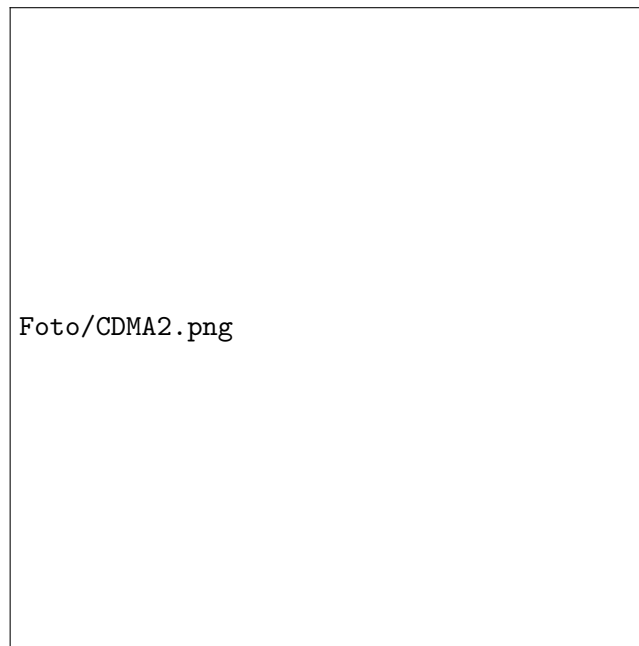


Figura 20: Esempio di CDMA con più mittenti

### Chipping sequence

Le chipping sequence devono essere ortogonali:

- Il prodotto scalare di due chipping sequence diverse deve essere 0:

$$S \cdot T = \frac{1}{M} \sum_{i=1}^M S_i T_i = 0 \quad \forall S, T, S \neq T.$$

- Il prodotto scalare di una chipping sequence con se stessa è sempre 1:

$$S \cdot S = 1.$$

### Esempio

Tre nodi  $A$ ,  $B$  e  $C$  trasmettono rispettivamente i bit 1, 0, 1. Alla ricezione, il segnale è:

$$S = A + \overline{B} + C.$$

Decodificando  $C$ , otteniamo:

$$S \cdot C = (A + \overline{B} + C) \cdot C = A \cdot C + \overline{B} \cdot C + C \cdot C = 0 + 0 + 1 = 1.$$

## 3.3 Reti WLAN 802.11

Lo standard delle trasmissioni wireless è 802.11, anche detto WiFi. Tutti i protocolli 802.11 utilizzano lo stesso protocollo di accesso al mezzo (a livello data link), CSMA/CA, e supportano modalità con infrastruttura o ad hoc.

### 3.3.1 Architettura

Gli host comunicano attraverso una base station, chiamata **access point (AP)** di livello data link. Il blocco fondamentale dell'architettura 802.11 è il **BSS (Basic Service Set)**, che include tutti i nodi

che si trovano all'interno della copertura dell'AP.

### 3.3.2 Canali e associazioni

Quando un nodo vuole connettersi alla rete, deve **associarsi** a uno specifico AP. Lo spettro è diviso in intervalli di frequenza, chiamati **canali**. L'AP sceglie un intervallo di frequenza su cui trasmettere, e il nodo, per comunicare, deve usare lo stesso intervallo. Interferenze possono verificarsi se più AP utilizzano lo stesso canale.

Gli host si associano all'AP tramite **beacon frame**, inviati periodicamente dall'AP. I beacon frame includono l'SSID (Service Set Identifier) e l'indirizzo MAC dell'AP. Dopo aver scelto un AP:

- L'host può autenticarsi (se richiesto) tramite indirizzo MAC o credenziali;
- L'AP comunica con un server di autenticazione per verificare l'accesso;
- Infine, l'host utilizza il DHCP per ottenere un indirizzo IP nella subnet dell'AP.

**Scanning passivo** Il nodo ascolta i beacon frame inviati dagli AP.

**Scanning attivo** Il nodo trasmette una richiesta in broadcast per rilevare gli AP disponibili. Gli AP rispondono, e il nodo sceglie con chi associarsi.

### 3.3.3 Accesso multiplo al canale

Lo standard utilizza **CSMA/CA (Collision Avoidance)** per evitare collisioni.

#### Protocollo MAC CSMA/CA

##### Mittente

1. Ascolta il canale, che deve essere in idle. Se è libero per un intervallo chiamato **DIFS**, trasmette il frame.
2. Se il canale è occupato, esegue il **backoff**: attende un tempo casuale, proporzionale al numero di collisioni, prima di ritentare. Se rileva trasmissioni durante il backoff, interrompe il timer e lo riprende quando il canale torna libero.

Foto/CSMA\_CA.png

##### Ricevente

- Se il frame è ricevuto correttamente, il nodo risponde con un **ACK** dopo un intervallo **SIFS**, per evitare il problema del terminale nascosto.

**Spaziatura interframe** I tempi **SIFS** e **DIFS** garantiscono priorità ai frame già in trasmissione.





### 3.3.4 Soluzione alternativa (RTS-CTS)

Un'altra modalità di accesso al canale prevede la **prenotazione** del canale:

- Il mittente trasmette un piccolo pacchetto **RTS (Request To Send)** all'AP utilizzando CSMA;
  - Sebbene gli RTS possano collidere, essendo corti riducono l'impatto.
- L'AP risponde in broadcast con un **CTS (Clear To Send)**;
- Il mittente trasmette il frame, mentre gli altri nodi posticipano le loro trasmissioni.

Questa modalità viene poco usata, perché lenta e non ci sono troppi frame così grandi per giustificarla. Tipicamente si configurava l'AP in modo che attivasse questa modalità solo per pacchetti di dimensioni importanti altrimenti usavano la modalità normale CSMA/CA.

### 3.3.5 Frame 802.11

2	2	6	6	6	2	6	0 - 2312	4
frame control	duration	address 1	address 2	address 3	seq control	address 4	payload	CRC

- **frame control** = preambolo, descrive il tipo di frame;

2	2	4	1	1	1	1	1	1	1
protocol version	type	subtype	to AP	from AP	more frag	retry	more data	WEP	rsvd

- **to AP** = l'indirizzo MAC destinatario è l'AP;

- **from AP** = il mittente è l'AP.
  - **more frag** = per gestire la frammentazioni;
  - **retry** = dice se è un tentativo di ritrasmissione oppure no;
  - **power mgt** = gestione della potenza del segnale;
  - **WEP** = modalità di autenticazione;
  - **rsvd** = dati per la prenotazione del canale.
- **duration** = durata, nel caso di modalità con prenotazione;
  - **address 1** = indirizzo MAC del destinatario (nodo o AP);
  - **address 2** = indirizzo MAC del mittente (nodo o AP);
  - **address 3** = indirizzo MAC dell'interfaccia del router a cui è collegato l'AP;
  - **seq control** = sequential control necessario per la presenza dell'ACK;
  - **address 4** = indirizz MAC utilizzato solo in modalità ad hoc.

### 3.3.6 Mobilità nella stessa sotto-rete IP

Se un nodo cambia AP nella stessa subnet:

- L'indirizzo IP rimane invariato.
- Gli switch aggiornano le tabelle grazie al **self-learning**.
- In passato, l'AP trasmetteva in broadcast il nuovo indirizzo per aggiornare le tabelle.

### 3.3.7 Caratteristiche avanzate di 802.11

**Rate adaption** La base station può cambiare dinamicamente la codifica (quindi la velocità di trasmissione).

Alcune implementazioni di 802.11 sono in grado di selezionare adattivamente la tecnica di modulazione, variando di conseguenza il SNR. Quando un host si allontana dalla base station, il SNR diminuisce e aumenta il BER, quando il BER diventa troppo elevato, la stazione riduce il tasso di trasmissione per ridurlo.

**Power management** I nodi possono entrare in modalità **sleep**, notificandolo all'AP, che bufferizza i dati fino al risveglio del nodo, risparmiando energia.

## 3.4 Bluetooth 802.15

Bluetooth è progettato per reti con un raggio d'azione molto limitato (meno di 20 metri), denominate **Personal Area Network** (PAN). Il protocollo è nato con l'obiettivo di sostituire i cavi, offrendo una comunicazione wireless semplice e immediata. Utilizza una **rete ad hoc** senza infrastruttura, con una modalità di comunicazione basata sul modello **master/slave**, in cui un master centrale gestisce diversi slave. Sono previsti anche dispositivi in stato di *parked*, ossia in attesa.

### 3.4.1 Topologia (Piconet)

La topologia base di Bluetooth è detta **piconet** e presenta le seguenti caratteristiche:

- Possono essere attivi al massimo **8 dispositivi** in una piconet:
  - 1 **master**;
  - fino a 7 **slave**;
  - fino a 200 dispositivi **parked**.

- **Slave:**
  - devono rimanere sincronizzati con il master;
  - devono chiedere il permesso al master prima di trasmettere.
- **Master:**
  - gestisce la comunicazione e l'accesso al mezzo condiviso;
  - attiva i dispositivi *parked*.
- **Dispositivi parked:**
  - non partecipano alla comunicazione attiva;
  - rimangono sincronizzati con il master.

### 3.4.2 Scatternet

La topologia di base della piconet è limitata in termini di copertura e numero di dispositivi attivi. Per superare questa limitazione si utilizzano le **scatternet**, che collegano tra loro più piconet. La comunicazione tra le diverse piconet è garantita dai master delle rispettive reti.

### 3.4.3 Comunicazioni Bluetooth

Bluetooth opera nella stessa banda di frequenza utilizzata da 802.11 ( $2.4\text{ GHz}$ ), suddivisa in 79 canali. Per la modulazione di frequenza, utilizza la tecnica **2-FSK** e applica il **FHSS** (*Frequency Hopping Spread Spectrum*), una tecnica che prevede il cambio periodico della frequenza di trasmissione per ridurre le interferenze.

**FHSS (Frequency Hopping Spread Spectrum)** L'uso dell'FHSS riduce al minimo le interferenze cambiando rapidamente frequenza. I dispositivi passano da un canale all'altro **1600 volte al secondo**. Il master sceglie una sequenza pseudo-casuale di frequenze basata su un *seed*, e gli slave seguono questa sequenza rimanendo sincronizzati con il master.

### 3.4.4 Medium Access Control

**TDM (Time Division Multiplexing)** Il protocollo utilizza una divisione temporale (*TDM*) con slot di durata pari a  $625\mu\text{sec}$ .

- Negli slot pari trasmette il **master**.
- Negli slot dispari possono trasmettere gli **slave**.

La comunicazione è di tipo *single slave*: il master interroga uno slave alla volta, e se questo ha dati da trasmettere, utilizza lo slot successivo per farlo.

### 3.4.5 Modalità

Bluetooth supporta diverse modalità operative:

**Inquiry** Permette di scoprire altri dispositivi Bluetooth nelle vicinanze.

**Pairing** Consente di stabilire una connessione effettiva tra i dispositivi, che potranno successivamente comunicare.

**Durante la connessione** Durante una connessione, i dispositivi possono trovarsi in diverse modalità:

- **Active mode:** il dispositivo partecipa attivamente alla comunicazione;
- **Sniff mode:** il dispositivo entra in uno stato di basso consumo energetico, risvegliandosi periodicamente per ascoltare il master;
- **Hold mode:** il dispositivo dorme per un periodo specificato dal master;
- **Park mode:** il dispositivo è in stato di attesa (parked). Rimane sincronizzato con il master ma non partecipa alla comunicazione attiva. Si risveglia periodicamente per sincronizzarsi e ascoltare il master.

### 3.5 ZigBee 802.15.4

ZigBee è l'unico standard open source specificamente sviluppato per la domotica e i dispositivi IoT. Una delle caratteristiche principali di questo protocollo è il basso consumo energetico, essenziale per i dispositivi IoT, che trasmettono una quantità limitata di informazioni. ZigBee utilizza frequenze nella banda  $2.4\text{ GHz}$ . La tecnologia è supportata da un'alleanza di aziende che garantiscono l'interoperabilità dei dispositivi compatibili. Tuttavia, le connessioni offerte sono a bassa affidabilità.

Esistono tre bande di frequenza previste:

- $868\text{ MHz}$  (canale 0);
- $915\text{ MHz}$  (canali da 1 a 10);
- $2.4\text{ GHz}$  (canali da 11 a 26).

In totale sono disponibili **27 canali**.

#### 3.5.1 Topologia

ZigBee utilizza una topologia di tipo **mesh** con percorsi ridondanti. Questa topologia è scalabile, consente di coprire ampie aree e si basa su dispositivi a basso consumo energetico.



Figura 21: Topologia ZigBee

#### 3.5.2 Architettura

L'architettura ZigBee prevede due tipi di indirizzi:

- **64 bit:** identificano un dispositivo specifico (simile all'indirizzo MAC);
- **16 bit:** identificano il servizio (simile alle porte TCP/IP).

**Stack di ZigBee** Lo stack di ZigBee è strutturato come segue:

<b>Applications</b>		ZigBee specification
<b>Application Framework</b>		
<b>Network &amp; Security</b>		
<b>MAC Layer</b>	802.15.4	
<b>PHY Layer</b>		

Tabella 2: Struttura dello stack ZigBee

**Tipi di nodi** Nella rete ZigBee si distinguono tre tipi di nodi:

1. **PAN coordinator**: coordina la rete, scopre nuovi dispositivi e li configura;
  - Assegna un ID a 16 bit e il canale di comunicazione ai nuovi dispositivi.
2. **FFDs (Full Function Devices)**: dispositivi in grado di funzionare come router. Non gestiscono la rete ma inoltrano i pacchetti;
3. **RFDs (Reduced Function Devices)**: nodi terminali che raccolgono e inviano dati.

### 3.5.3 Medium access control

**Livello MAC** Il livello MAC gestisce i superframe e il controllo di accesso al canale. Valida i frame e invia gli acknowledgment (ACK) per garantire l'affidabilità della comunicazione.

**Slotted CSMA/CA** Gestito dal PAN coordinator, utilizza i superframe per controllare gli slot temporali.

**CSMA/CA** I nodi possono comunicare direttamente, in maniera simile alla modalità Ad Hoc del Wi-Fi.

**Superframe** Il superframe è un intervallo di tempo compreso tra due beacon frame, inviati periodicamente dal master.

Si tratta di un protocollo adattativo che include due sezioni principali:

- **CAP (Contention Access Period)**: fase in cui si utilizza lo slotted CSMA/CA con slot assegnati dal master;
- **CFP (Contention Free Period)**: fase libera gestita autonomamente dai nodi con CSMA/CA classico.

Il master decide e comunica, tramite il beacon, i parametri della rete, come la durata delle fasi CAP e CFP e il periodo di inattività.



Figura 22: Struttura del superframe ZigBee

**Frame** Sono definiti quattro tipi di frame:

1. **Beacon frame:** utilizzato dal coordinator per sincronizzare la rete;
2. **Data frame:** utilizzato per trasmettere i dati;
3. **Ack frame:** garantisce l'affidabilità del canale con conferme esplicite;
4. **MAC command frame:** utilizzato per configurare le interfacce.

### 3.6 Z-Wave

Z-Wave è un protocollo proprietario basato su una topologia *mesh*, progettato per la comunicazione tra dispositivi IoT (Internet of Things).

#### Caratteristiche principali

- I dispositivi comunicano in modalità *punto-a-punto* con una portata massima di 35 metri tra ciascun nodo.
- Le reti Z-Wave possono essere collegate fra loro, permettendo di coprire porzioni di spazio più ampie.
- Ogni singola rete può supportare fino a 232 dispositivi.
- Opera su frequenze radio a **868,42 MHz** (per l'Europa), garantendo comunicazioni a basso consumo energetico e interferenze ridotte rispetto alle frequenze Wi-Fi.

### Vantaggi della topologia *mesh*

- Aumenta la copertura: i dispositivi possono inoltrare messaggi ad altri nodi, estendendo la rete oltre il limite di 35 metri per singolo collegamento.
- Alta affidabilità: se un nodo fallisce, la rete può riorganizzarsi automaticamente utilizzando percorsi alternativi.

### 3.7 Reti cellulari 4G e 5G

Il termine cellulare si riferisce al fatto che la regione coperta da una rete cellulare è divisa in una serie di aree geografiche, dette **celle**. Ogni cella contiene una base station che trasmette e riceve i segnali dai dispositivi mobili presenti nella cella.

#### Similitudini con le reti cablate

- Vi è distinzione tra i dispositivi all'edge e al core;
- Entrambi sono composti da reti di reti;
- Si interfacciano direttamente con la rete internet pubblica;
- Utilizzano i protocolli standard (HTTP, DNS, TCP, UDP, IP, NAT, ecc.);
- Sono interconnessi con la rete internet cablata.

#### Differenze con le reti cablate

- Physical e data link layer molto diversi tra le due tecnologie;
- Focalizzata sulla mobilità dei nodi;
- L'identificazione degli utenti avviene tramite SIM card;
- Modello di business (gli utenti stipulano un contratto con il provider).

#### 3.7.1 Elementi di una rete 4G

- **Dispositivi mobili (UE)**: Si connettono alla rete cellulare. Hanno un identificatore unico detto IMSI (International Mobile Subscriber Identity) da 64 bit, memorizzato nella SIM;
- **Base station (eNode-B)**: Gestisce le risorse radio e i dispositivi mobili nella sua cella, coordinando l'autenticazione dei dispositivi e l'allocazione delle risorse; ha un ruolo attivo per la mobilità;
- **HSS (Home Subscriber Server)**: È un database che memorizza informazioni sui dispositivi nell'home network;
- **S-GW (Serving Gateway), P-GW (Packet Data Network Gateway)**: Sono due router nel percorso tra il dispositivo mobile e internet. P-GW svolge funzioni simili a un router gateway e fornisce servizi NAT;
- **MME (Mobility Management Entity)**: È un elemento del piano di controllo. Insieme all'HSS, fornisce autenticazione dei dispositivi, gestisce il passaggio di cella e il setup del tunneling tra il dispositivo mobile e internet.

### 3.7.2 Protocollo LTE

#### 3.7.3 Separazione piano controllo e dati

**Piano di controllo** Gli attori del control plane sono: la base station, l'MME, l'HSS e il P-GW. È un nuovo protocollo per la gestione della mobilità, sicurezza e autenticazione.



**Piano dati** Gli attori del piano dati sono: base station, S-GW e P-GW. È un nuovo protocollo a livello collegamento e fisico.



#### 3.7.4 Stack data plane

Dispositivo mobile	Base station
Application	IP
Transport	Packet Data Convergence
IP	Radio Link
Packet Data Convergence	Medium Access
Radio Link	Physical
Medium Access	
Physical	

- **Packet Data Convergence:** Si occupa della compressione dei pacchetti e della cifratura;
- **RLC (Radio Link Control):** Si occupa della frammentazione e riassettaggio dei datagrammi IP, e garantisce trasporto affidabile grazie al protocollo ARQ;



- **Medium Access:** Gestisce l'uso degli slot di trasmissione assegnati in modo dinamico ai vari dispositivi.

**LTE radio access network** I canali sono divisi in:

- **Downstream:** Dalla base station al dispositivo. Si utilizzano FDM e TDM tra le frequenze del canale (OFDM, Orthogonal Frequency Division Multiplexing);
- **Upstream:** Dal dispositivo alla base station. Si utilizzano FDM e TDM simile a OFDM.

Ad ogni dispositivo possono essere allocati, in modo dinamico, due o più slot di tempo da  $0,5\text{ ms}$  su  $12\text{ frequenze}$  diverse. Ci sono poi algoritmi di scheduling, non standardizzati, che assegnano in modo dinamico queste risorse radio (slot e frequenze) ai diversi dispositivi.

**Packet core**

Base station	S-GW
IP	GTP-U
Packet Data Convergence	UDP
Radio Link	IP
Medium Access	Link
Physical	Physical

La base station, oltre ai livelli utilizzati per comunicare con il dispositivo mobile, gestisce altri livelli. GTP-U permette di definire dei tunnel (su UDP) tra la base station e il S-GW. A sua volta, il S-GW ritunnellizza il datagramma al P-GW. Cambiano solo gli endpoint quando il dispositivo mobile si sposta da una parte all'altra della rete.

**Associazione con una base station**

1. La BS fa un broadcast su tutti i canali per dare informazioni sui servizi che offre e si identifica per permettere ai dispositivi di sincronizzarsi con la base station;
2. Il dispositivo mobile fa uno scanning (partizionato) della rete finché non trova un primo segnale. Da questo primo segnale trova la frequenza da usare. Quando trova le informazioni, viene a conoscenza dell'ampiezza del canale e delle configurazioni. Se è coperto da diverse celle può ricevere diverse informazioni da diverse base station;
3. Il dispositivo mobile seleziona con quale BS associarsi;
4. Ulteriori passi per l'autenticazione, stabilire la connessione e fare il setup del piano dati.

**Sleep mode** Anche i dispositivi mobili, per risparmiare batteria, hanno una modalità di sleep. Esistono due modalità:

- **Light sleep:** Ogni  $100\text{ msec}$  di inattività si sveglia e controlla se ci sono delle trasmissioni downstream, o se deve trasmettere;
- **Deep sleep:** Ogni  $5-10\text{ sec}$  di inattività si risveglia ma deve ristabilire l'associazione con la BS.

**Rete cellulare globale** L'home network di un utente è collegato alle reti degli altri carrier mobili e a internet tramite router gateway. Le reti mobili sono connesse tra loro tramite l'internet pubblico o un IPX (Internet Protocol Packet eXchange) Network.

### 3.7.5 Reti 5G

L'obiettivo del 5G è incrementare il bitrate di picco di 10 volte, diminuire la latenza di 10 volte e incrementare la capacità di traffico di 100 volte rispetto al 4G.

Questi obiettivi si cercano di raggiungere utilizzando due bande di frequenze molto alte: FR1 (450 MHz - 6 GHz) e FR2 (24 GHz - 52 GHz). Queste ultime sono chiamate millimeter wave frequencies perché la lunghezza d'onda è nell'ordine dei millimetri e le distanze raggiunte sono dai 10 ai 100 metri. Il 5G non è retrocompatibile con il 4G. Per avere la stessa copertura, è necessario utilizzare più BS (MIMO - Multiple Directional Antennae).

#### Vantaggi

- Con frequenze più elevate avremo più banda di trasmissione e maggiore velocità.

#### Svantaggi

- Frequenze più elevate, simile alla luce, hanno una portata minore;
- Sono più facilmente assorbite dagli ostacoli materiali.

### 3.8 Gestione della mobilità

Se un dispositivo si muove dal suo access network continuando a inviare datagrammi IP, la rete dovrà gestire l'**handover**.

#### Spettro di mobilità

No mobilità				Alta mobilità
Il dispositivo si muove ma stacca la connessione durante il passaggio		Il dispositivo si muove nella stessa AP in una provider network	Il dispositivo si muove tra diverse AP in una provider network	Il dispositivo si muove tra più provider network, mantenendo una connessione attiva

Tabella 3: Spettro della mobilità.

**Approccio** La gestione della mobilità non può avvenire a livello del core della rete. Si cerca di gestirla nell'*edge* della rete, utilizzando due modalità:

- **Routing indiretto:** La comunicazione passa per la *home network*, che funge da tramite verso la nuova rete in cui si è spostato il nodo.
- **Routing diretto:** Il corrispondente ottiene le informazioni necessarie per raggiungere direttamente il nodo mobile che si è spostato.

**Home network** Rete con cui il dispositivo ha un contratto con il provider. L'HSS conserva le informazioni relative all'identificativo e ai servizi disponibili.

**Visited network** Qualsiasi rete diversa dalla *home network*. I servizi sono stabiliti attraverso accordi tra le reti per garantire accesso ai dispositivi visitatori.

Nel contesto ISP/WiFi, la gestione globale di un utente non è usuale. Le credenziali sono conservate nel dispositivo o ricordate dall'utente. Differenti reti richiedono credenziali diverse, con alcune eccezioni come *eduroam*.

**Registrazione** La *home network* deve sapere dove si trova il dispositivo. Il dispositivo mobile si associa al *visited mobility manager*, che registra la posizione del dispositivo nell'HSS della rete *home*. Alla fine:

- Il *visited mobility manager* conosce l'esistenza del dispositivo mobile.
- L'HSS della rete *home* sa dove si trova il dispositivo.

### 3.8.1 Routing indiretto

1. I datagrammi sono inviati all'*home network*, indirizzati all'indirizzo permanente del dispositivo mobile.
2. Il gateway della *home network* intercetta i datagrammi, consulta l'HSS per determinare la *visited network* dove risiede il dispositivo mobile, incapsula i datagrammi originali in altri datagrammi e li inoltra al gateway della *visited network* tramite tunneling.
3. Il gateway della *visited network* riceve e decapsula i datagrammi, inoltrandoli al dispositivo mobile.
4. Opzioni:
  - (a) Il dispositivo invia i datagrammi tramite l'*home network*.
  - (b) Il dispositivo invia i datagrammi direttamente al corrispondente (*local breakout*).

#### Vantaggi

- Totalmente trasparente per il corrispondente, permettendo di mantenere connessioni già instaurate.

#### Svantaggi

- **Routing triangolare:** Inefficiente se il corrispondente e il dispositivo mobile si trovano nella stessa rete, poiché i dati passano prima dalla *home network*.

**Cambio di rete** Quando il dispositivo si sposta in un'altra rete, l'agente della nuova *visited network* informa quello della precedente e viene modificato l'estremo del tunnel.

### 3.8.2 Routing diretto

1. Il corrispondente contatta la *home network* e richiede l'indirizzo del dispositivo mobile.
2. Riceve l'indirizzo del dispositivo mobile.
3. Il corrispondente invia i datagrammi all'indirizzo nella *visited network*.
4. Il gateway della *visited network* inoltra i datagrammi al dispositivo mobile.

#### Vantaggi

- Risolve il problema del *routing triangolare*.

#### Svantaggi

- Non è più trasparente per il corrispondente.
- Se il dispositivo cambia rete, la gestione richiede complessità aggiuntiva.

**Cambio di rete** In caso di spostamento del dispositivo in una nuova rete, viene creato un tunnel tra la vecchia *visited network* e la nuova.

### 3.9 Mobilità nelle reti 4G

1. **Base station association:** Associazione del nodo mobile alla base station.
  - Il dispositivo invia le informazioni IMSI per identificarsi e identificare la *home network*.
2. **Control-plane configuration:** L'MME della *visited network* e l'HSS della *home network* stabiliscono un *control-plane* (tunnel per trasferire i dati).
3. **Data-plane configuration:** L'MME configura i tunnel di inoltro per il dispositivo mobile. La *visited network* e la *home network* stabiliscono tunnel dal P-GW della *home network* fino al dispositivo.
4. **Mobile handover:** Il nodo si associa a una nuova base station.

#### 3.9.1 Configurazione degli elementi del *control-plane* LTE

Il dispositivo mobile comunica con l'MME locale della *visited network*. Tramite le informazioni presenti nella SIM, identifica l'*home subscriber server* e lo contatta.

L'HSS fornisce informazioni sull'autenticazione, l'accesso alla rete, i servizi per cui il dispositivo ha pagato e i dettagli di rete. Ora l'HSS sa che il nodo si è spostato e aggiorna le sue informazioni sulla posizione del dispositivo mobile.

La base station, insieme al dispositivo mobile, seleziona i parametri necessari per configurare il canale dati. La base station assegna risorse radio al dispositivo, come slot di tempo e frequenze.



Figura 23: Configurazione degli elementi del control-plane LTE.

#### 3.9.2 Configurazione dei tunnel del *data-plane*

Si instaura un tunnel tra la base station associata al dispositivo mobile e un server gateway all'interno della rete. Un secondo tunnel si crea tra il server gateway e il packet gateway della *home network*. Il tunneling avviene tramite GTP, dove i datagrammi sono incapsulati in GTP dentro UDP.

In caso di *handover*, sarà sufficiente aggiornare gli estremi del tunnel.



Figura 24: Configurazione dei tunnel del data-plane.

### 3.9.3 Handover all'interno della stessa rete

Lo spostamento è gestito e deciso dall'infrastruttura (base station). I dispositivi mobili inviano periodicamente informazioni alla base station, come potenza del segnale, latenza, ecc. La base station, in base a queste informazioni e alla sua situazione di carico, può innescare un handover.

1. La base station sorgente seleziona una base station di destinazione e invia un messaggio di **handover request** alla BS di destinazione.
2. La BS di destinazione pre-allocata slot temporali radio e risponde con un **HR ACK** contenente le informazioni necessarie per il dispositivo mobile.
3. La BS sorgente informa il dispositivo mobile della nuova BS; il dispositivo mobile può ora comunicare tramite la nuova BS, completando apparentemente il trasferimento.
4. La BS sorgente smette di inviare datagrammi al dispositivo mobile e li inoltra invece alla nuova BS, che li trasmette al dispositivo mobile tramite il canale radio.
5. La BS di destinazione informa l'MME che è la nuova BS per il dispositivo mobile; l'MME istruisce l'S-GW a modificare l'endpoint del tunnel per puntare alla nuova BS di destinazione.
6. La BS di destinazione invia un ACK alla BS sorgente indicando che il trasferimento è completo e la BS sorgente può rilasciare le risorse.
7. I datagrammi del dispositivo mobile ora fluiscono attraverso il nuovo tunnel dalla BS di destinazione all'S-GW.



Figura 25: Procedura di handover all'interno della stessa rete.

### 3.9.4 Mobile IP

#### Architettura Mobile IP:

- Routing indiretto verso il nodo (tramite la rete domestica) utilizzando tunnel.
- **Mobile IP home agent:** Combina i ruoli dell'HSS e del P-GW domestico nel 4G.
- **Mobile IP foreign agent:** Combina i ruoli dell'MME e dell'S-GW nel 4G.
- Protocolli per:
  - la scoperta degli agenti nella rete visitata;
  - la registrazione della posizione visitata nella rete domestica tramite estensioni ICMP.

### 3.9.5 Impatto sui protocolli di livello superiore

Logicamente, l'impatto dovrebbe essere minimo poiché il sistema lavora su rete IP *best effort* con TCP e UDP che possono funzionare sulle reti wireless. Tuttavia, nella pratica emergono alcune problematiche:

- **Perdita/ritardo dei pacchetti:** Errori di bit causano pacchetti scartati e ritardi dovuti a ritrasmissioni a livello di collegamento, oltre a perdite durante l'*handover*.
- **Impatto su TCP:** TCP interpreta le perdite come congestione, riducendo inutilmente la finestra di congestione.
- **Ritardo e traffico in tempo reale:** Il ritardo ha un impatto negativo sul traffico in tempo reale.
- **Larghezza di banda limitata:** La larghezza di banda è una risorsa scarsa nei collegamenti wireless.

## 4 Architettura a microservizi

### 4.1 Cos'è un'architettura software

Un'architettura software di un sistema computerizzato è un insieme di strutture ed entità che servono per ragionare sul sistema. Queste comprendono:

- elementi software;
- relazioni tra questi elementi;
- proprietà dei componenti.

Le architetture possono essere rappresentate come una tripla:

- **elementi** - i componenti del sistema;
- **relazioni** - i legami tra i componenti;
- **proprietà** - le caratteristiche associate ai componenti.

La struttura di un'architettura è multidimensionale: a seconda della prospettiva adottata, si mettono in evidenza differenti aspetti del sistema. Questo consente di avere **visioni multiple** dello stesso prodotto, concentrandosi su diversi elementi, relazioni o proprietà.

### 4.2 Modello 4+1 viste

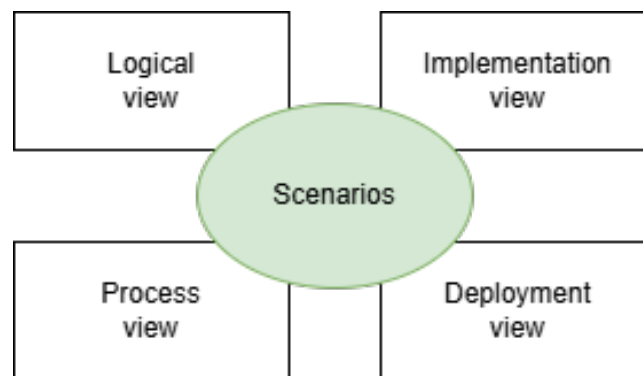


Figura 26: Rappresentazione del modello 4+1 viste.

Le viste considerate sono:

- **Vista logica:** dedicata a cosa lo sviluppatore genera.
  - **Elementi:** classi e package;
  - **Relazioni:** relazioni tra classi e package.



Figura 27: Esempio di vista logica.

Non dice nulla su come interagiscono o vengono implementate le classi.

- **Vista di implementazione:** considera i prodotti generati da un sistema di build.
  - **Elementi:** moduli (es. file JAR) e componenti (es. WAR file o eseguibili);
  - **Relazioni:** dipendenze tra i file.



Figura 28: Esempio di vista di implementazione.

- **Vista di processo:** descrive i componenti attivi.
  - **Elementi:** processi;
  - **Relazioni:** comunicazioni tra processi.

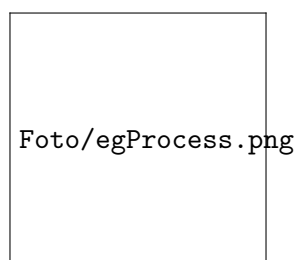


Figura 29: Esempio di vista di processo.



- **Vista di deployment:** descrive dove i processi vengono eseguiti nelle varie macchine.
  - **Elementi:** macchine e processi;
  - **Relazioni:** comunicazioni di rete tra processi.



Figura 30: Esempio di vista di deployment.

- **(+1) Scenari:** determinano cosa accade tra i componenti nei vari aspetti, derivati tipicamente da use case o user stories.



Figura 31: Esempio di scenario.

La decomposizione è importante perchè:

- facilita la divisione del lavoro e delle conoscenze, permettendo a più team di collaborare;
- definisce come gli elementi software interagiscono;
- influisce sulle caratteristiche dell'applicazione come scalabilità, manutenibilità e disponibilità.

### 4.3 Stile architetturale a strati (Layers/Tiers)



Figura 32: Architettura a strati.

Ogni strato ha specifiche responsabilità:

- **Presentation Layer/Interfaccia utente:** gestisce come le informazioni vengono presentate agli utenti (es. HTML, PHP, JavaScript).
- **Business Logic Layer/Application Layer:** implementa la logica di business, fornendo i servizi principali dell'applicazione.
- **Data Access Layer:** gestisce la persistenza dei dati attraverso i database.

#### 4.3.1 Differenza tra layer e tier

**Layer** Si riferisce a una separazione logica dei componenti software. Gli elementi sono organizzati logicamente in base alle loro responsabilità, senza riferimenti al deployment.

**Tier** Si riferisce a una separazione fisica a livello di deployment. Ogni livello è eseguito su macchine diverse tramite applicazioni indipendenti.

**Architettura monolitica** Quando la separazione è logica ma implementata in un unico eseguibile, si parla di architettura monolitica.

### 4.4 Architettura a esagoni

Questo stile è particolarmente adatto per lo sviluppo di microservizi. Le funzionalità sono suddivise in più eseguibili che interagiscono tra loro tramite primitive di comunicazione inter-processo (IPC).



Figura 33: Architettura a esagoni.

La struttura è suddivisa in:

- **Parte interna:** implementa la business logic focalizzata sul dominio applicativo.
- **Parte esterna:** gestisce l'interazione con il mondo esterno tramite inbound e outbound adapter.

**Vantaggi** Questa separazione permette di isolare le componenti stabili da quelle che possono cambiare più facilmente.

#### 4.5 Svantaggi dell'architettura monolitica

##### Esempio di architettura monolitica: FTGO (Food To Go)

FTGO coordina clienti, ristoranti e corrieri per la consegna di cibo, includendo funzionalità come pagamenti, notifiche e gestione degli ordini.



Figura 34: Architettura monolitica di FTGO.

Gli svantaggi di un'architettura monolitica includono:

- difficoltà di manutenzione con l'aumentare delle dimensioni del sistema;
- problemi di scalabilità;
- complessità nella gestione di versioni multiple;
- tempi lunghi per rilasciare aggiornamenti.

#### 4.5.1 Sviluppo di un architettura monolitica



Ogni team deve lavorare su un certo aspetto, l'eseguibile è unica, quindi è su un singolo repository dove ci lavorano tutti. All'aumentare dei gruppi e sviluppatori devono riuscire a coordinarsi e potrebbero insorgere numerosi problemi.

Il risultato è che lo sviluppo è ritardato per risolvere tutti i problemi di non coordinazione tra i gruppi.

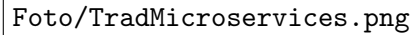
Questo tipo di sviluppo era sviluppato per un tipo di sviluppo waterfall.

#### 4.6 Traduzione all'architettura a microservizi

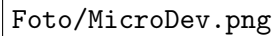
Ogni servizio gira tramite un diverso applicativo; possono essere sviluppati in modo totalmente diverso tra loro.

I microservizi interagiscono tra di loro attraverso IPCs (Inter Process Connection), in questo caso REST API.

In questo esempio c'è un microservizi che fa da API gateway per i vari client che prevede di smistare le richieste in base al microservizio richiesto. Vi è anche un'interfaccia web che inoltra le richieste ai microservizi giusti. Ogni microservizio può interagire con un altro e interagiscono tra di loro attraverso REST API.

A rectangular box with a thin black border, intended for a screenshot of a microservices architecture diagram.

#### 4.6.1 Sviluppo di un architettura a microservizi

A rectangular box with a thin black border, intended for a screenshot of a microservices development environment or diagram.

Visto che gli applicativi sono differenti, posso utilizzare dei repository differenti per ogni servizio con lo sviluppo del loro codice. Visto che il microservizio non è l'intera applicazione il team che lavorerà su uno specifico microservizio sarà più piccolo, più autonomo (forzo la separazione dei vari aspetti). Presumibilmente anche lo sviluppo è velocizzato perché questi microservizi sono più piccoli. I vari team hanno anche il vantaggio di non dover sapere tutto perché ognuno può sviluppare il proprio servizio con il linguaggio conosciuto. Anche l'introduzione di un nuovo elemento nel team è più facile perché il codice sarà più piccolo.

La parte più complicata è l'orchestrazione tra i vari servizi.

#### 4.6.2 Scalabilità

**Distribuzione** Un'architettura a microservizi offre un notevole vantaggio in termini di distribuzione, consentendo di eseguire i vari servizi su macchine differenti. Questo approccio elimina la necessità di un server centralizzato e particolarmente potente per gestire tutto il carico di lavoro.

Grazie alla distribuzione delle task su più macchine, il sistema beneficia di migliori prestazioni di I/O, dato che i processi possono lavorare in parallelo. Tuttavia, questa scalabilità è efficace solo se l'applicazione è tollerante ai ritardi causati dalla comunicazione di rete tra i vari servizi distribuiti.

**Decomposizione** Un'architettura a microservizi si basa sulla suddivisione di un'applicazione in piccoli servizi indipendenti, ognuno progettato per svolgere una funzione specifica. Questo processo di decomposizione permette di gestire ogni servizio separatamente, facilitando lo sviluppo, il test, il deployment e la manutenzione.

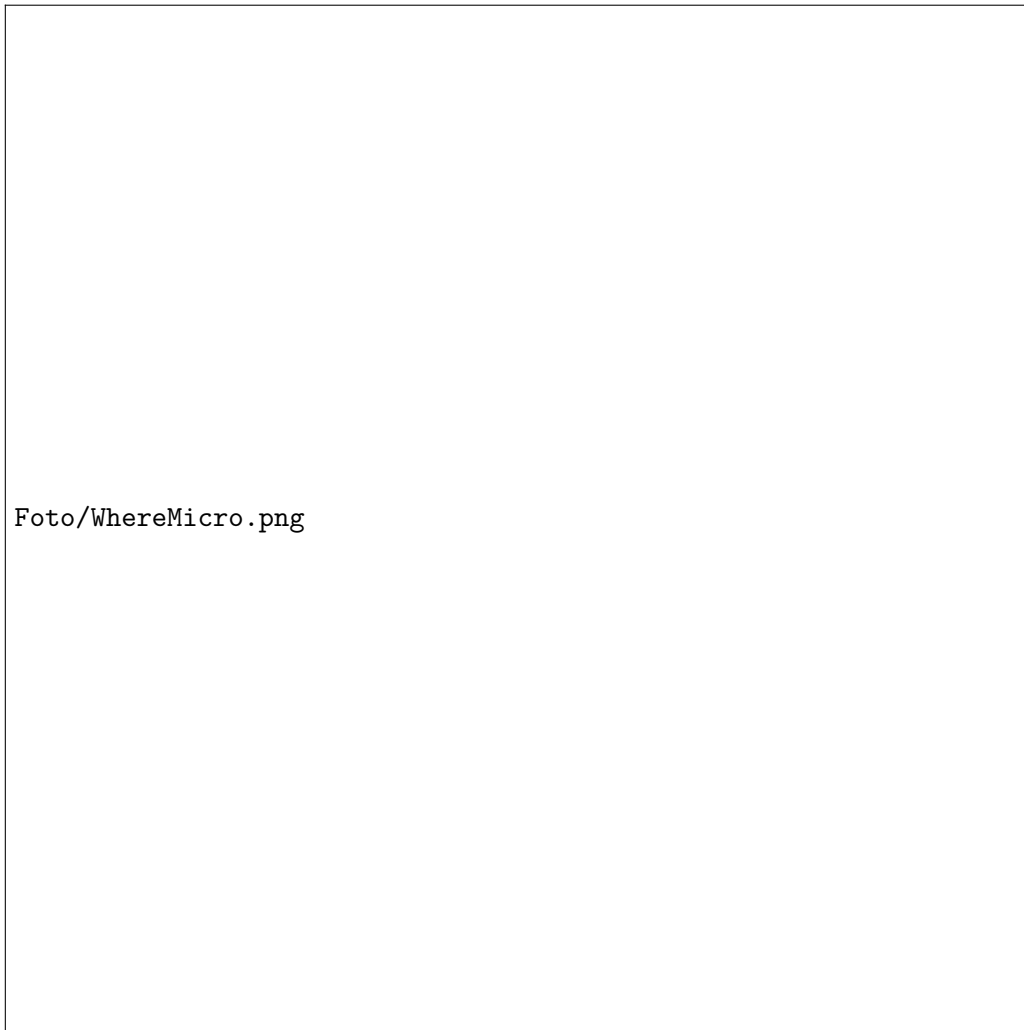
Ogni microservizio può essere implementato usando tecnologie diverse e aggiornato senza influire sugli altri, garantendo una maggiore flessibilità. La decomposizione consente anche di distribuire il lavoro tra diversi team, ognuno responsabile di un insieme di microservizi, promuovendo un approccio più modulare e organizzato nello sviluppo del software.

Inoltre, grazie a questa separazione, è possibile scalare in modo selettivo solo i microservizi che richiedono maggiori risorse, invece di dover gestire l'intera applicazione come un unico blocco.

**Scalabilità non uniforme** In un'architettura monolitica, la scalabilità si basa tipicamente sulla replica dell'intera applicazione, utilizzando un dispositivo di *load balancing* per indirizzare le richieste verso l'istanza meno carica. Questo approccio, sebbene funzionale, risulta inefficiente in termini di utilizzo delle risorse.

Con un'architettura a microservizi, invece, è possibile scalare in modo mirato, replicando solo i microservizi che presentano un carico di lavoro maggiore. Un *load balancer* si occuperà di distribuire le richieste tra le varie istanze di tali microservizi. In questo modo si ottimizza l'uso delle risorse, evitando colli di bottiglia e migliorando le prestazioni complessive del sistema.

### 4.6.3 Quando scegliere un'architettura a microservizi



In un sistema monolitico, è possibile sfruttare lo stato in memoria per gestire dati come le sessioni, riducendo significativamente la latenza e migliorando le prestazioni. Questo approccio è particolarmente vantaggioso perché tutti i componenti dell'applicazione condividono lo stesso spazio di memoria.

Al contrario, in un'architettura a microservizi, la natura distribuita del sistema impedisce la condivisione diretta dei dati in memoria tra i servizi. Di conseguenza, è necessario utilizzare soluzioni esterne come database o cache distribuite per la condivisione dei dati, il che può introdurre una latenza aggiuntiva.

Se un'applicazione fa un uso intensivo dello stato in memoria o non può tollerare i ritardi causati dalla condivisione dei dati in un'architettura distribuita, l'adozione di un'architettura a microservizi potrebbe non essere la scelta ideale. In questi casi, un sistema monolitico potrebbe risultare più appropriato per soddisfare i requisiti di prestazioni e semplicità.

## 4.7 Da request driven a event driven

Si passa da un sistema guidato dalle richieste (*request-response*) tra server e client, a un sistema in cui ci sono:

- Generatori di eventi;
- Un sistema che distribuisce questi eventi;
- Consumatori degli eventi, che a loro volta possono generarne altri.

Questo tipo di approccio è particolarmente adatto nei sistemi IoT.

#### 4.7.1 Evento

Un evento è qualunque cosa che può accadere (o non accadere) che comporta il cambiamento di uno stato. Una certa condizione su questo evento può innescare un'azione.

Gli eventi in tempo reale (*real-time*) inviano una notifica appena si verificano. Questo processo è tipicamente monodirezionale (*fire & forget*), ossia la notifica viene inviata senza attendere una risposta, seguita infine da una reazione.

#### 4.7.2 Benefici di event-driven

- Supporta le richieste di business per i vari servizi (evitando sistemi a lotti, con minore attesa);
- Nessuna interazione punto-punto (*fire & forget*);
- Garantisce tolleranza ai guasti, scalabilità, versatilità e altri benefici dell'accoppiamento lento (*loose coupling*);
- Offre una buona efficienza operativa.

### 4.8 Positività dei microservizi

- Forte scalabilità;
- Facilità di distribuzione;
- Supporto a scalabilità non uniforme;
- Elevata portabilità;
- Alta disponibilità;
- Indipendenza dal linguaggio di programmazione: è facile trovare sviluppatori per sviluppare microservizi;
- Per sistemi complessi, l'architettura a microservizi garantisce una maggiore produttività;
- Compatibile con metodologie *Agile*;
- Architettura adottata da aziende come: *Netflix*, *eBay*, *Amazon*, e molte altre.

#### 4.8.1 Negatività dei microservizi

- La granularità di un'architettura a microservizi è difficile da decidere;
- I microservizi richiedono una documentazione più precisa;
- Necessitano di essere integrati con sviluppo continuo e rilascio continuo (*CI/CD*), altrimenti i costi di gestione in termini di risorse umane diventano elevati;
- **NO SILVER BULLET!** Non esiste una soluzione unica adatta a tutti i contesti.

### 4.9 Definire un'architettura a microservizi

#### 4.9.1 Cos'è un servizio

Un servizio è un componente software che possiede le seguenti caratteristiche:

- **Standalone:** è autonomo e indipendente da altri componenti;
- **Indipendentemente deployabile:** può essere distribuito in modo autonomo.

Un servizio implementa una funzionalità utile e mette a disposizione un'API che permette di accedere alla sua funzionalità.

Un servizio può eseguire due tipi di operazioni:



- **Command:** operazioni che modificano lo stato del sistema;
- **Query:** operazioni che restituiscono informazioni senza modificare lo stato.

Inoltre, un servizio può pubblicare eventi ed è caratterizzato da una propria vista logica nell'architettura del sistema.

#### 4.9.2 Come fare

Il punto di partenza sono i requisiti dell'applicazione, concordati con il cliente, tipicamente si interagisce con gli esperti del dominio di business (tipicamente non operazioni che non cambiano nel tempo).

#### Processo a tre step

1. Identificare le operazioni del sistema;
2. Identificare i servizi;
3. Definire le API e le collaborazioni per i servizi.

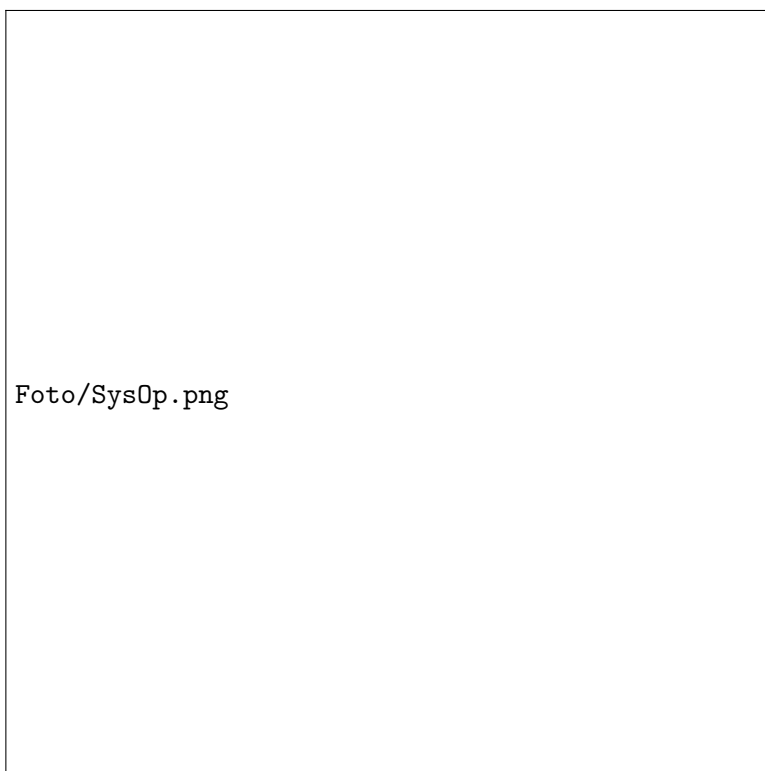
#### 4.9.3 Identificazione delle operazioni di sistema

Le operazioni di sistema rappresentano un'**astrazione** delle richieste che vengono gestite dalle applicazioni. Esse sono classificate in due tipologie principali:

- **Command:** utilizzate per aggiornare i dati;
- **Query:** utilizzate per recuperare i dati.

Il comportamento delle operazioni di sistema è definito in base al modello di dominio. Per identificare le operazioni di sistema, è necessario:

- Derivare un modello di dominio, ovvero un insieme di classi che fungono da vocabolario per descrivere le operazioni di sistema;
- Analizzare e identificare le operazioni di sistema rilevanti.



## User scenario 1

Consideriamo il seguente scenario:

- **Dato un consumatore:**
  - un ristorante;
  - un indirizzo di consegna e un orario che possono essere serviti da quel ristorante;
  - un totale dell'ordine che soddisfa il minimo richiesto dal ristorante.
- **Quando** il consumatore effettua un ordine presso il ristorante.

**Allora:**

- La carta di credito del consumatore viene autorizzata;
- L'ordine viene creato nello stato `PENDING_ACCEPTANCE`;
- L'ordine viene associato al consumatore;
- L'ordine viene associato al ristorante.

## User scenario 2

Consideriamo il seguente scenario:

- **Dato un ordine** che si trova nello stato `PENDING_ACCEPTANCE`;
- E un corriere disponibile per consegnare l'ordine.

**Quando:**

- Un ristorante accetta un ordine con una promessa di prepararlo entro un determinato orario.

**Allora:**

- Lo stato dell'ordine viene cambiato a `ACCEPTED`;
- Il campo `promiseByTime` dell'ordine viene aggiornato con l'orario promesso;
- Il corriere viene assegnato per consegnare l'ordine.

Dall'analisi complessiva dei casi d'uso si può identificare il modello del dominio dell'applicazione. Descritto da un diagramma a classi in cui ci sono le varie associazioni tra le diverse classi.



Figura 35: FTGO domain model

**Responsabilità delle classi** Le responsabilità delle classi sono definite in base a ciò che ciascuna classe sa o fa:

- **Consumer:** rappresenta un consumatore che effettua ordini.
- **Order:** descrive un ordine effettuato da un consumatore e ne traccia lo stato.
- **OrderLineItem:** rappresenta un elemento (line item) di un ordine.
- **DeliveryInfo:** specifica l'orario e il luogo di consegna dell'ordine.
- **Restaurant:** rappresenta un ristorante che prepara ordini per la consegna ai consumatori.
- **MenuItem:** descrive un piatto presente nel menu di un ristorante.
- **Courier:** rappresenta un corriere che consegna gli ordini, tracciando la sua disponibilità e posizione attuale.
- **Address:** gestisce gli indirizzi di consumatori e ristoranti.
- **Location:** definisce la latitudine e la longitudine di un corriere.

#### 4.9.4 Definizione delle operazioni di sistema

Dopo aver identificato gli elementi e gli oggetti delle classi bisogna identificare le operazioni da fare su questi oggetti.

Si analizzano i verbi in uno scenario o nelle storie.

Actor	Story	Command	Description
Consumer	Create Order	createOrder()	Creates an order
Restaurant	Accept Order	acceptOrder()	Indicates that the restaurant has accepted the order and is committed to preparing it by the indicated time

<b>Operation</b>	<i>createOrder(consumer id, payment method, delivery address, delivery time, restaurant id, order line items)</i>
<b>Returns</b>	<i>orderId, ...</i>
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• The consumer exists and can place orders;</li> <li>• The line items correspond to the restaurant's menu items;</li> <li>• The delivery address and time can be serviced by the restaurant.</li> </ul>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The consumer's credit card was authorized for the order total;</li> <li>• An order was created in the <i>PENDING_ACCEPTANCE</i> state.</li> </ul>

Tabella 4: Description of the createOrder operation

<b>Operation</b>	<i>acceptOrder(restaurantId, orderId, readyByTime)</i>
<b>Returns</b>	-
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• The <i>order.status</i> is <i>PENDING_ACCEPTANCE</i>;</li> <li>• A courier is available to deliver the order.</li> </ul>
<b>Post-conditions</b>	<ul style="list-style-type: none"> <li>• The <i>order.status</i> was changed to <i>ACCEPTED</i>;</li> <li>• The <i>order.readtByTime</i> was changed to the <i>readyByTime</i>;</li> <li>• The courier was assigned to deliver the order.</li> </ul>

Tabella 5: Description of the acceptOrder operation

#### 4.9.5 Definire le query

- Quando un cliente fa un ordine:
  1. L'utente inserisce l'ora e l'indirizzo della consegna;
  2. Il sistema mostra i ristoranti disponibili;
  3. L'utente seleziona il ristorante;
  4. L'utente seleziona l'oggetto e fa il check out;
  5. Il sistema crea l'ordine.
- *findAvailableRestaurants(deliveryAddress, deliveryTime)*
  - Restituisce il ristorante che può consegnare alla specifica ora allo specifico indirizzo.
- *findRestaurantMenu(id)*
  - Restituisce le informazioni del ristorante inclusi gli oggetti del menù.

#### 4.9.6 Decomposizione delle business capabilities

Le **business capability** sono qualcosa che il business fa per generare un certo valore (gestione dell'ordine, gestione di un inventario, ecc.).

Non interessa come un'operazione viene fatta ma l'operazione in sé. (Se voglio fare un prelievo non mi interessa come viene fatto, la capability è poter fare il prelievo)

Dipende dal tipo di business.



**Dalle business capabilities ai servizi** Definisco un servizio per ogni (sub)capability o gruppo di capabilities.



#### 4.9.7 Decomposizione da sotto domini (DDD)

Un altro modo per riuscire a decomporre in servizi una certa applicazione è sfruttare le conoscenze del dominio. In base alle conoscenze del dominio applicativo, ci sono già certi aspetti che sono confinati e ci sono poche interazioni tra di loro (chiamati **sotto domini**).

Si mappa ogni sotto dominio ad un servizio specifico.

Foto/DDD.png

Le business capabilities e i sotto domini sono molto simili. le business capabilities sono più legate a certi servizi (scomposizione **funzionale** dell'applicazione), nel caso del sotto dominio è legato all'area che si occupa di determinati aspetti che poi viene mappato in un certo servizio.

Ogni sotto dominio ha una visione diversa dell'intera applicazione, quindi il modello potrebbe essere diverso per ogni servizio.

#### 4.9.8 Principi della decomposizione

Non c'è una risposta preconfezionata per tutto quindi si seguono dei principi di base.

**Singola responsabilità** una classe dovrebbe avere un singolo motivo per cambiare, se differenti cambiamenti portano a dover cambiare questa classe non ha una singola responsabilità.

**Common closure Principle** una classe che appartiene ad un certo package dovrebbe essere soggetta allo stesso tipo di cambiamenti di un'altra classe nello stesso package.

#### 4.9.9 Mappatura delle operazioni e servizi

Mappatura delle operazioni di sistema astratte in servizi, seguendo i principi anche per i servizi.

Service	Operations
Consumer Service	<i>createConsumer()</i>
Order Service	<i>createOrder()</i>
Restaurant Service	<i>findAvailableRestaurants()</i>
Kitchen Service	<i>acceptOrder()</i> <i>noteOrderReadyForPickup()</i>
Delivery Service	<i>noteUpdateLocation()</i> <i>noteDeliveryPickedUp()</i> <i>noteDeliveryDelivered()</i>

**Collaborazione tra più servizi** Le operazioni di sistema sono gestite da singoli servizi e i servizi dovranno collaborare tra loro.

Esempio: *createOrder()* in Order Service dipende da:

- Consumer Service, verifica se il cliente può fare l'ordine e pagare;
- Restaurant Service,
  - valida gli oggetti dell'ordine;
  - verifica l'ora e l'indirizzo di consegna;

– ottiene il prezzo per gli oggetti ordinati.

- Kitchen Service, crea un Ticket;
- Accounting Service, autorizza la carta di credito del cliente.

#### 4.9.10 Identificazione delle collaborazioni

Dall'analisi delle varie operazioni astratte che devono essere svolte si può identificare una serie di collaborazioni tra un certo servizio, l'operazione che deve svolgere e con chi deve collaborare per svolgere l'operazione.

Da queste collaborazioni devo definire delle API per consentire la collaborazione.

Se la quantità di collaborazioni è eccessiva si può pensare di aggregare più servizi in uno unico per diminuire il numero.

Service	Operations	Collaborators
Consumer Service	<i>verifyConsumerDetails()</i>	-
Order Service	<i>createOrder()</i>	Consumer Service <i>verifyConsumerDetails()</i> Restaurant Service <i>verifyOrderDetails()</i> Kitchen Service <i>createTicket()</i> Accounting Service <i>authorizeCard()</i>
Restaurant Service	<i>findAvailableRestaurants()</i> <i>verifyOrderDetails()</i>	-
Kitchen Service	<i>createTicket()</i> <i>acceptOrder()</i> <i>noteOrderReadyForPickup()</i>	Delivery Service <i>scheduleDelivery()</i>
Delivery Service	<i>scheduleDelivery()</i> <i>noteUpdatedLocation()</i> <i>noteDeliveryPickedUp()</i> <i>noteDeliveryDelivered()</i>	-
Accounting Service	<i>authorizeCard()</i>	-



#### 4.9.11 Scelta del paradigma di comunicazione

##### Architettura senza broker (comunicazione diretta)

- + Traffico di rete inferiore e migliore latenza;

- + No single point of failure;
- - Necessario fare il service discovery;
- - Ridotta disponibilità.

#### **Architettura basata su broker**

- + Disaccoppia i vari servizi (posso aggiungere a run-time nuovi servizi);
- + Buffering dei messaggi;
- - Bottleneck delle performance;
- - Single point of failure (broker);
- - Complessità aggiuntiva.