

# Reti di telecomunicazioni

sav

2025

# Contents

<b>1 Conoscenze di base</b>	<b>3</b>
1.1 Informatica . . . . .	3
1.1.1 Conversione binario esadecimale . . . . .	3
1.1.2 Tipologia reti . . . . .	3
1.2 Matematica . . . . .	5
1.2.1 Potenza di due . . . . .	5
1.2.2 Complemento a 1 - NOT . . . . .	6
1.2.3 Aritmetica a 2 - XOR . . . . .	6
1.3 Fisica . . . . .	7
1.3.1 Comunicazione radio, cablata e ottica . . . . .	7
1.4 Reti di telecomunicazioni - basi . . . . .	8
1.4.1 Comutazione di circuito e commutazione di pacchetto . . . . .	8
1.4.2 Ritardo end-to-end . . . . .	9
<b>2 Architetture di protocolli</b>	<b>11</b>
2.1 Modello ISO-OSI . . . . .	11
2.1.1 Segmentazione e riassemblaggio (incapsulamento e decapsulamento) . . . . .	12
2.2 Modello ibrido . . . . .	13
2.2.1 Enc/decapsulation dati nel modello ibrido . . . . .	13
2.2.2 Comunicazione tra sistemi . . . . .	14
2.3 Modello TCP/IP . . . . .	15
<b>3 Livello 4 trasporto TCP/UDP</b>	<b>16</b>
3.1 Nozioni preliminari . . . . .	16
3.1.1 Sock e ports . . . . .	16
3.2 UDP - datagram . . . . .	18
3.2.1 Calcolo checksum . . . . .	19
3.3 TCP - segmento . . . . .	20
3.3.1 Apertura connessione client-server (3way handshake) . . . . .	21
3.3.2 Chiusura connessione . . . . .	22
3.3.3 TCP sequenze di stati . . . . .	24
3.3.4 Trasmissione dei segmenti - sliding window . . . . .	25
3.3.5 Finestre di trasmissione e ricezione . . . . .	26
3.3.6 Controllo di flusso - come gestisce il TCP il flusso dei dati? . . . . .	26
3.3.7 Perdita di segmenti - RTO e 3 dupack . . . . .	27
3.3.8 Controllo di congestione TCP . . . . .	28

<b>4 Livello 3 rete</b>	<b>31</b>
4.1 Internet Protocol (IP) version 4 . . . . .	31
4.1.1 Datagramma IP . . . . .	32
4.1.2 Indirizzo IP - classful . . . . .	34
4.1.3 Indirizzo IP - classless CIDR . . . . .	34
4.1.4 Address Resolution Protocol (ARP) - mappatura IP a MAC . . . . .	35
4.1.5 Dynamic Host Configuration Protocol (DHCP) - configurazione automatica degli indirizzi IP tramite server DHCP . . . . .	38
4.1.6 Zero-configuration networking . . . . .	39
4.1.7 Network Address Translation (NAT) - da IP privato a IP pubblico . . . . .	39
4.1.8 Internet control message protocol (ICMP) - diagnostica della rete . . . . .	40
4.2 Algoritmi di routing . . . . .	41
4.2.1 Introduzione agli algoritmi di routing - tipologie . . . . .	41
4.2.2 Metriche per il costo del percorso . . . . .	42
4.2.3 Instradamento gerarchico - protocolli intra e inter AS . . . . .	43
4.2.4 Distance Vector . . . . .	43
4.2.5 Algoritmo Bellman-Ford . . . . .	44
4.2.6 Link State e Dijkstra . . . . .	48
4.2.7 Problemi di routing - routing loop . . . . .	51
4.2.8 Protocolli di routing intra AS (RIP - OSPF) . . . . .	52
4.2.9 Protocolli di routing inter-AS - (BGP) . . . . .	55
<b>5 Livello 2 Data link</b>	<b>56</b>
5.1 Strategie di controllo errore . . . . .	56
5.1.1 Protocolli ARQ (Automatic Repeat-reQuest) . . . . .	56
5.1.2 Efficienza di stop and wait in assenza di errori . . . . .	57
5.1.3 Efficienza di stop and wait in presenza di errori . . . . .	58
5.1.4 Protocolli Continuous ARQ . . . . .	61
5.1.5 Dimensione delle finestre di trasmissione/ricezione . . . . .	66
5.1.6 Efficienza GBN & SR . . . . .	66
5.1.7 Efficienza GBN & SR . . . . .	68
5.1.8 Efficienza con riscontri fuori dalla finestra di trasmissione $W_s$ . . . . .	69
5.1.9 Confronto tra protocolli di linea . . . . .	70
5.2 Livello 2 Frame DHLC . . . . .	71
5.2.1 Struttura del frame . . . . .	71
5.2.2 Bit stuffing . . . . .	71
5.2.3 Rivelazione dell'errore con CRC . . . . .	71
5.2.4 Protocollo point to point (PPP) . . . . .	74
5.3 Protocolli di accesso . . . . .	76
5.3.1 Troughput di rete . . . . .	77
5.3.2 Protocollo ALOHA puro . . . . .	78
5.3.3 Protocollo ALOHA slotted - prestazioni . . . . .	81
5.3.4 Protocolli Carrier Sense Multiple Access - CSMA (rilevazione canale) . . . . .	82
5.4 Il Progetto IEEE 802 . . . . .	84
5.5 Reti wireless . . . . .	85
5.5.1 Wireless LAN - WLAN . . . . .	85
5.5.2 Struttura dei canali 802.11b . . . . .	86
5.5.3 Nodi nascosti ed esposti . . . . .	88
5.5.4 Dispositivi wireless . . . . .	89

# Chapter 1

## Conoscenze di base

### 1.1 Informatica

#### 1.1.1 Conversione binario esadecimale

La conversione numerica si ottiene raggruppando i bit del numero binario in gruppi da quattro a partire dal punto e in entrambe le direzioni, sostituendo poi ogni gruppo di 4 bit con l'equivalente cifra del sistema esadecimale.

- |            |            |            |            |
|------------|------------|------------|------------|
| • 0000 = 0 | • 0100 = 4 | • 1000 = 8 | • 1100 = C |
| • 0001 = 1 | • 0101 = 5 | • 1001 = 9 | • 1101 = D |
| • 0010 = 2 | • 0110 = 6 | • 1010 = A | • 1110 = E |
| • 0011 = 3 | • 0111 = 7 | • 1011 = B | • 1111 = F |

#### 1.1.2 Tipologia reti

Le reti di telecomunicazione sono costituite da un insieme di nodi e rami che consentono la trasmissione di informazioni tra i vari nodi. Le reti possono essere classificate in base a diversi criteri:

- **Estensione geografica:**
  - **PAN** (Personal Area Network): copre pochi metri, ad esempio collegamento tra dispositivi personali (Bluetooth).
  - **LAN** (Local Area Network): copre un edificio o un campus, tipicamente reti aziendali o domestiche.
  - **MAN** (Metropolitan Area Network): copre una città o un'area metropolitana.
  - **WAN** (Wide Area Network): copre aree geografiche molto estese, anche nazioni o continenti (es. Internet).

- **Topologia:**

- **Bus:** tutti i nodi sono collegati a un unico canale di comunicazione.
- **Stella:** tutti i nodi sono collegati a un nodo centrale.
- **Anello:** i nodi sono collegati in modo circolare.
- **Maglia (mesh):** ogni nodo è collegato a più nodi, anche direttamente.
- **Albero:** struttura gerarchica a livelli.
- **Catena (chain):** i nodi sono collegati in sequenza lineare.
- **Full mesh:** ogni nodo è collegato a tutti gli altri nodi.

### Nodi e rami

I nodi e i rami(link) sono gli elementi che costituiscono una rete. I nodi sono i punti di intersezione tra i rami e possono essere di diversi tipi:

- Host: un nodo che genera o consuma informazioni
- Router: un nodo che instrada le informazioni tra i vari host
- Switch: un nodo che instrada le informazioni tra i vari host, ma a livello di collegamento
- Bridge: un nodo che collega due reti locali
- Gateway: un nodo che collega due reti diverse

Il ramo è il sistema trasmissivo che consente il trasporto delle unità informative da nodo a nodo secondo tre possibili modalità di trasmissione:

- simplex(unidirezionale): un solo verso di trasmissione, esempio: radio
- half-duplex(bidirezionale alternato): due direzioni di trasmissione, ma non contemporaneamente, esempio: walkie-talkie
- full-duplex(bidirezionale contemporanea): due direzioni di trasmissione contemporaneamente, esempio: telefono
- broadcast: un nodo invia informazioni a tutti gli altri nodi della rete, esempio: radio

### Modalità di invio dati

- broadcast: un nodo invia informazioni a tutti i nodi della rete, esempio: radio
- multicast: un nodo invia informazioni a un gruppo di nodi della rete, esempio: videoconferenza
- unicast: un nodo invia informazioni a un solo nodo della rete, esempio: email
- point-to-point: un nodo invia informazioni a un altro nodo della rete, esempio: telefono

### Reti di accesso e reti di trasporto

Se mando una richiesta sul web tramite il mio router di casa, collegandomi alla rete tim(rete di accesso), da quel punto in poi tim si appoggerà ad un'altra rete per rispondere alla mia richiesta(rete di trasporto), quindi è possibile distinguere velocità differenti:

- velocità di accesso(relativa ai dati che viaggiano nella rete di accesso, tipicamente agli utenti semplici non viene garantita una velocità minima(da casa non abbiamo questa garanzia), nella rete di trasporto invece viene garantita(contratto SLA))
- velocità di trasporto(relativa ai dati che viaggiano nella rete di trasporto)

## Differenza tra quantità di valori rappresentabili e byte occupati in memoria

Quando si lavora con i dati binari, è importante distinguere tra:

- **Quantità di valori rappresentabili:** indica il numero di combinazioni possibili che si possono ottenere con un certo numero di bit. Ad esempio, con  $n$  bit si possono rappresentare  $2^n$  valori distinti. Questo determina l'intervallo dei numeri che si possono codificare. Ad esempio, con 16 bit (senza segno) si possono rappresentare i numeri interi da 0 a  $2^{16} - 1 = 65535$ .
- **Byte occupati in memoria:** rappresenta lo spazio fisico necessario per memorizzare quel valore. Poiché 1 byte corrisponde a 8 bit, un valore a 16 bit occupa esattamente 2 byte in memoria. Questo è indipendente dai valori specifici che si possono rappresentare con quei bit.

In sintesi, *la quantità di valori rappresentabili dipende dal numero di bit*, mentre *i byte occupati in memoria dipendono da come è strutturato il campo nella memoria del computer*. Un campo a 16 bit occupa sempre 2 byte, ma può rappresentare fino a 65536 valori distinti (da 0 a 65535 nel caso senza segno).

## 1.2 Matematica

### 1.2.1 Potenza di due

$2^1$	=	2	$2^{11}$	=	2 048	$2^{21}$	=	2 097 152
$2^2$	=	4	$2^{12}$	=	<b>4 096</b>	$2^{22}$	=	4 194 304
$2^3$	=	8	$2^{13}$	=	8 192	$2^{23}$	=	8 388 608
<b>2<sup>4</sup></b>	=	16	$2^{14}$	=	16 384	<b>2<sup>24</sup></b>	=	<b>16 777 216</b>
$2^5$	=	32	$2^{15}$	=	32 768	$2^{25}$	=	33 554 432
$2^6$	=	64	<b>2<sup>16</sup></b>	=	<b>65 536</b>	$2^{26}$	=	67 108 864
$2^7$	=	128	$2^{17}$	=	131 072	$2^{27}$	=	134 217 728
<b>2<sup>8</sup></b>	=	256	$2^{18}$	=	262 144	<b>2<sup>28</sup></b>	=	<b>268 435 456</b>
$2^9$	=	512	$2^{19}$	=	524 288	$2^{29}$	=	536 870 912
$2^{10}$	=	1 024	<b>2<sup>20</sup></b>	=	<b>1 048 576</b>	$2^{30}$	=	1 073 741 824

Figure 1.1: Tabella delle potenze di due

### 1.2.2 Complemento a 1 - NOT

Il complemento a 1 è una rappresentazione binaria in cui ogni bit di un numero binario viene invertito. Ad esempio, il complemento a 1 di 1010 è 0101. Questo metodo è spesso utilizzato per rappresentare numeri negativi nei sistemi binari.

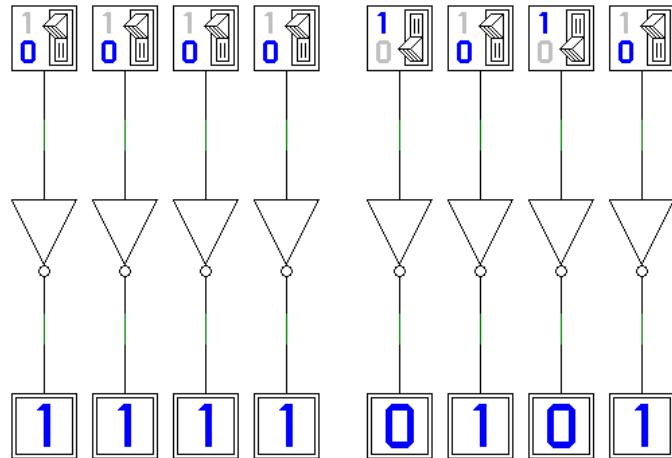


Figure 1.2: Esempio di complemento a 1

### 1.2.3 Aritmetica a 2 - XOR

XOR è un operatore logico, chiamato anche OR esclusivo, che restituisce TRUE solo se uno e solo uno dei suoi operandi è TRUE.

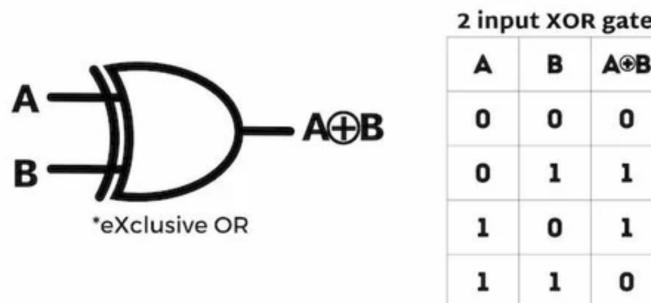


Figure 1.3: Esempio di operazione XOR

## 1.3 Fisica

### 1.3.1 Comunicazione radio, cablata e ottica

la trasmissione di un segnale a distanza avviene attraverso le onde elettromagnetiche, che siano trasmesse tramite antenne e quindi che si propaghino nel vuoto oppure in un cavo di fibra ottica, che anch'esso trasmette la luce, quindi l'onda em. dopo aver trasformato l'informazione in bit(codifica) questa viene trasmessa in dei canali(radio, mezzi guidati). quella radio avviene nello spazio libero(attraverso l'aria o nel vuoto), mezzi guidati(cavi)

- Mezzi guidati: cavi coassiali, cavi in fibra ottica, cavi in rame
- Mezzi non guidati: onde radio, microonde, infrarossi
- Mezzi ottici: fibra ottica

#### Pro e contro del canale radio

- Pro: risparmio sul mezzo fisico(l'aria e il vuoto sono gratis), un segnale radio non ha confini, può arrivare all'infinito, teoricamente. a noi interessa che il segnale elettromagnetico arrivi dove voglio e che venga decodificato correttamente, e non a infinito. un'antenna è un dispositivo che genera un campo magnetico(è un pezzo di ferro su cui scorre della corrente e che quindi genera un campo magnetico, fisica 2), aiuta la comunicazione radio.
- Contro: è più difficile gestire la privacy e sicurezza perchè il canale(l'aria, il vuoto) è condiviso tra tutti gli utenti. il canale radio ha più rumore(dovuto ad effetti naturali, il calore ad esempio) rispetto al cavo. l'interferenza è invece dovuta ad altre comunicazioni e non ad effetti naturali, nel canale radio anche l'interferenza può essere maggiore. il fading è un problema del canale radio, ossia il segnale non viaggia correttamente per via di ostacoli come le montagne. problema dell'effetto doppler. problema dei cammini multipli

il mezzo radio è passabanda, in base alla frequenza con la quale l'onda viene trasmessa quest'ultima si comporta in modo differente. ad esempio a bassa frequenza si sfrutta il fenomeno del waving per il quale l'onda riflette sulla ionosfera e quindi si possono raggiungere distanze più elevate e per esempio attraversare l'oceano. quando trasmetto via radio in base alla potenza con cui trasmetto regolo la frequenza con cui invio il segnale. maggiore potenza significa frequenza più alta. frequenza più alta vuol dire anche maggiore velocità di trasmissione ma copertura minore.

#### Canale radio e ostacoli fisici

- los(line of sight): il segnale radio non ha ostacoli, trasmissione diretta
- nlos(non line of sight): il segnale radio ha ostacoli, trasmissione non diretta

**Mezzi guidati** Esistono i mezzi guidati(cavi) come il doppino, il cavo coassiale e la fibra ottica. il doppino ha al suo interno 2 fili, la corrente per scorrere ha bisogno di un circuito chiuso. infatti questo circuito chiuso si ottiene tramite due fili all'interno del cavo(doppino). quindi si usa un doppino per trasmettere e un doppino per ricevere, quindi 4 fili in totale, cioè due circuiti chiusi. ricordare che i doppini possono essere di diversi tipi, viene principalmente usato il tipo UTP(unshielded twisted pair), coppia di conduttori intrecciati, non schermato(uno schermo serve ad evitare problemi con le correnti parassite, dovute ad altri fenomeni, non riguardano questo corso). quello schermato costa di più il cavo coassiale ha 8 fili. la fibra ottica trasmette onde em, non corrente (la corrente è un flusso di cariche elettriche e non è un'onda em), la banda di trasmissione è molto alta,  $10^{14}\text{Hz}$ . I cavi in cui scorre corrente vengono tipicamente intrecciati per annullare gli effetti dei campi magnetici indotti in maniera vicendevole, così da non avere disturbo sul passaggio della corrente e quindi dell'informazione.

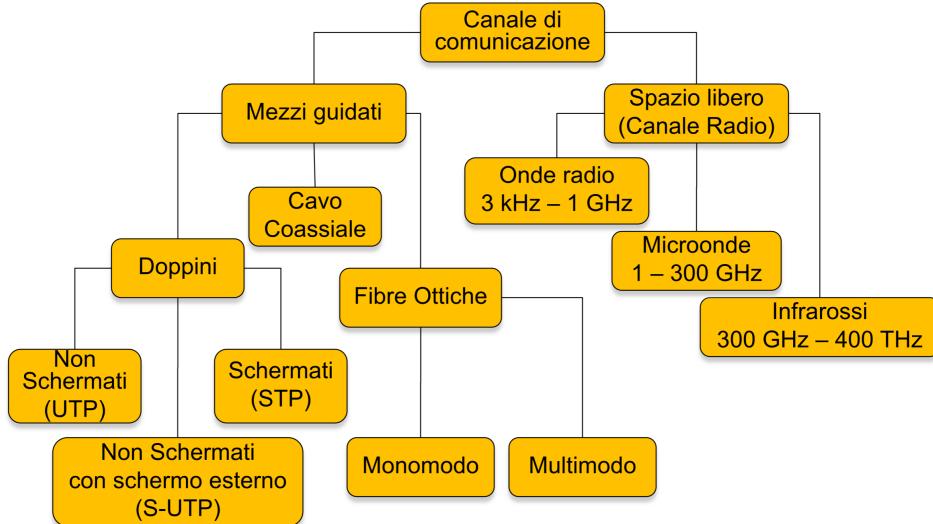


Figure 1.4: Tipi di canali di comunicazione

## 1.4 Reti di telecomunicazioni - basi

### 1.4.1 Comutazione di circuito e commutazione di pacchetto

**Comutazione di circuito** Si usava negli anni 20 del 900, quando chiamavo qualcuno inserivo il numero e nella centrale c'era qualcuno che fisicamente connetteva il mio cavo con quello a cui corrisponde il numero che ho digitato, rendendo il mezzo di comunicazione tra le due persone lo stesso cavo.

**Commutazione di pacchetto** l'informazione viene impacchettata e divisa in unità, questo vuol dire che ci saranno dei ritardi dovuti a quest'impacchettamento. Le informazioni riguardanti il pacchetto stesso sono contenute nell'header, mentre nel payload c'è l'informazione, i dati che voglio trasportare. i nodi intermedi decidono come gestire il trasporto(commutazione) di tali pacchetti leggendo il contenuto dell'header. l'operatore ha un miglior uso di risorse. un problema è che arrivano più pacchetti allo stesso nodo e devono attendere la commutazione del nodo che è arrivato prima.

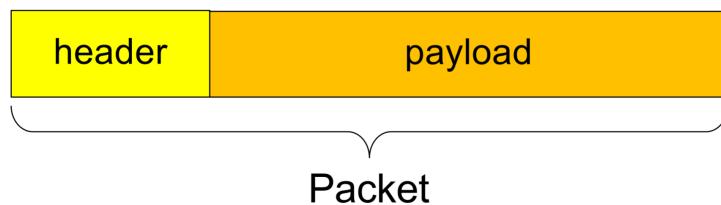


Figure 1.5: Comutazione di pacchetto

Due tipi di commutazione di pacchetto:

- Datagram: ogni pacchetto, che prende il nome di datagramma, contiene nell'header le informazioni del mittente e destinatario. i nodi decidono la strada che deve fare il pacchetto(routing), questo viene fatto
- Virtual circuit: la comunicazione avviene in 3 fasi (setup, data exchange, teardown) e l'instradamento è effettuato solo durante il setup. Pertanto tutti i pacchetti vengono instradati lungo lo stesso percorso (circuito), ma i nodi intermedi non devono ogni volta rieffettuare il routing, ma solo lo switching (inoltrare il pacchetto dall'ingresso all'uscita corretta).

### 1.4.2 Ritardo end-to-end

Il ritardo end-to-end è il tempo che intercorre tra l'invio di un pacchetto da parte del mittente e la ricezione dello stesso pacchetto da parte del destinatario. Questo ritardo è influenzato da diversi fattori, tra cui il tipo di rete, la distanza tra i nodi, la congestione della rete e le caratteristiche dei nodi stessi.

Il ritardo nell'invio di un'informazione da mittente a destinatario ha più componenti:

**Ritardo di trasmissione:** è il tempo necessario a inviare i bit sul mezzo,  $T_f = \frac{L}{C}$ , ad esempio il ritardo per la trasmissione di 1000 bit (informazione) su un cavo (canale) da 1 Gbit,  $T_f = \frac{1000}{10^9}$ , dove  $L$  è la lunghezza in bit dell'informazione,  $C$  è la frequenza di cifra.

**Ritardo di propagazione:** è il tempo impiegato dai dati per attraversare la rete, tipicamente è il tempo che impiegano le onde elettromagnetiche a viaggiare in un mezzo di una certa lunghezza:  $t = \frac{d}{v}$ , dove  $v$  è la velocità di propagazione (velocità della luce) e  $d$  è la distanza considerata.

**Ritardo di elaborazione:**  $T_p$ , è il tempo necessario all'elaborazione di un pacchetto in un nodo, dipende dalla velocità di calcolo dei processori.

**Ritardo in accodamento:**  $T_q$ , tempo di attesa in coda nel buffer di ricezione e dipende dal traffico generato da tutti i flussi che attraversano il nodo in considerazione, è quello più difficile da prevedere e calcolare.

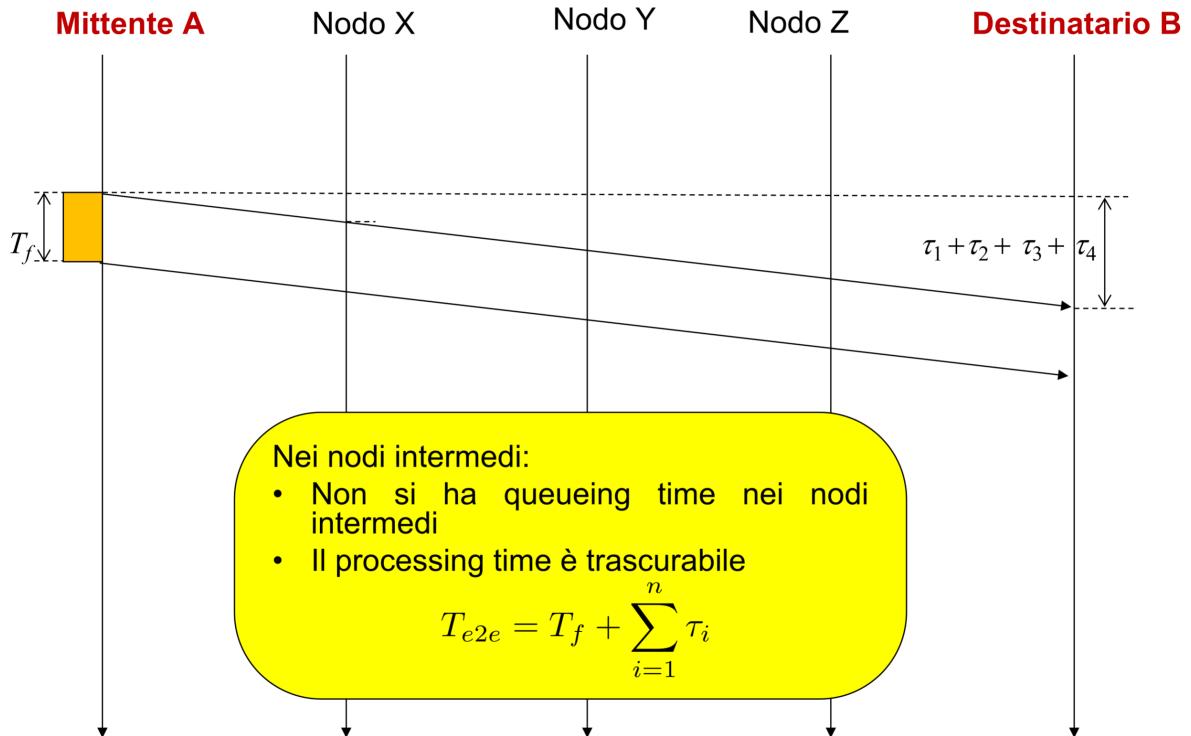


Figure 1.6: Ritardo e2e circuit switching: sommo i ritardi dovuti al mezzo(ritardo di trasmissione) e i ritardi dovuti alla propagazione nel mezzo

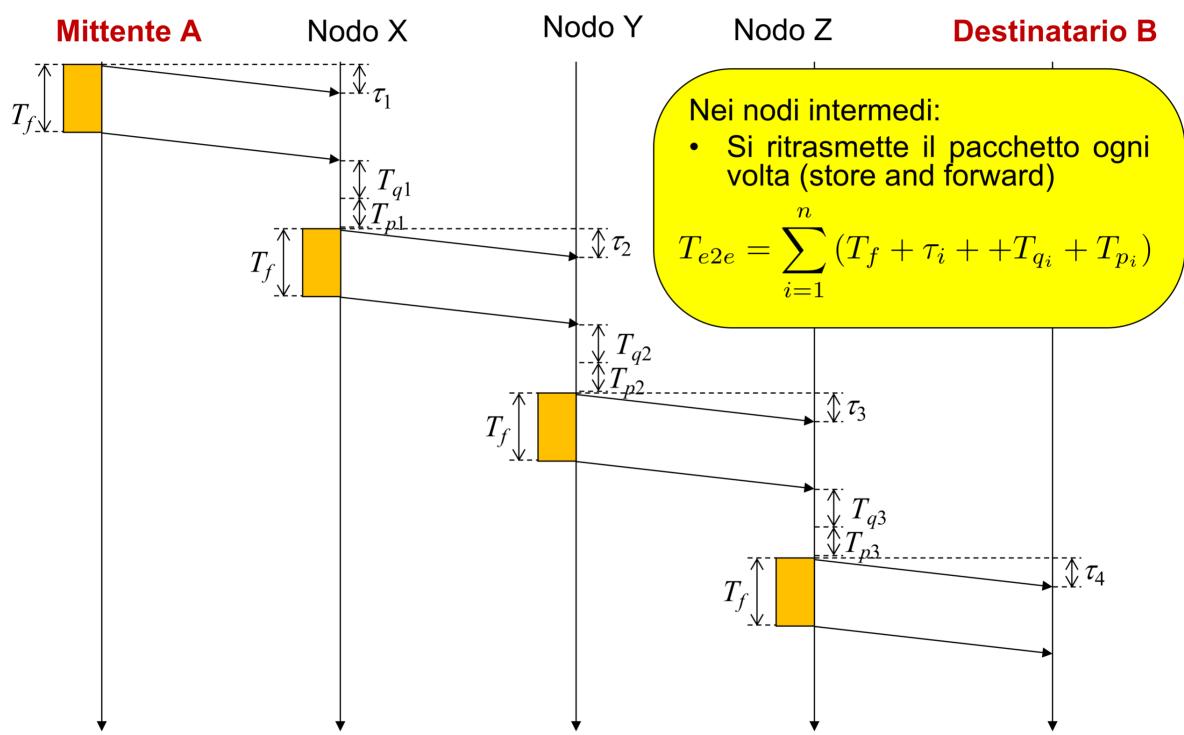


Figure 1.7: Ritardo e2e packet switching: (sommo TUTTI i ritardi) il packet switching ha una latenza maggiore rispetto al circuit switching

# Chapter 2

## Architetture di protocolli

### 2.1 Modello ISO-OSI

#### Introduzione

Il modello ISO-OSI è un modello di riferimento per la progettazione e l'implementazione di reti di telecomunicazioni. È stato sviluppato dall'Organizzazione Internazionale per la Standardizzazione (ISO) negli anni '80 e fornisce un framework concettuale per comprendere come i diversi protocolli e tecnologie di rete interagiscono tra loro.

ISO OSI non ha un numero fisso di livelli, è importante capire la logica dei livelli che devono comporre un modello iso osi ma il numero specifico dei livelli può variare a seconda della necessità.

1. **Livello fisico:** si occupa della trasmissione dei dati attraverso il mezzo fisico (cavi, fibre ottiche, onde radio, ecc.). Definisce le caratteristiche elettriche e meccaniche dei dispositivi di rete.
2. **Livello di collegamento dati:** gestisce la comunicazione tra i dispositivi sulla stessa rete locale. Si occupa della rilevazione e correzione degli errori, del controllo del flusso e dell'indirizzamento fisico (MAC).
3. **Livello di rete:** si occupa dell'instradamento dei pacchetti tra reti diverse. Utilizza indirizzi logici (come gli indirizzi IP) per identificare i dispositivi sulla rete e determina il percorso migliore per inviare i dati.
4. **Livello di trasporto:** garantisce la consegna affidabile dei dati tra i dispositivi. Si occupa della segmentazione dei dati, del controllo degli errori e del controllo del flusso. I protocolli TCP e UDP.
  - Multiplexing e demultiplexing
  - Indirizzamento delle unità dei dati (SAP)
  - Controllo di flusso end-to-end
  - Segmentazione e riassemblaggio delle unità dati
  - Controllo errori end-to-end
5. **Livello di sessione:** gestisce le sessioni di comunicazione tra i dispositivi. Si occupa dell'apertura, della chiusura e del mantenimento delle sessioni, nonché della sincronizzazione dei dati.
6. **Livello di presentazione:** si occupa della formattazione e della codifica dei dati. Converte i dati in un formato comprensibile per il livello applicativo e gestisce la compressione e la crittografia dei dati.
7. **Livello applicativo:** è il livello più alto del modello e fornisce agli utenti finali le interfacce per l'accesso al servizio. Esempio: posta elettronica, messaggistica istantanea.



Figure 2.1: I sette livelli del modello ISO-OSI

L'ISO-OSI serve a descrivere l'architettura dei protocolli, tramite il raggruppamento e la stratificazione. Vengono raggruppate tra loro le funzioni simili per logica o tecnologia.

Questi gruppi vengono organizzati in modo gerarchico, quindi a strati (stratificazione). Con ogni strato ci si può interfacciare tramite le interfacce.

Lo strato N è costituito da una o più entità e l'interazione tra due sistemi avviene tra strati di eguale livello e con entità pari.

Con il modello iso osi i "problemi" nella progettazione vengono divisi, rendendo la progettazione più modulare e più semplice, potendo specializzarsi su un livello specifico del modello. dividere un problema in più livelli è più conveniente.

**Entità e pacchetti(PDU, PCI, SDU)** Ogni livello ha un'entità, queste elaborano i pacchetti, detti PDU(protocol data unit); ogni PDU è composto da un header PCI(protocol control information) e un payload SDU(service data unit)

### 2.1.1 Segmentazione e riassemblaggio (incapsulamento e decapsulamento)

La segmentazione è il processo di suddivisione dei dati in pacchetti più piccoli per facilitarne la trasmissione attraverso la rete. Ogni pacchetto viene inviato separatamente e può seguire percorsi diversi attraverso la rete. Il riassemblaggio è il processo inverso, in cui i pacchetti ricevuti vengono ricomposti nell'ordine corretto per ricostruire i dati originali. (dall'alto verso il basso(segmentazione) vengono aggiunti gli header ai pacchetti; dal basso verso l'alto(riassemblaggio) vengono rimossi, quindi utilizzati, gli header dei pacchetti)

Dall'alto verso il basso: trasmissione → da tenere a mente che il livello più basso c'è il livello fisico, mentre il livello più alto è quello a cui tipicamente ci interfacciamo(applicazione del telefono ad esempio, whatsapp) quando trasmetto quindi faccio encapsulation, ossia aggiungo informazioni utili alla trasmissione del payload. questo perchè tra un livello è l'altro potrei avere la necessità di dover dividere il pacchetto in altri pacchetti, perciò va aggiunto negli header le informazioni necessarie a far riassemblare il pacchetto intero una volta finita la trasmissione.

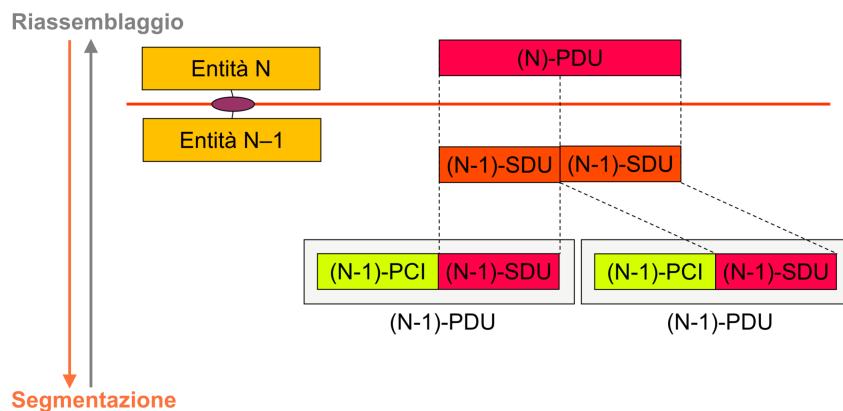


Figure 2.2: Pacchetti e incapsulamento/decapsulamento

## 2.2 Modello ibrido

Il modello ibrido è un modello di rete che combina elementi del modello ISO-OSI e del modello TCP/IP. È stato sviluppato per affrontare le limitazioni del modello ISO-OSI e per adattarsi meglio alle esigenze delle reti moderne. Il modello ibrido è composto da cinque livelli principali:

1. **Livello fisico:** simile al livello fisico del modello ISO-OSI, si occupa della trasmissione dei dati attraverso il mezzo fisico.
2. **Livello di collegamento dati:** gestisce la comunicazione tra i dispositivi sulla stessa rete locale e include funzioni di rilevamento degli errori e controllo del flusso.
3. **Livello di rete:** simile al livello di rete del modello ISO-OSI, si occupa dell'instradamento dei pacchetti tra reti diverse.
4. **Livello di trasporto:** fornisce servizi di trasporto affidabili e non affidabili, come TCP e UDP.
5. **Livello applicativo:** fornisce servizi di rete agli utenti finali e include protocolli come HTTP, FTP e SMTP.

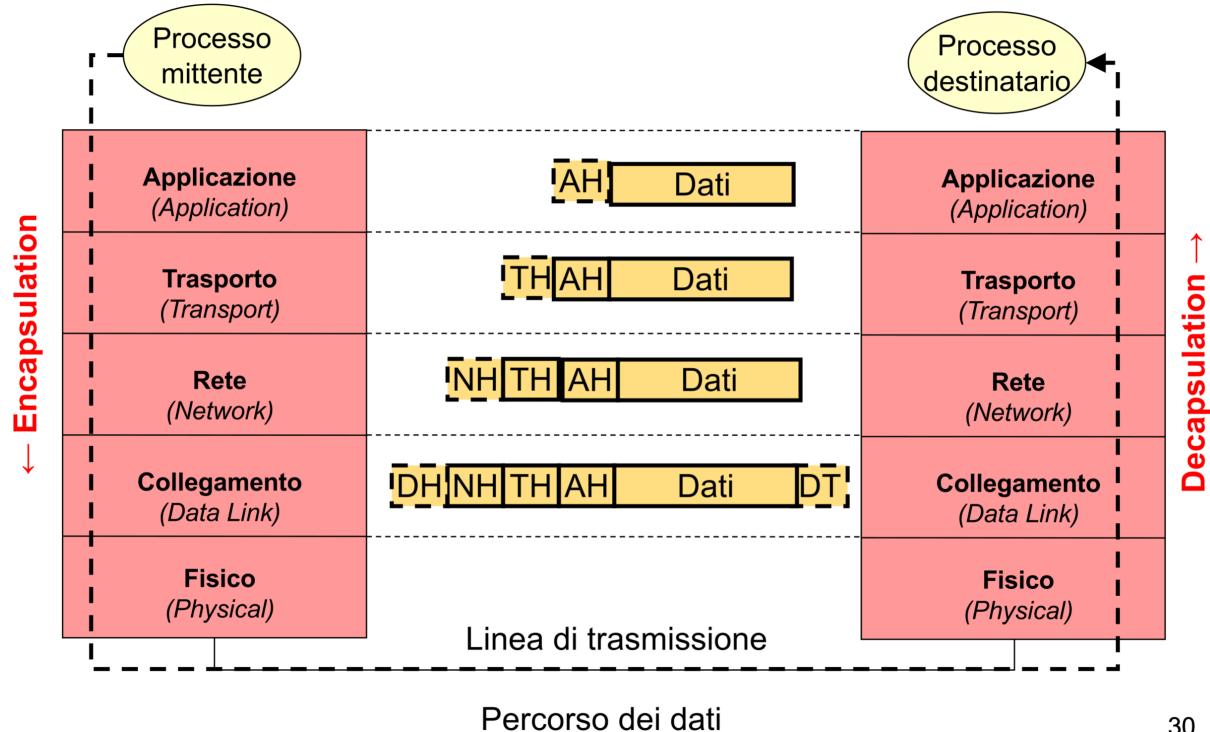


Figure 2.3: Incapsulamento e decapsulamento dei dati nel modello ibrido

### 2.2.1 Enc/decapsulation dati nel modello ibrido

Durante l'encapsulation vengono aggiunti a ogni livello header (AH = application header, TH = transport header, NH = network header ecc), durante la decapsulation vengono spaccati grazie agli header precedentemente aggiunti ad ogni livello l'unico livello che aggiunge oltre ad un header anche una coda (tail) è il "data link", "collegamento", questo perché vuole verificare la correttezza del pacchetto, tramite tecniche che vedrò avanti (algoritmi di controllo, non esiste un metodo che garantisce che il pacchetto sia 100% corretto).

## 2.2.2 Comunicazione tra sistemi

i livelli rete, data link e fisico si occupano del trasporto dell'informazione tra il sistema a ed il sistema b. questo può avvenire tramite il nodi, intermediari, x e y, oppure in modo diretto. il datalink presuppone un collegamento diretto ed è fondamentale che riceva informazion dal livello fisico poichè gestisce il passaggio di info tra un mezzo fisico e l'altro. al pacchetto viene aggiunto tramite il datalink un header che gestisce il passaggio tra a e x, quindi utilizzando un indirizzo locale. a livello di datalink inoltre viene viene verificato che il pacchetto sia inviato correttamente(evitando il loop: deadlog). dopo aver verificato la correttezza del pacchetto, a livello di rete ci si assicura che il pacchetto ricevuto sia arrivato a destinazione. nel caso specifico il pacchetto che arriva al livello di rete del nodo x viene spedito al nodo successivo poichè il livello di rete del nodo x capisce che non era lui il destinatario, tramite la lettura dell'header inserito dal sistema a inizialmente. tra un nodo e l'altro è possibile usare tecnologie differenti, cavo, wireless ecc, livello fisico

se al livello b arriva un pacchetto errato allora viene reinviato il pacchetto da b ad a così da far capire ad a che deve reinviare il pacchetto, latenza. posso inserire un timer oltre il quale si interrompe la trasmissione, per evitare loop o perdite di informazioni.

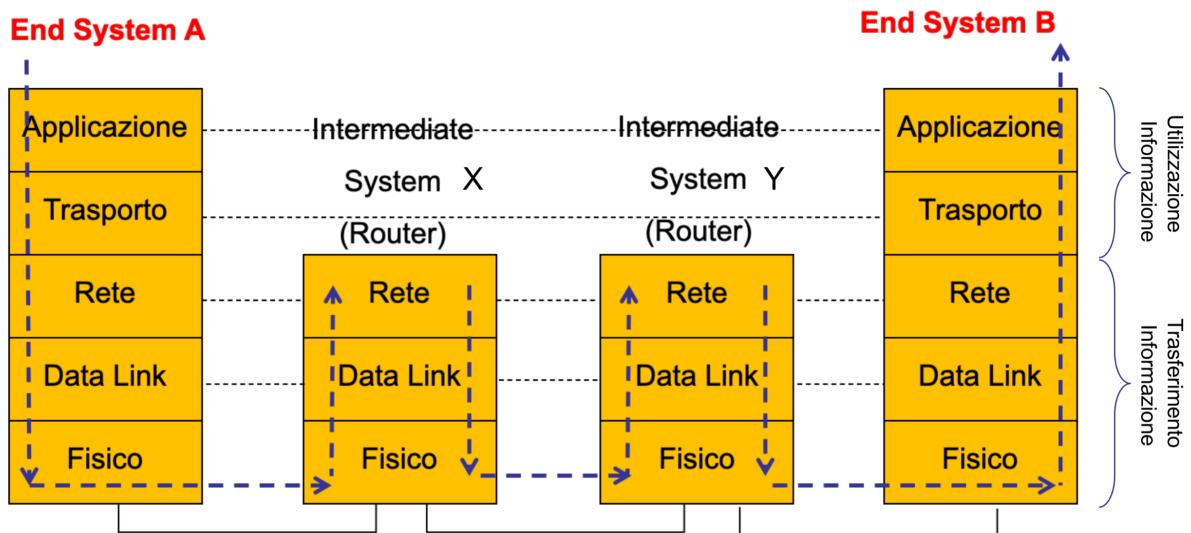


Figure 2.4: Comunicazione tra sistemi nel modello ibrido

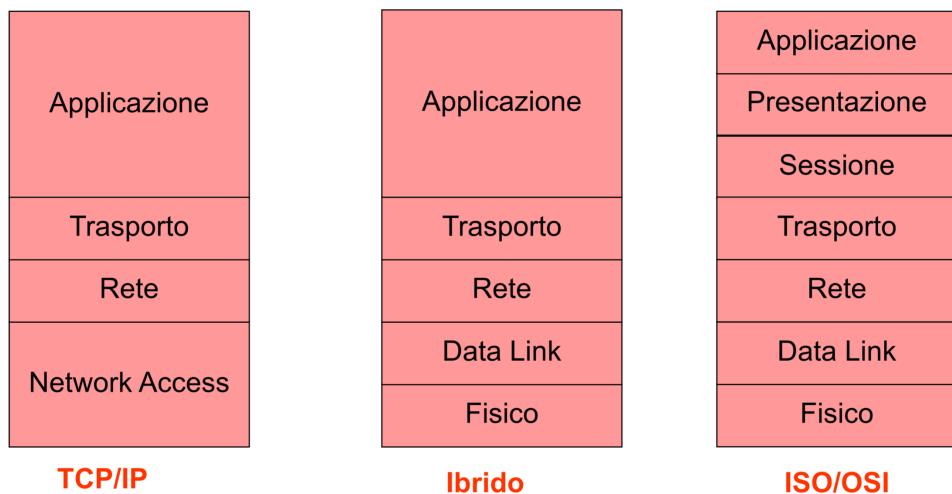


Figure 2.5: Confronto tra stack ISO-OSI, TCP/IP e modello ibrido

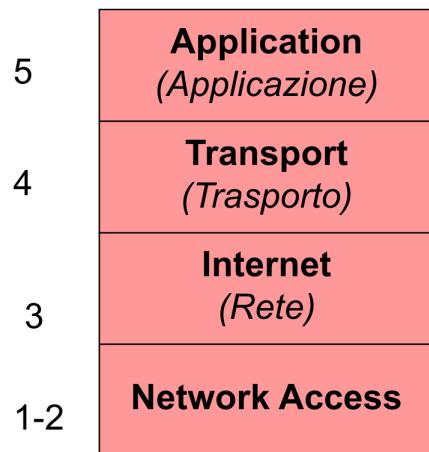
## 2.3 Modello TCP/IP

Il modello tcp/ip nasce a 4 livelli(negli anni 70), il livello di rete si chiama internet, poichè a quel livello si usa il protocollo ip(internet protocol).

Il suo nome è dovuto dai due protocolli principali che lo compongono: TCP (Transmission Control Protocol) e IP (Internet Protocol).

I livelli del modello TCP/IP:

1. **Livello di accesso alla rete:** corrisponde ai livelli DataLink e Fisico del modello ISO-OSI.
2. **Livello Internet:** si occupa dell'indirizzamento dei pacchetti tra reti diverse. Utilizza indirizzi logici (come gli indirizzi IP) per identificare i dispositivi sulla rete e determina il percorso migliore per inviare i dati. A consentire lo scambio di informazioni tra nodi della rete è l'ICMP, l'Internet Control Message Protocol. Al livello di rete sono presenti anche protocolli di routing. Importante: non prevede l'interlavoro tra protocolli di rete diversi, ma necessariamente tutti i nodi devono utilizzare IP.
3. **Livello di trasporto:** fornisce servizi di trasporto affidabili (TCP) e non affidabili (UDP), come TCP e UDP. Si occupa della segmentazione dei dati, del controllo degli errori e del controllo del flusso.
4. **Livello applicativo:** Come ISO OSI e ibrido



**Livelli TCP/IP**

Figure 2.6: I livelli del modello TCP/IP

### Confronto tra i modelli

I tre modelli — ISO-OSI, TCP/IP e il modello ibrido — sono utilizzati per descrivere l'architettura delle reti di telecomunicazione, ma presentano differenze significative in termini di struttura, approccio e utilizzo pratico.

**Modello ISO-OSI:** è un modello teorico a 7 livelli, sviluppato come riferimento universale per la progettazione delle reti. Ogni livello ha funzioni ben definite e interagisce solo con i livelli adiacenti. Il modello ISO-OSI è molto dettagliato e didattico, ma non è stato adottato integralmente nelle implementazioni pratiche.

**Modello TCP/IP:** nasce dall'esperienza pratica di Internet e si compone di 4 livelli. È meno dettagliato rispetto all'ISO-OSI, ma più semplice e orientato all'implementazione. Il modello TCP/IP è diventato lo standard de facto per le reti moderne.

**Modello ibrido:** rappresenta un compromesso tra i due modelli precedenti, combinando la chiarezza concettuale dell'ISO-OSI con la semplicità e la praticità del TCP/IP. Il modello ibrido solitamente prevede 5 livelli, raggruppando alcune funzioni e adattandosi meglio alle esigenze delle reti attuali.

In sintesi, il modello ISO-OSI è utile per comprendere i principi teorici delle reti, il modello TCP/IP è quello realmente utilizzato nelle reti Internet, mentre il modello ibrido cerca di unire i vantaggi di entrambi per una maggiore flessibilità e chiarezza.

# Chapter 3

## Livello 4 trasporto TCP/UDP

### 3.1 Nozioni preliminari

UDP(Used Datagram Protocol) e TCP(Transmission Control Protocol) sono i protocolli di trasporto(livello 4) più diffusi. Principali differenze:

- TCP è orientato alla connessione, UDP no (vuol dire che non è necessario stabilire una connessione prima di inviare i dati, cosa invece necessaria con TCP)
- TCP è un protocollo di trasporto affidabile, UDP è non affidabile(quindi TCP garantisce che i dati arrivino a destinazione e nell'ordine corretto, mentre UDP non lo fa)

I pacchetti(PDU) TCP sono chiamati segmenti, mentre i pacchetti UDP sono chiamati datagrammi.

**Funzioni principali UDP** Svolge l'unico compito di incapsulare i dati dell'applicazione in un pacchetto UDP, aggiungendo le informazioni necessarie per la consegna(Multiplexing). Non fornisce alcuna garanzia di consegna o ordine dei pacchetti.

#### Funzioni principali TCP

- Controllo flusso end to end
- Controllo congestione end to end
- Ritrasmette le PDU perse o corrotte
- Riordina i segmenti ricevuti in ordine corretto

Il TCP numera i singoli byte che arrivano dal livello applicativo, e non i segmenti; è fondamentale poichè dal livello 3(IP) non arrivano i byte in ordine, ma i pacchetti possono arrivare in ordine sparso.

#### 3.1.1 Sock e ports

Dei concetti fondamentali per il funzionamento di TCP e UDP sono i socket e le porte.

- Socket: è un punto finale di una connessione di rete, rappresentato da:
  - Indirizzo IP: identifica un host sulla rete
  - Porta: identifica un'applicazione in esecuzione su quell'host

I socket sono utilizzati per inviare e ricevere dati tra applicazioni su host diversi.

- Porta: è un numero che identifica un'applicazione in esecuzione su un host. Le porte sono numerate da 0 a 65535 e sono suddivise in tre categorie: well-known ports, registered ports e dynamic/private ports.

Il numero di porta è un numero a 16 bit che identifica un'applicazione in esecuzione su un host. Quando un client invia una richiesta HTTP a un server web, utilizza la porta 80 per comunicare con il server. Il server ascolta sulla porta 80 e risponde alla richiesta del client.

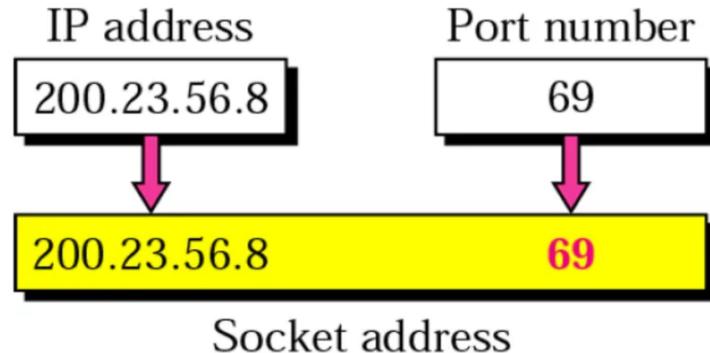


Figure 3.1: Relazione tra socket, indirizzo IP e porta

Le porte sono numerate da 0 a 65535 e sono suddivise in tre categorie:

- Well-known ports: da 0 a 1023, utilizzate da applicazioni di sistema e protocolli standard (es. HTTP su porta 80, HTTPS su porta 443, DNS(traduce il nome "simbolico" del servizio web per il web server) su porta 53)
- Registered ports: da 1024 a 49151, utilizzate da applicazioni registrate presso l'IANA
- Dynamic/Private ports: da 49152 a 65535, utilizzate per connessioni temporanee e dinamiche

Tramite i numeri di porta è possibile identificare a livello 4 con quale applicazione sto inviando/ricevendo dati

Come identificare un flusso dati in internet?

- Numero di porta locale
  - Indirizzo IP host mittente
  - Numero di porta remota
  - Indirizzo IP host destinatario
  - Tipologia di Protocollo di Trasporto
- } **IP flow**

Figure 3.2: Identificazione di un flusso dati tramite indirizzi IP e numeri di porta (tuple a 4 campi)

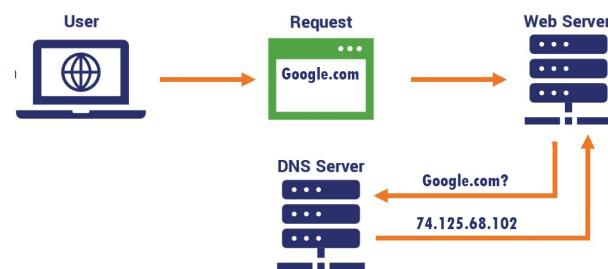


Figure 3.3: Esempio di funzionamento del DNS: risoluzione di un nome simbolico in indirizzo IP, tramite DNS server

traduce nomi comprensibili come `www.amazon.com` in indirizzi IP utilizzati dai computer, ad esempio `192.0.2.44`

## 3.2 UDP - datagram

è il più veloce del TCP poichè non esegue molte funzioni come l'ordinamento dei pacchetti i pacchetti sono da 32 bit, ossia 4 byte:

- datagram: 16 bit
- pseudoheader: 16 bit

Si occupa di fare multiplexing(trasmissione) e demultiplexing(ricezione) dei dati, ovvero incapsula i dati dell'applicazione in un pacchetto UDP e aggiunge le informazioni necessarie per la consegna. Viene usato soprattutto per applicazioni in tempo reale, come servizi streaming live.

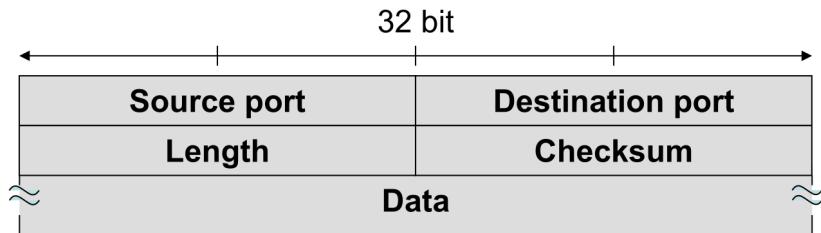


Figure 3.4: Struttura del datagram UDP

### Struttura del datagram UDP

- Source port e destination port(indirizzo IP del mittente/destinatario): 16 bit ciascuno
- Length (dimensione del datagram): 16 bit, quindi può rappresentare valori da 0 a  $2^{16} - 1 = 65535$  bytes, ma il campo include anche l'header, quindi la dimensione massima di un datagram UDP è 65535 byte.
- Checksum: 16 bit, utilizzato per verificare l'integrità dei dati del datagram. Il checksum è calcolato su un pseudoheader che include informazioni sull'indirizzo IP del mittente e del destinatario, oltre alla lunghezza del datagram e al protocollo utilizzato (UDP in questo caso). Il pseudoheader non viene trasmesso, ma viene utilizzato solo per il calcolo del checksum.
- Data: dimensione variabile, contiene i dati dell'applicazione

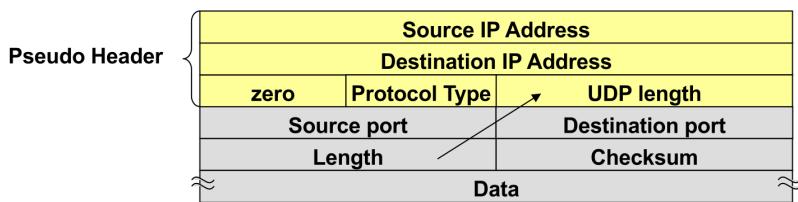


Figure 3.5: Pseudoheader UDP utilizzato per il calcolo del checksum

Nello pseudoheader sono presenti informazioni come il campo Protocol Type viene preso direttamente dal livello 3(protocollo IP).

Il checksum è la somma di controllo, è una tecnica di verifica veloce di controllo errore, non è affidabile al 100%, è comunque ottimo perchè è veloce

- Se il checksum è 0, significa che non ci sono errori, non affidabile.
- Se il checksum è diverso da 0, significa che ci sono errori, sicuramente(condizione necessaria).

### 3.2.1 Calcolo checksum

Il valore del checksum prima della trasmissione è impostato a 0.

**In trasmissione** In trasmissione vengono sommate in binario tutte le righe(mezze righe da 16 bit) del datagram, faccio il complemento ad 1(NOT) di questo risultato. Questo è il checksum inviato nel datagram di trasmissione.

**In ricezione** In ricezione viene ricevuto il datagram, con all'interno il checksum calcolato precedentemente in trasmissione.

Viene calcolato nuovamente il checksum del datagram ricevuto, sommando in binario tutte le righe (mezze righe da 16 bit) del datagram, e sommandolo al checksum ricevuto.

Se il risultato è diverso da 0 significa che i due valori calcolati in ricezione e trasmissione sono differenti, perciò c'è un errore.

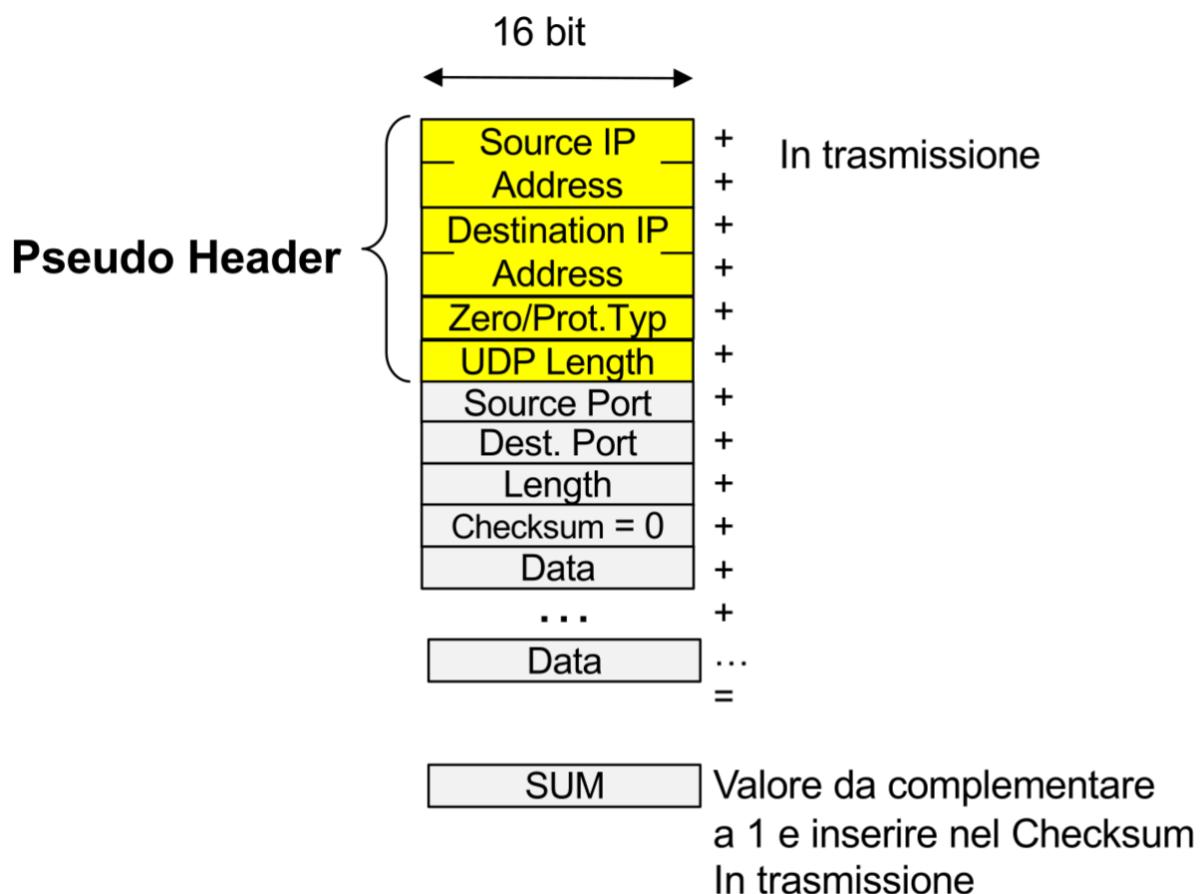


Figure 3.6: Esempio di calcolo del checksum UDP

### 3.3 TCP - segmento

Nei processi TCP viene utilizzata una coppia di socket per identificare un flusso di dati. Ogni socket è identificato da una tupla a 4 campi:

- Indirizzo IP del mittente
- Porta del mittente
- Indirizzo IP del destinatario
- Porta del destinatario

La connessione è full duplex → comunicazione affidabile, se qualcosa va storto chiede la ritrasmissione del pacchetto errato.

Il tcp è un protocollo greedy, ingordo, tende a prendere tutta la banda di cui necessita; questo protocollo non evita le congestioni ma le provoca, però quando le provoca cerca di risolverle.

L'affidabilità di TCP sta nei riscontri cumulativi che gli permettono di gestire le seguenti problematiche:

- Controllo di congestione e2e: regolazione del rate di trasmissione così da utilizzare completamente la banda disponibile evitando che la rete collassi
- Controllo del flusso e2e: regolarizza il rate di trasmissione per evitare la saturazione del buffer di ricezione

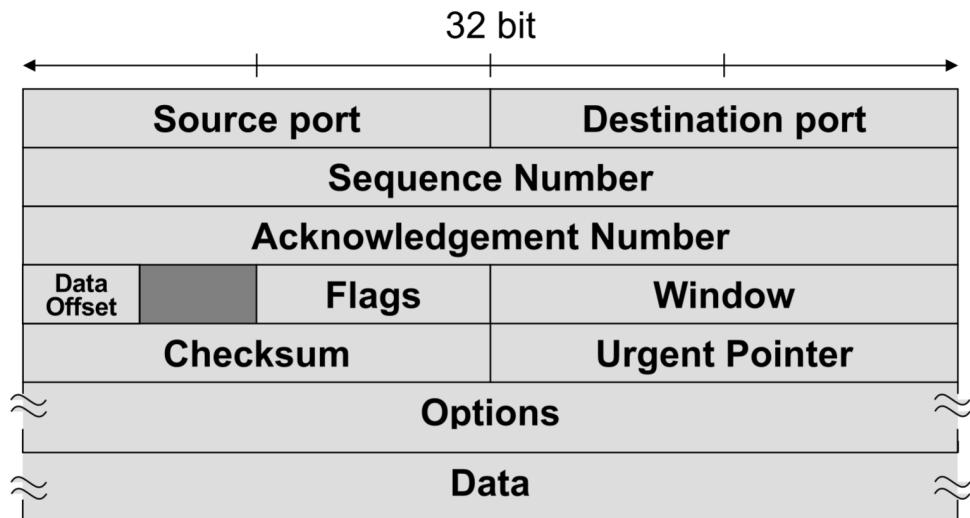


Figure 3.7: Struttura del segmento TCP

I pacchetti in tcp vengono chiamati segmenti, così strutturati:

- source port/destination port: 16 bit ciascuna e identificano il mittente e il destinatario del segmento
- sequence number: 32 bit, numero di sequenza del primo byte del segmento, utilizzato per ordinare i segmenti ricevuti; i byte del livello applicativo (che costituiranno il payload dei diversi segmenti trasmessi) sono numerati in modo continuo partendo da un valore casuale (zero relativo).

Nel campo Seq. Number si inserisce poi il numero di sequenza (di questa numerazione continua) relativo al primo byte contenuto nel campo Data

- acknowledgement number: 32 bit, nel campo ack.number si inserisce il numero di sequenza del primo byte che ci si aspetta di ricevere (quindi il numero di sequenza del primo byte che non è stato ricevuto), si usa per capire fino a che punto il segmento è stato ricevuto correttamente. Così

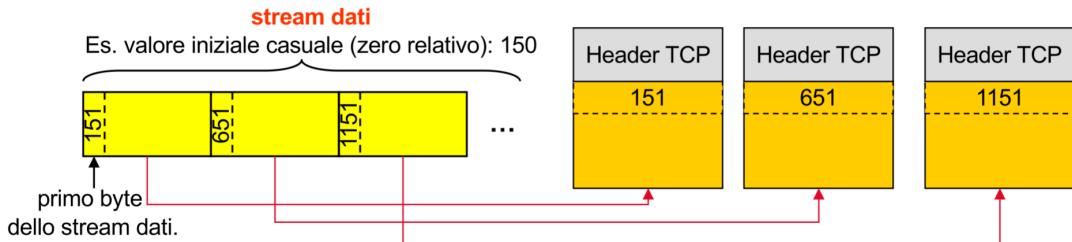


Figure 3.8: Esempio di utilizzo del campo Sequence Number nei segmenti TCP

facendo sì anche da che punto riprendere la trasmissione dei dati.(è utile a dare un riscontro di ciò che mi sta inviando il mittente, tenendolo aggiornato sulla correttezza di ciò che mi ha inviato, connessione full duplex appunto); può essere anche un campo vuoto, 0.

- data offset: 4 bit, indica la lunghezza dell'header del TCP
- flags: 6 bit, sono dei flag di controllo che indicano lo stato della connessione TCP:
  - URG: indica se il segmento contiene dati urgenti.
  - ACK: acknowledgment number
  - PSH: push, richiesta di invio immediato dei dati al livello applicativo
  - RST, SYN, FIN: reset, synchronize, finish della connessione
- window: è un campo utilizzato nella gestione del controllo di flusso, indica la dimensione della finestra di ricezione, ovvero la quantità di dati che il mittente può inviare prima di ricevere un acknowledgment dal destinatario. La dimensione della finestra può variare durante la connessione in base alla disponibilità di buffer del destinatario.
- urgent pointer: indica i dati che il ricevitore deve elaborare per prima
- checksum: è come in UDP
- options: almeno 32 bit, è opzionale, può contenere info di configurazione del segmento TCP

### 3.3.1 Apertura connessione client-server (3way handshake)

La connessione viene sempre aperta dal client.

Quando ricevo un segmento ho bisogno di sapere in che modo sta contando i byte il mittente;

Il client sceglie lo zero relativo, ossia il numero di sequenza(sequence number scelto casualmente,  $SEQ.NUMBER = x$ ) da cui partire per numerare i byte che invierà al server, inviando un segmento vuoto in cui setta il flag  $SYN = 1$ , ( $ACK = 0$  all'inizio della comunicazione) così da iniziare la comunicazione.

A questo punto il server risponde alla richiesta del client, inviando:

- un segmento vuoto con  $SYN = 1$  e  $ACK = 1$ (questo flag a 1 fa capire al client che il campo ACKNUMBER è stato modificato), in cui il numero di sequenza è  $SEQ.NUMBER = y$ (per i dati che il server invierà, non dimenticare che la comunicazione è full duplex) e il numero di ack è  $ACK.NUMBER = x + 1$  (il server ha ricevuto la richiesta del client, quindi il numero di ack è incrementato di 1)
- un segmento con i dati richiesti dal client, in cui il numero di sequenza è  $SEQ.NUMBER = y + 1$  e il numero di ack è  $ACK.NUMBER = x + 1$

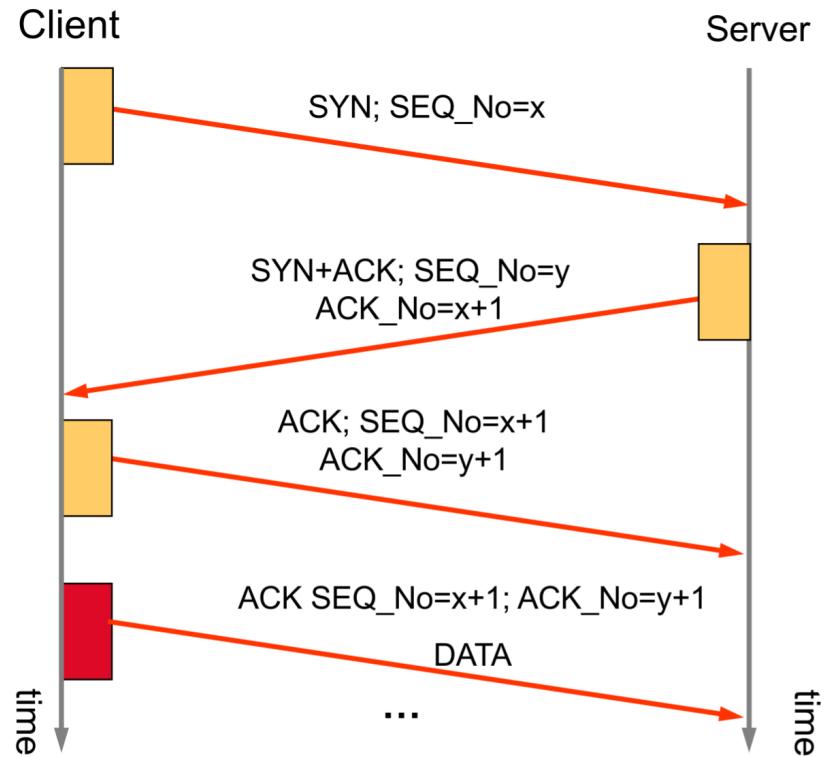


Figure 3.9: Three-way handshake: stabilimento della connessione TCP tra client e server

Per terminare la "conversazione" il client invia un segmento(vuoto, ossia senza dati) con il flag SYN a 0, e conferma la corretta ricezione del messaggio precedente (y) con l'ACK.NUMBER ricevuto dal server più di 1(y + 1).

Quello in rosso nell'immagine è un ulteriore segmento, uguale al precedente, ma " pieno", ossia con i dati da inviare.

### 3.3.2 Chiusura connessione

La chiusura della connessione può essere avviata da uno dei due host, il client o il server (tipicamente la chiude il client).

**Chiusura con half-close** La chiusura della connessione avviene tramite due mezze chiusure:

- Il client invia un segmento con il flag FIN a 1, per indicare che non invierà più dati al server.(I mezza chiusura)
- Il server continua ad inviare i dati finchè non termina e invia un segmento con flag FIN settato.(II mezza chiusura)
- Il server invia un segmento con il flag FIN a 1 e il flag ACK a 1, per indicare che non invierà più dati al client.
- Il client risponde con un segmento con il flag ACK a 1, per confermare la ricezione della richiesta di chiusura del server.

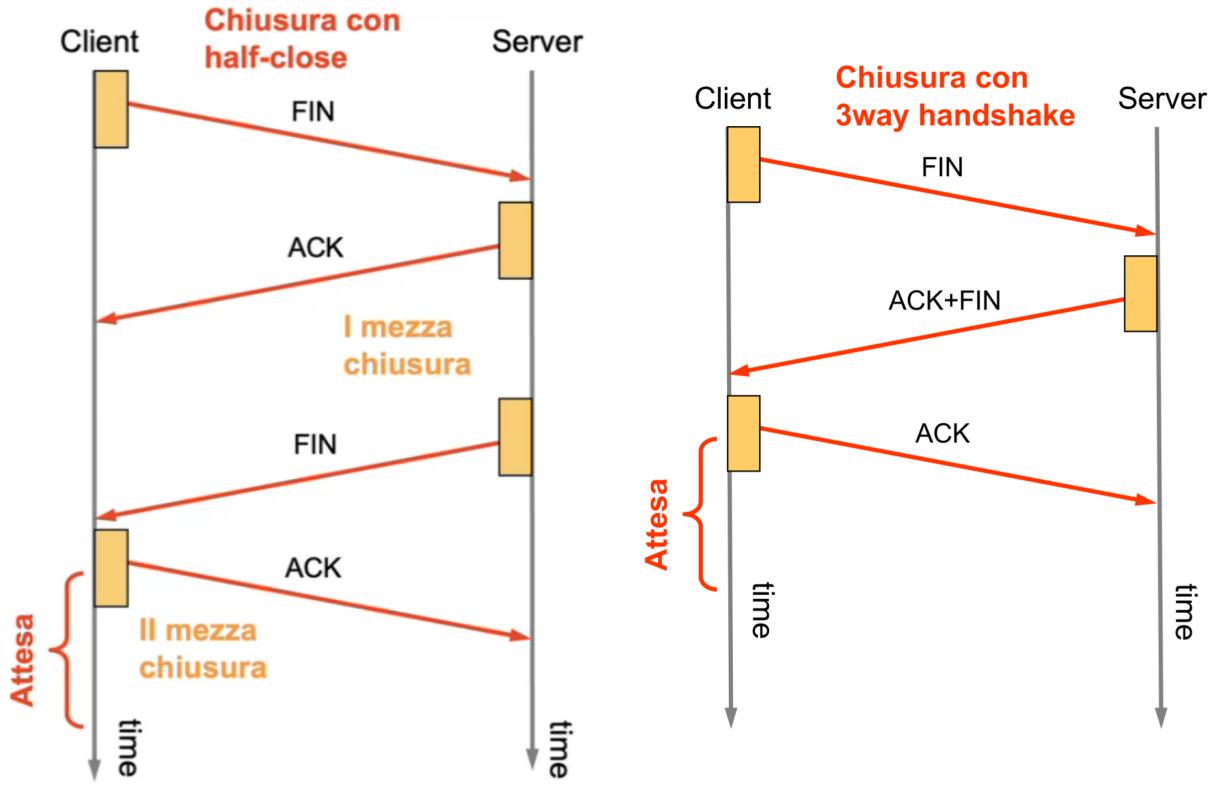


Figure 3.10: Chiusura della connessione TCP tramite half-close

Figure 3.11: Chiusura della connessione TCP tramite 3-way handshake

**Chiusura con 3way handshake** in risposta al segmento FIN inviato dal client, il server invia un unico segmento con FIN+ACK settati (e con eventualmente gli ultimi dati).

Questa soluzione non prevede che il server invii ulteriori dati e il client risponde con il segmento di ACK di chiusura della connessione.

**TIMEWAIT** in entrambe le tecniche (con half-close o 3way handshake) dopo il segmento di ACK si avvia un timer (Time Wait) prima della definitiva chiusura della connessione. Può succedere, infatti, che il segmento ACK vada perso e il server ritrasmetta il suo FIN. Il client deve quindi essere ancora in grado di poter ritrasmettere l'ACK finale.

### 3.3.3 TCP sequenze di stati

TCP client

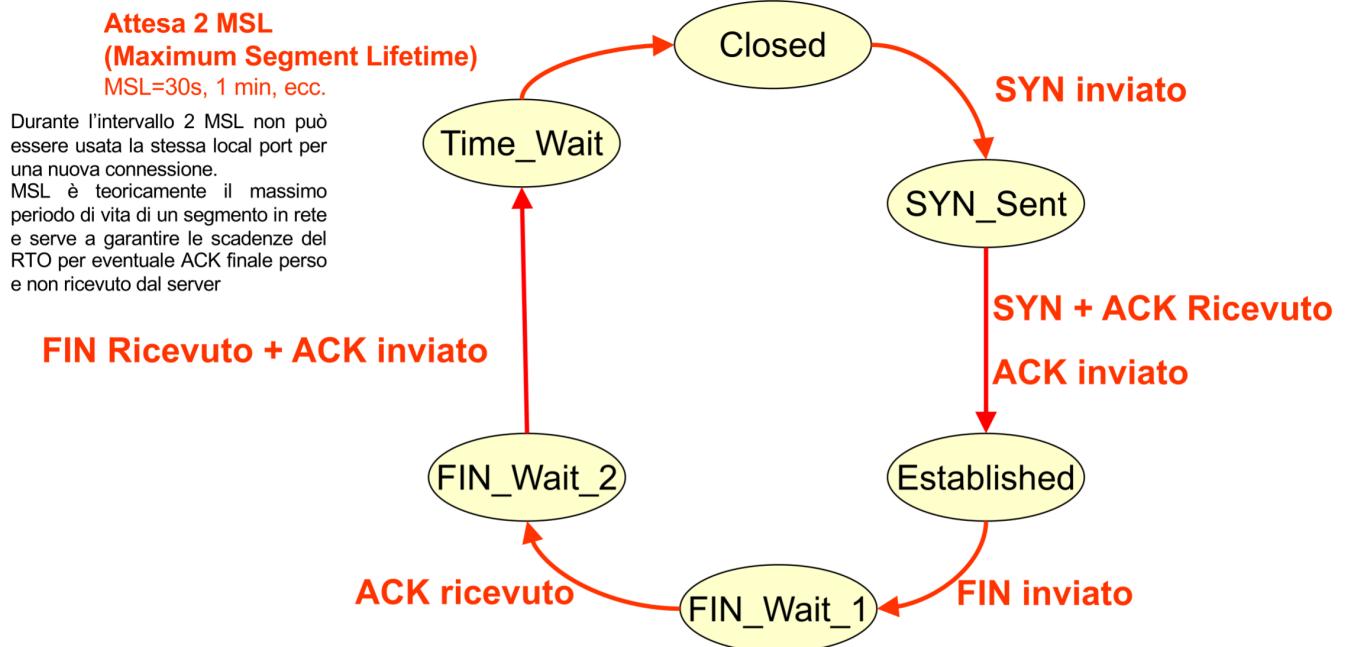


Figure 3.12: Sequenza di stati TCP lato client

TCP server

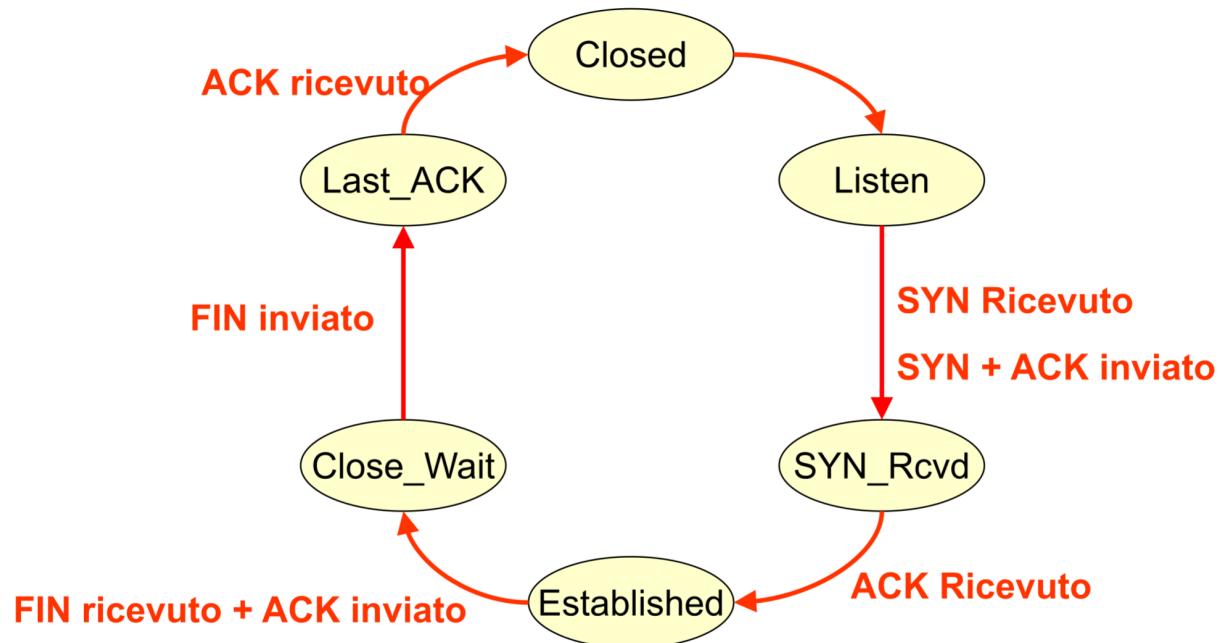


Figure 3.13: Sequenza di stati TCP lato server

### 3.3.4 Trasmissione dei segmenti - sliding window

Il TCP usa un meccanismo di trasmissione dei segmenti basato su finestre scorrevoli, finestra di trasmissione e finestra di ricezione; in base alla loro larghezza si possono inviare più o meno bytes.

Attua "self-clocking", ossia il mittente regola la velocità di invio dei segmenti in base alla velocità di ricezione del destinatario. Non c'è una velocità fissa delle sliding windows.

#### Round trip time - RTT

Quando invia un segmento, il mittente deve attendere un certo tempo prima di ricevere un (riscontro)acknowledgment dal destinatario. Questo tempo è chiamato round trip time (RTT) e rappresenta il tempo necessario per inviare un segmento e ricevere l'ACK.

Data  $W$ , dimensione della finestra di trasmissione, e RTT, il tempo di round trip, la velocità media di trasmissione dei segmenti è data da:

$$\text{rate medio} = \frac{W}{RTT} \quad (3.1)$$

Come migliora questo rate?

Si può agire solo sulla dimensione della finestra  $W$ , non sul RTT.

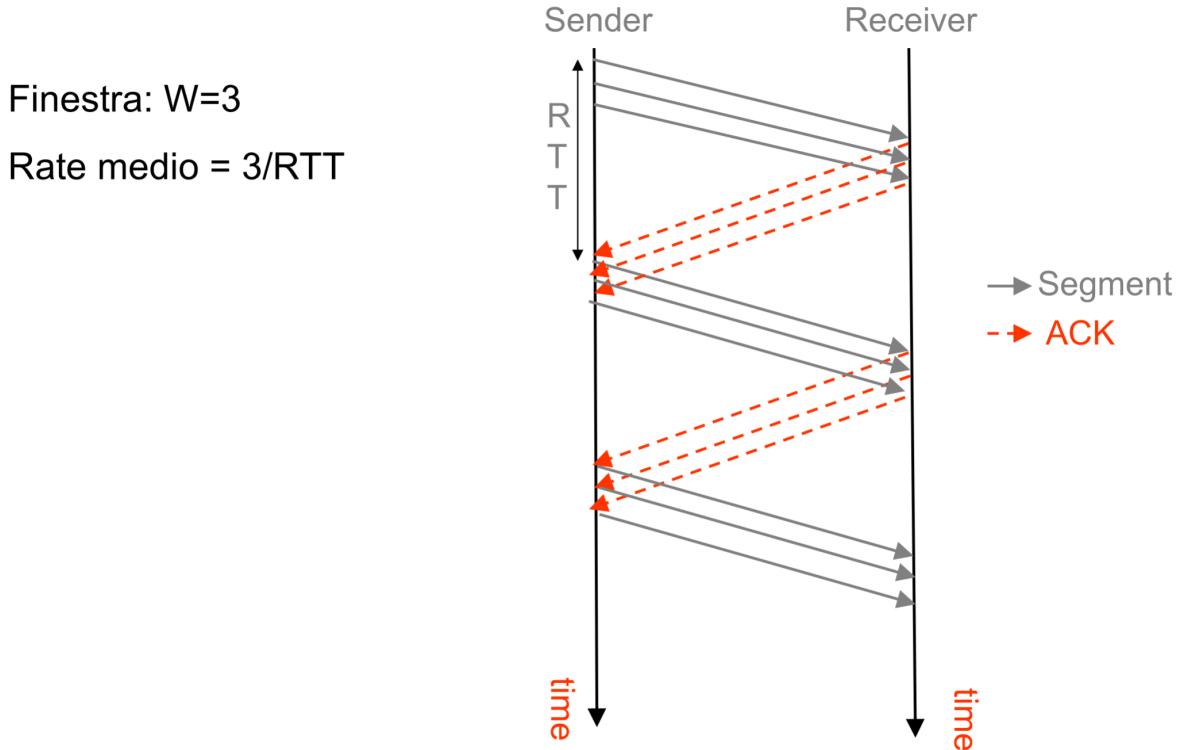


Figure 3.14: Esempio di sliding window e round trip time (RTT)

### 3.3.5 Finestre di trasmissione e ricezione

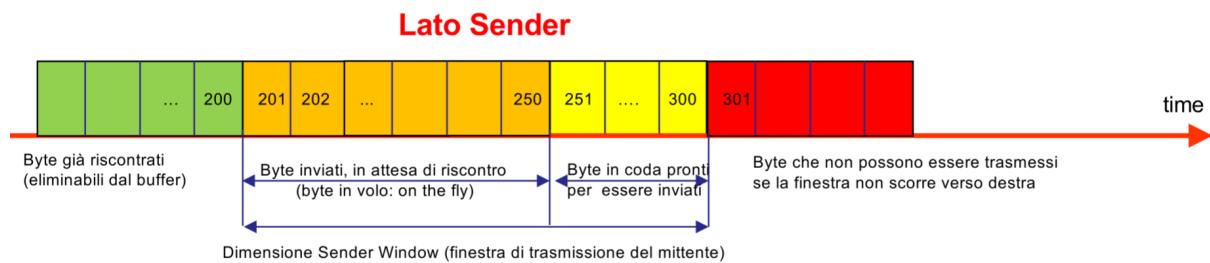


Figure 3.15: Esempio di finestra di trasmissione

Sono gli ACK ricevuti che fanno scorrere la finestra. Se ricevo l'ACK 240 allora la finestra scorre fino a 240 (riscontri cumulativi, non devo ricevere quelli 201, 202 ecc..., ma mi basta 240 per ricevere anche tutti i byte precedenti a 240).

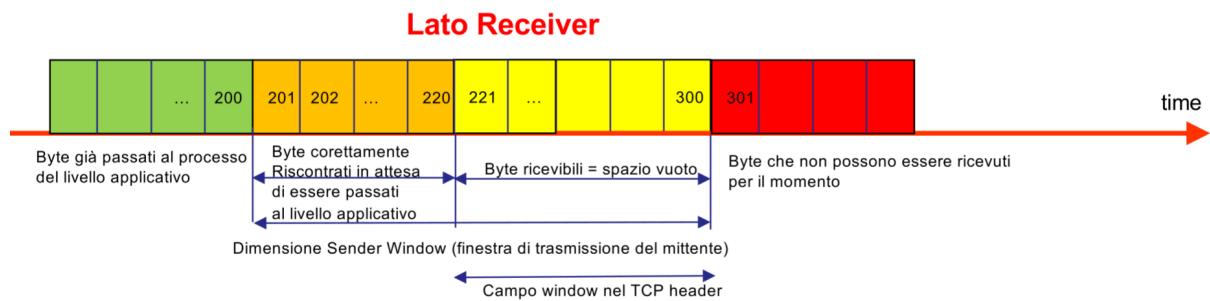


Figure 3.16: Esempio di finestra di ricezione

### 3.3.6 Controllo di flusso - come gestisce il TCP il flusso dei dati?

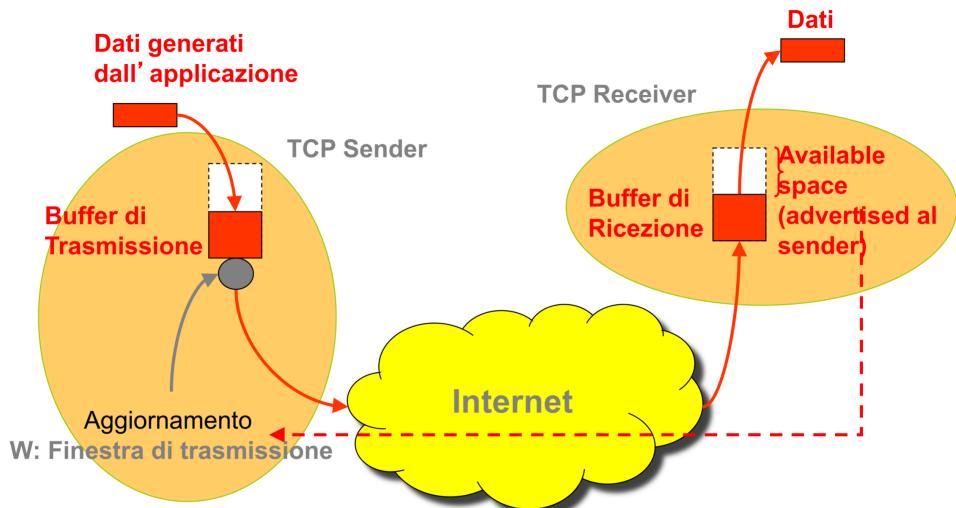


Figure 3.17: Esempio di flusso TCP

IMPORTANTE: ciò che è presente nel buffer di ricezione è stato riscontrato dal mittente.

Il receiver comunica con il sender la dimensione del buffer di ricezione, in modo che il mittente possa regolare la velocità di invio dei segmenti, quindi regolare la finestra di trasmissione.

### 3.3.7 Perdita di segmenti - RTO e 3 dupack

Si assume che un segmento sia stato perso se si verificano: 3 dupack, RTO.

#### Retransmission Time Out - RTO

In questo caso, a differenza del 3 dupack, nel quale si perdeva un solo segmento, nell'esempio seguente non vengono riconosciuti ben 3 segmenti.

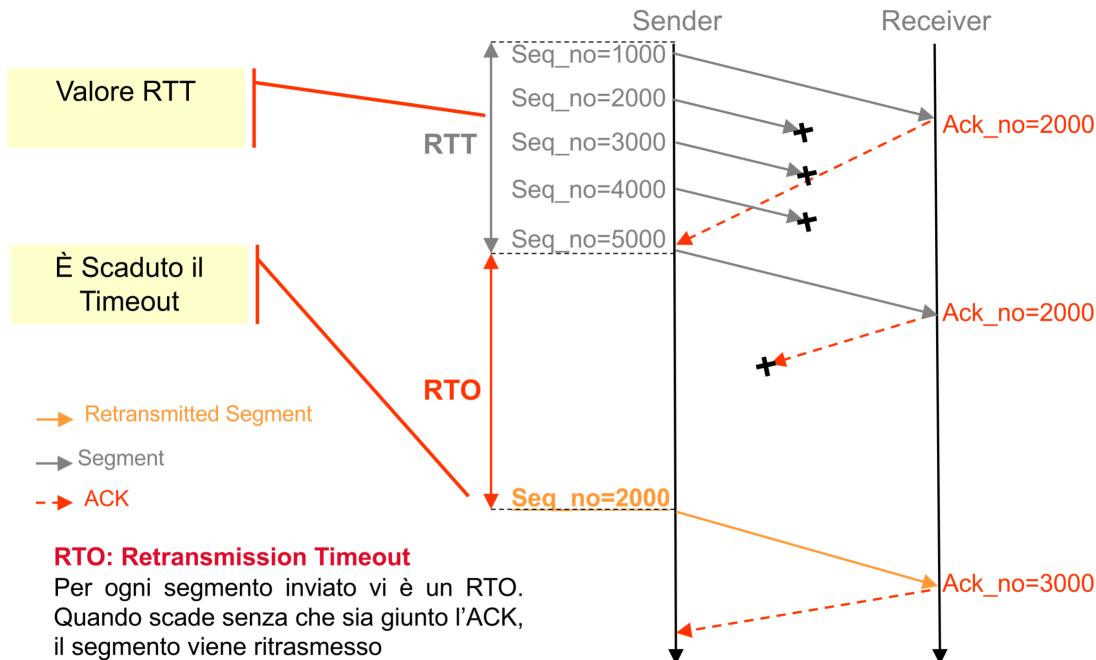


Figure 3.18: Esempio di Retransmission Time Out (RTO) in TCP

Per ogni segmento inviato, viene avviato un RTO, ossia un timer che scade dopo un certo intervallo di tempo. Se il timer scade prima di ricevere l'ACK, il mittente ritrasmette il segmento.

**Calcolo del RTO** ?

### 3 Dupack(three duplicate ACK)

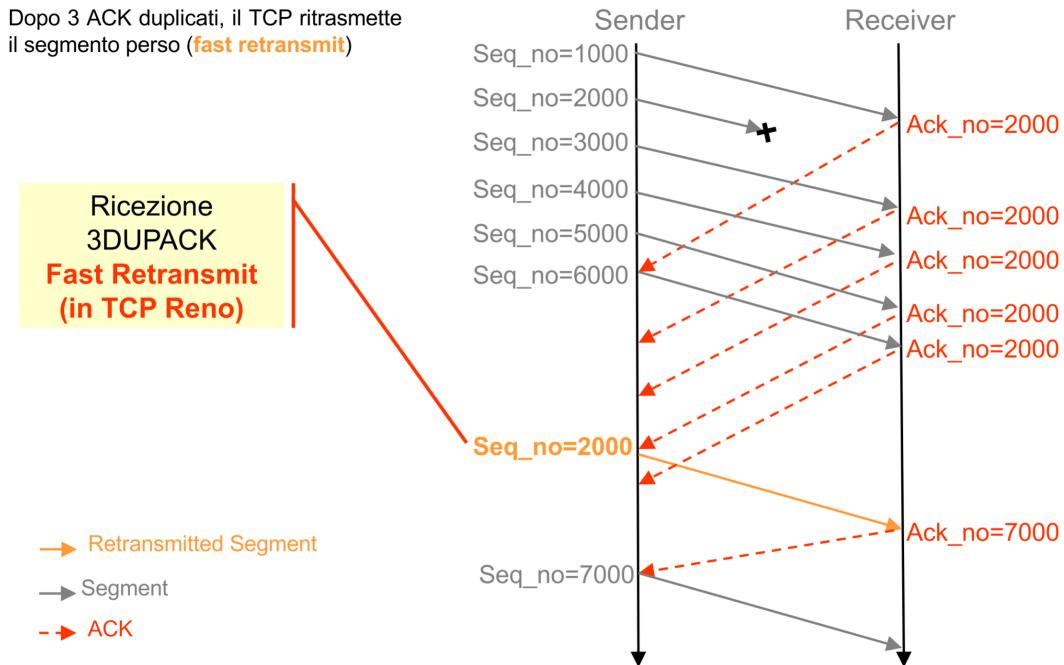


Figure 3.19: Esempio di 3 duplicate ACK (3 Dupack) in TCP

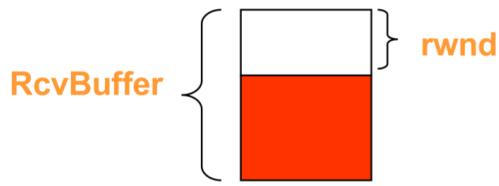
#### 3.3.8 Controllo di congestione TCP

I problemi di congestione si verificano quando la rete è sovraccarica(buff di ricezione saturo) di traffico e non riesce a gestire tutte le richieste. TCP utilizza diversi algoritmi per gestire la congestione e garantire una trasmissione affidabile dei dati.

Il protocollo TCP si rende conto dei limiti della rete solo quando avviene una congestione, quindi il TCP testa i limiti della rete(probing).

A tal scopo è importante definire le seguenti variabili:

- rwnd(ricezione): è la dimensione della finestra di ricezione, ossia la quantità di dati che il mittente può inviare prima di ricevere un ACK dal destinatario.
- cwnd(finestra di congestione): è dinamicamente calcolata tramite un algoritmo di controllo di congestione.



La finestra di congestione W è posta uguale al minimo tra la congestion window e la rwnd:

$$W = \min(cwnd, rwnd) \quad (3.2)$$

Figure 3.20: Esempio di finestra di ricezione (rwnd) in TCP

La variabile cwnd ha il vantaggio di poter essere controllata, perciò se c'è una congestione in atto posso diminuire la variabile cwnd, così da evitare di saturare la rete.

## Algoritmo di controllo di congestione

**Additive Increase Multiplicative Decrease (AIMD)** TCP aumenta la finestra di ricezione rallentando "addittivamente" la velocità di invio dei segmenti (inizialmente è molto veloce, piano piano questa velocità tende a diminuire in modo additivo).

Quando però avviene una congestione, la finestra non viene rallentata in modo additivo, ma anzi, viene ridotta drasticamente in modo moltiplicativo, moltiplicando la finestra per un fattore frazionario.

### Evoluzione della cwnd

**Slow start** Inizialmente la cwnd è pari ad 1 MSS, minimo valore possibile di cwnd, invia un segmento e inizia la fase di probing (slow start: parte "piano" per poi aumentare). Ogni volta che si ha un riscontro viene raddoppiata la cwnd.

- **MSS(maximum segment size):** rappresenta la massima quantità di dati (in byte) che un host TCP può ricevere in un singolo segmento. tipicamente 1460 byte (se MTU = 1500 e header IP+TCP = 40 byte)
- **ssthresh(slow start threshold):** è una soglia che separa la fase di Slow Start da quella di Congestion Avoidance

Quando si verifica una congestione (3DUPACK o scadenza RTO), la cwnd e la soglia sshtresh (viene posta pari alla metà della finestra di congestione al momento della congestione) vengono modificate in base all'algoritmo di congestione utilizzato.

**Congestion avoidance** Nel caso in cui la cwnd sia maggiore della soglia sshtresh, inizia la fase di congestion avoidance, in cui la cwnd viene aumentata linearmente, ossia viene incrementata di 1 MSS ogni RTT.

L'obiettivo è quello di aumentare la cwnd in modo più lento, evitando di saturare la rete.

$$cwnd = cwnd + \frac{1}{cwnd} \quad (3.3)$$

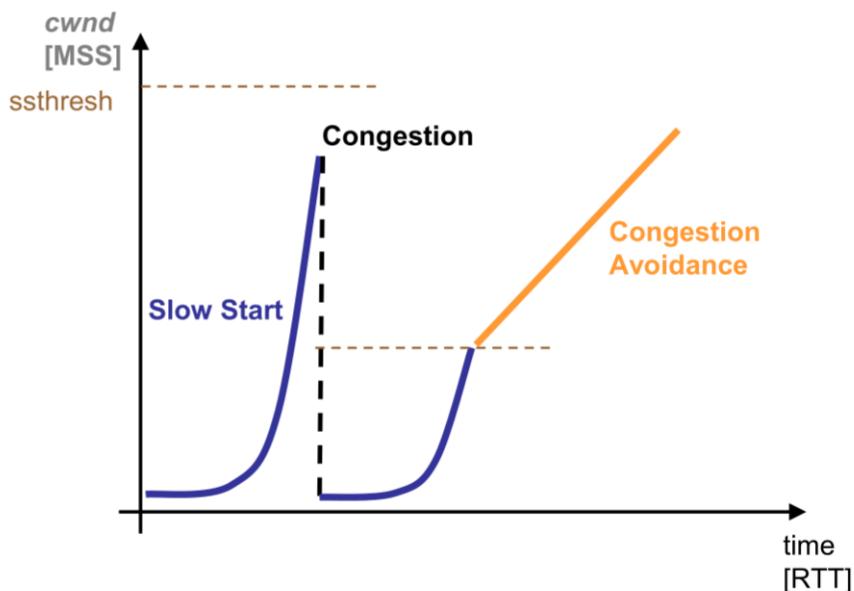
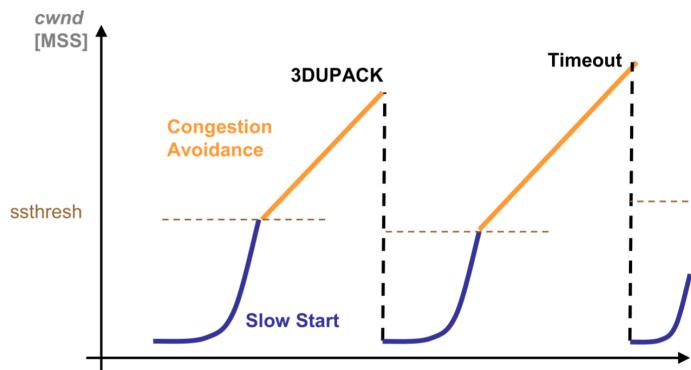


Figure 3.21: Andamento della finestra di congestione

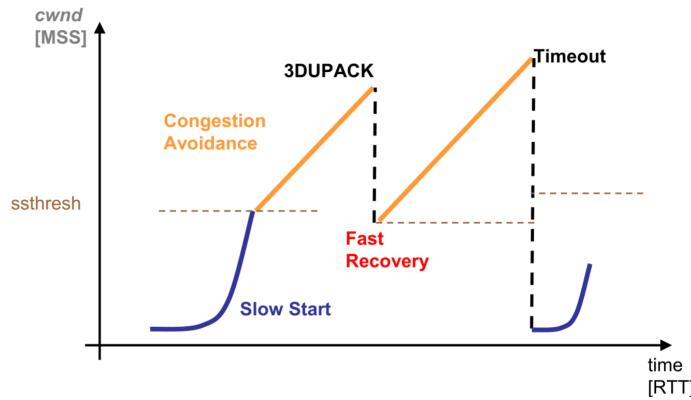
## TCP Tahoe



TCP Tahoe non distingue tra i due eventi di congestione(3DUPACK e RTO), quando avviene riporta la cwnd a 1; rientrando in slow start.

Figure 3.22: Andamento della finestra di congestione in Tahoe

## TCP Reno

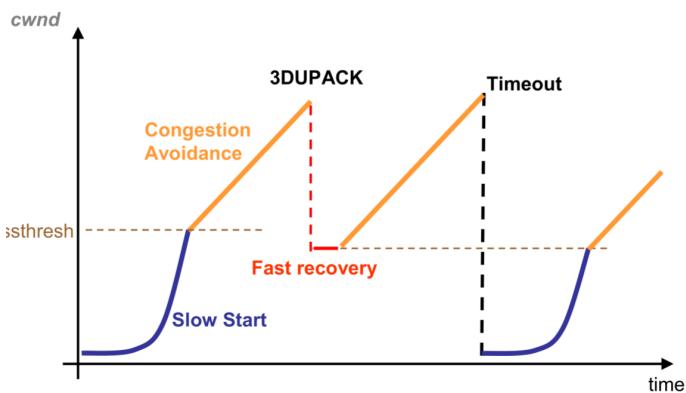


TCP Reno distingue i due eventi di congestione, (considerando che un 3DUPACK è un evento di congestione più "leggero" rispetto ad un RTO):

- In caso di 3DUPACK, la cwnd viene dimezzata e si entra nella fase di fast recovery, in cui la cwnd viene aumentata linearmente fino a raggiungere la soglia sshtresh.
- In caso di RTO, la cwnd viene riportata a 1 e si entra nella fase di slow start.

Figure 3.23: Andamento della finestra di congestione in Reno

## TCP New Reno



Nel reno visto in precedenza ad ogni 3DUPACK viene dimezzata la cwnd, ma questo non è ottimale nel caso in cui avvengano più 3DUPACK nella stessa finestra.

Perciò esiste la variante New Reno, che non dimezza la cwnd ad ogni 3DUPACK, lo fa solamente la prima volta che avviene all'interno della finestra; quelli successivi vengono ignorati.

Figure 3.24: Andamento della finestra di cong. in New Reno

# Chapter 4

## Livello 3 rete

### 4.1 Internet Protocol (IP) version 4

#### Introduzione

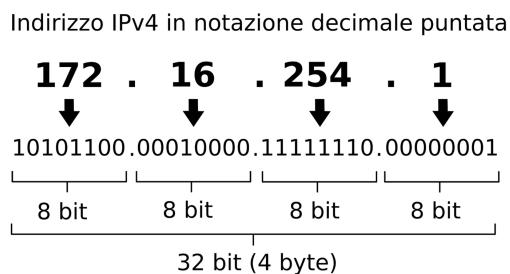


Figure 4.1: Struttura di un indirizzo IP

Al livello 3 ci preoccupiamo di trovare la strada migliore per raggiungere il destinatario.

IPv4 è il protocollo di rete più diffuso al mondo, IP instrada i pacchetti immessi nella rete, ogni nodo in rete ha indirizzo ip.

Ogni pdu(pacchetto) contiene gli indirizzi IP del host mittente e destinatario.

IP è connectionless, ad inoltrare i pacchetti(datagram) sono i router, lo fanno attraverso algoritmi di routing(leggendo l'header del pacchetto).

Ip è inoltre “best-effort”, ossia fa del suo meglio, non garantisce affidabilità massima o ottimale.

Gli indirizzi ip sono da 32 bit(4 numeri decimali compresi tra 0 e 255):

Gli host che appartengono alla stessa rete hanno in comune il livello fisico, un dominio di broadcast è un insieme di computer di una stessa rete(stesso indirizzo ip di rete), esempio rete di casa(collegandoci al router di casa è possibile inviare un messaggio broadcast che arriverà a tutti i dispositivi che fanno parte della rete IP). Le porte del router vengono chiamate interfacce, ogni interfaccia corrisponde ad un host; ogni rete ha un indirizzo IP di rete.

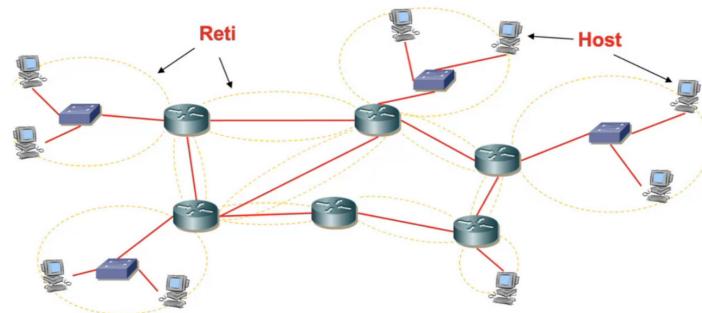


Figure 4.2: Esempio di rete IP di base

#### 4.1.1 Datagramma IP

I datagrams IP sono i pacchetti di dati che vengono inviati attraverso la rete utilizzando il protocollo IP. Ogni datagramma contiene un header e un payload.

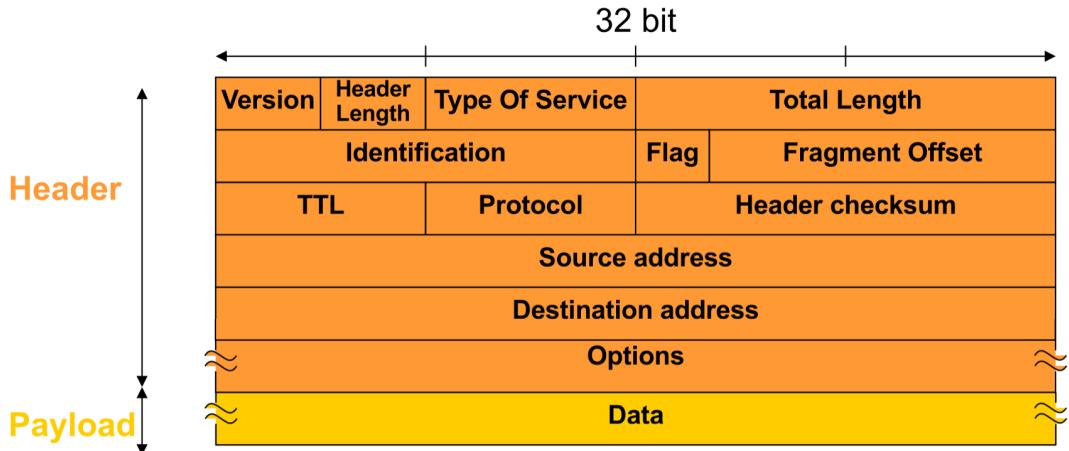


Figure 4.3: Struttura di un datagramma IP

- version: mi dice la versione del protocollo(IPv4 o IPv6) (è di 4 bit, mezzo byte)
- header length: lunghezza del header (4 bit, mezzo byte), se voglio dire che l'header è lungo 5 posso scrivere in questo campo 0101, che è  $1 + 0 + 4 + 0$
- type of service(TOS): (1 byte, 8 bit) distingue i tipi diversi di pacchetti, questo è utile nel momento in cui un router deve dare o no la precedenza a certi pacchetti, in base al tipo di servizio. esempio: precedenza al servizio x piuttosto che a y il TOS viene utilizzato per definire il livello di quality of service con cui trattare il pacchetto(primi 6 bit)(gli ultimi 2 servono ad evitare la congestione)



Figure 4.4: Campo Type of Service (ToS) nell'header IP

- total length:(16 bit) lunghezza del datagramma in byte(max 65535 byte)
- identification(16 bit) identifica in modo univoco un datagramma generato dal mittente, quindi anche se il pacchetto viene frammentato, i suoi “frammenti”, ossia altri pacchetti che contengono parte del payload iniziale, avranno lo stesso id. serve quindi a gestire la frammentazione dei pacchetti ip
- options: opzioni per il routing del pacchetto, è un campo poco usato per via del carico di elaborazione molto variabile
- data: è l'informazione da inviare

- flag(3 bit)

il primo bit non usato, il secondo bit è il flag DF (don't fragment: specifica che il router può frammentare il pacchetto o no), il terzo bit è il flag MF (more fragment: indica se il seguente datagramma è l'ultimo dei frammenti). Per utilizzare questi flag metto il valore del bit corrispondente al flag ad 1. Esempio: 010, 0 iniziale è indifferente, 1 indica il flag DF attivo, mentre lo 0 alla fine mi dice che non è MF, quindi che questo datagram è l'ultimo frammento.

- fragment offset(13 bit) mi dice la posizione del frammento all'interno del pacchetto originario, lo spiazzamento. è espresso in multipli di 8 byte.

**Frammentazione:** quando un datagramma IP deve attraversare una rete con una MTU (Maximum Transmission Unit) inferiore alla sua dimensione, viene suddiviso in frammenti più piccoli. Ogni frammento contiene parte dei dati originali e un header IP con informazioni per il riasssemblaggio.

Nella figura il pacchetto è grande 4000 byte(total length), in questo caso viene frammentato in 3 poichè la sua MTU vale 1500 byte , i frammenti condividono lo stesso id del pacchetto non frammentato.

I frammenti devono avere un offset multiplo di 8.

Solamente il primo frammento ha offset pari a 0, rappresenta la testa. Solamente l'ultimo frammento ha MF pari a 0 poichè sta ad indicare che quello è l'ultimo pezzo del pacchetto, la coda. CHIARIRE



Figure 4.5: Campo Flags nell'header IP

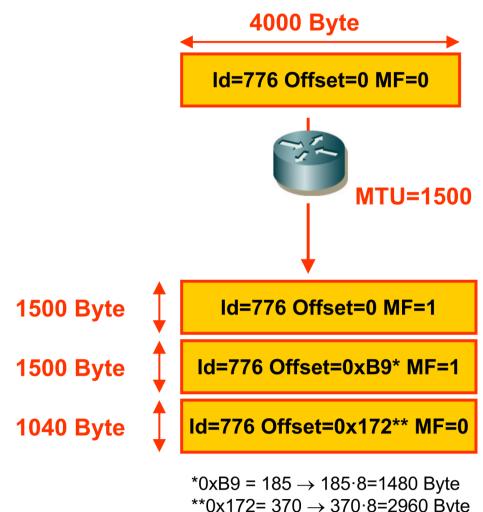


Figure 4.6: Esempio di frammentazione di un datagramma IP

- time to live(8 bit) evita il problema dei pacchetti che entrano in loop poichè non trovano il destinatario. è il tempo di vita del pacchetto, ogni router che riceve il pacchetto(ad ogni hop) decrementa il valore time to live, facendo così arriverà un punto in cui il time to live del pacchetto è stato decrementato fino a zero. quando arriva a 0 il datagram viene scartato. è un valore stabilito dal mittente(esempio 128)
  - protocol(8 bit): contiene un numero che specifica il protocollo usato dalla PDU encapsulata dal datagram. esempio TCP = 6 . UDP = 17 → livello 4
  - header checksum(16 bit): controlla i byte del header, è un controllo veloce su header. se c'è un errore sull'header allora il pacchetto viene scartato, come funziona? : somma 16 bit a 16 bit gli altri byte dell'header, fa la somma dell'eventuale resto e fa il complemento ad 1 del risultato.
- Se non ci sono errori, la somma a 16 bit di tutti questi valori deve dare come risultato 0xFFFF (cioè tutti i bit a 1).
- source/destination address: (32 bit) è l'indirizzo ip del mittente/destinatario

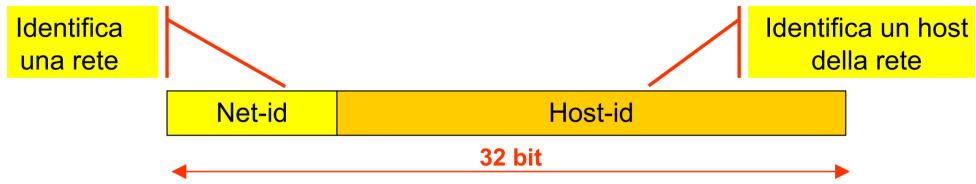


Figure 4.7: Struttura di un indirizzo IP: suddivisione in parte di rete e parte host

### Struttura indirizzo IP

Attualmente si utilizza l'indirizzamento classless CIDR (Classless Inter-Domain Routing), che permette di specificare la lunghezza della maschera di rete in bit.

L'assegnazione degli indirizzi è compito dell'Internet Assigned Number Authority (IANA) gestita dall'ICANN (Internet Corporation for Assigned Names and Numbers).

Ogni interfaccia (host) della rete IP ha un suo indirizzo.

#### 4.1.2 Indirizzo IP - classful

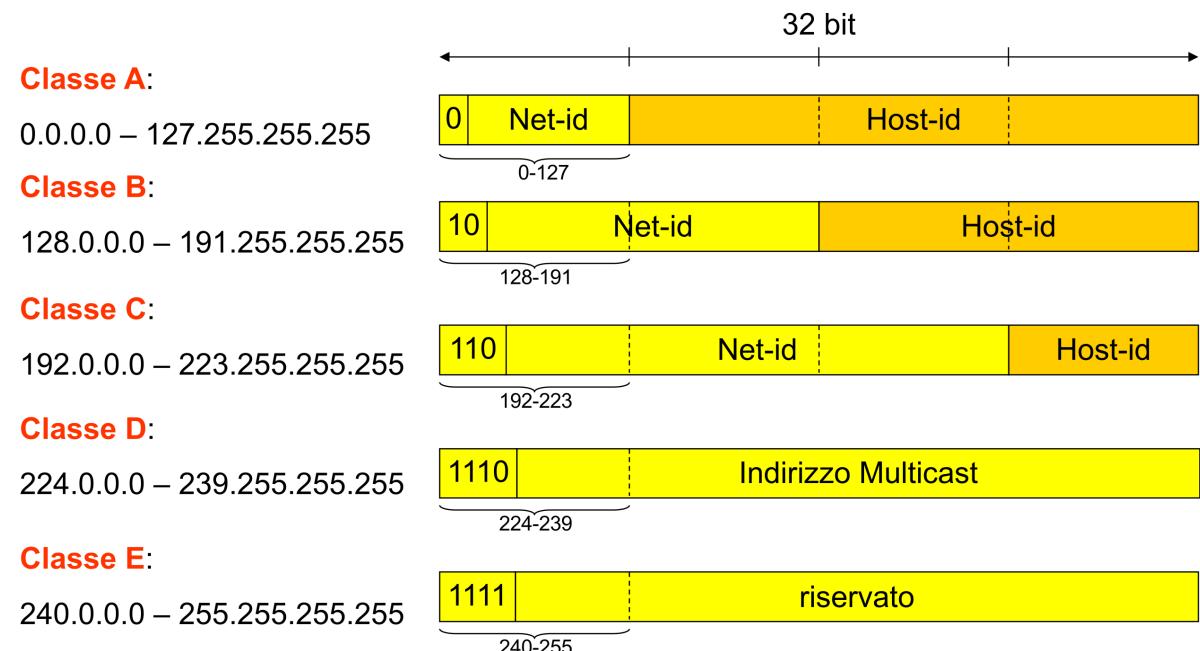
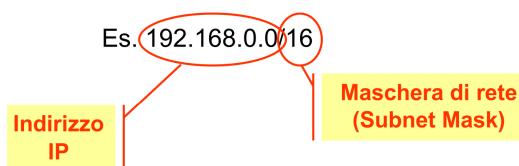


Figure 4.8: Classful Addressing: suddivisione degli indirizzi IP in classi A, B, C, D, E

#### 4.1.3 Indirizzo IP - classless CIDR



Il formato dell'indirizzo è del tipo  $a.b.c.d/x$  dove  $x$  rappresenta la lunghezza della maschera di rete in bit. La maschera di rete permette di distinguere la parte di indirizzo relativa alla rete da quella relativa all'host. In notazione CIDR, la maschera viene indicata dopo una barra, ad esempio 192.168.1.0/24.

Figure 4.9: Esempio di maschera di rete (net-mask) in notazione decimale e binaria

#### 4.1.4 Address Resolution Protocol (ARP) - mappatura IP a MAC

Con ART si intende un protocollo di rete appartenente alla suite del protocollo internet (IP) versione 4 e operante a livello di accesso alla rete (livello collegamento se si considera nomenclatura ISO/OSI), il cui compito è fornire la "mappatura" tra l'indirizzo IP (32 bit - 4 byte) e l'indirizzo MAC (48 bit - 6 byte) corrispondente di un terminale in una rete locale ethernet.

Cos'è l'indirizzo MAC?

Un indirizzo MAC (dove MAC sta per Media Access Control), detto anche "indirizzo fisico" o "indirizzo Ethernet", è un codice di 48 bit associato ad ogni dispositivo di rete che implementa lo standard Ethernet. Il suo scopo principale è quello di attribuire un'identità univoca a ciascuno dei nodi collegati ad uno stesso segmento di rete, consentendo quindi comunicazioni locali di tipo unicast.

Al Livello 2 del modello ibrido, i pacchetti viaggiano attraverso il collegamento di più nodi, questi nodi sono caratterizzati da indirizzi fisici detti mac. Invece a livello tre l'indirizzo che possiede il pacchetto è l'indirizzo definitivo!(il destinatario, non i nodi intermedi), la metà ultima del pacchetto. Però questi pacchetti viaggiano attraverso dei nodi, quindi a livello 2 i pacchetti avranno come destinazioni i nodi, e per raggiungerli hanno bisogno di conoscere gli indirizzi fisici dei nodi, perciò tramite gli indirizzi MAC e il protocollo arp si risolve il problema locale del collegamento tra i vari nodi che fanno sì che il pacchetto viaggi attraverso quelli.

ARP risponde quindi a questa domanda: ho questo indirizzo ip, mi dici quale sia l'indirizzo di livello due corrispondente a questo ind ip??

Mi dice l'indirizzo del "prossimo nodo", livello 2.

Quindi un protocollo fondamentale che individua da un indirizzo ip (liv 3) uno o più di livello 2

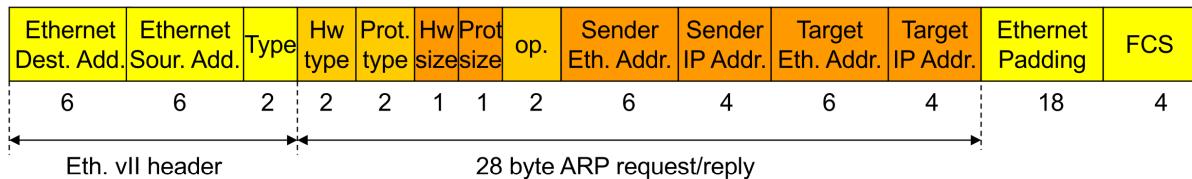


Figure 4.10: Struttura di un frame Ethernet

Questo è il frame ethernet: in giallo ho l'header e la coda di ethernet , la coda svolge il ruolo di verificare che ci siano errori(fcs da 4 byte), il campo padding in coda serve a raggiungere i 64 byte(è un filler) minimi del frame totale tra ethernet e arp(arpa è interno a 1 frame ethernet) il frame deve avere una dimensione minima di 64 byte, il protocollo arp è quello arancione chiaro e scuro al centro.

Struttura del frame ethernet:

- ethernet destination address: ind liv 2 del mittente - uguale a sender eth address
  - ethernet source address
  - type: tipologia tecnologia del livello 2
  - hw type: specifica l'hardware che si sta usando al livello 2
  - protocol type: specifica che protocollo si vuole convertire dal liv 3 al 2(quindi non ha utilizzo esclusivo per indirizzi ip)
  - hw size: dimensione indirizzo di liv 3
  - prot size: dimensione indirizzo liv 2
  - options:
  - sender ethernet address: indirizzo liv 2 del mittente
  - sender ip address: indirizzo liv 3 del mittente
  - target ethernet address: indirizzo liv 2 del destinatario
  - target ip address: indirizzo liv 3 del destinatario
  - op: operazione che si vuole effettuare, 1 per richiesta arp, 2 per risposta arp

## Esempio ARP all'interno di una rete

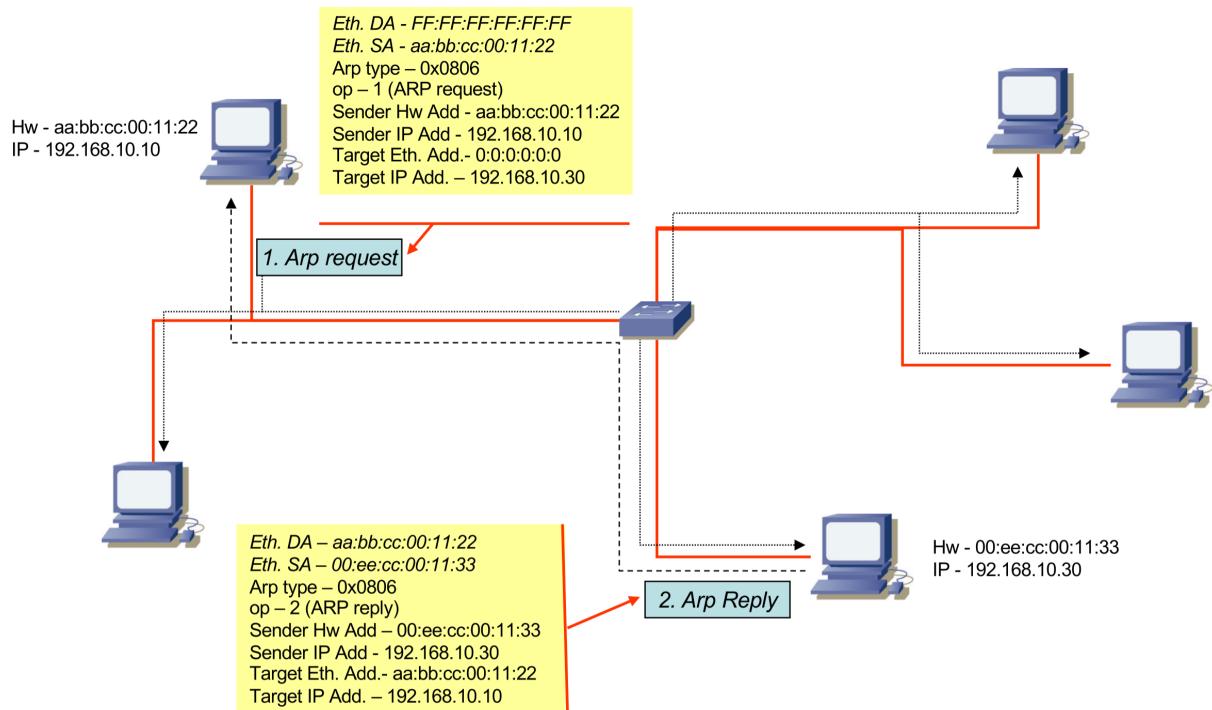


Figure 4.11: Esempio di funzionamento del protocollo ARP

L'indirizzo hw è l'indirizzo fisico anche detto mac, è tipicamente in esadecimali ed è di 6 byte.

I riquadri in giallo sono pacchetti ARP utili alla comunicazione tra i due pc(visto che il dispositivo sta utilizzando arp per inviare un pacchetto allora sta usando un arp request, che viene segnato come op - 1 nel frame ethernet); i pacchetti in giallo quindi encapsulano una serie di informazioni del frame ethernet.

Inoltre l'Ethernet destination address viene impostato a ff:ff:ff:ff:ff:ff (broadcast per indirizzi ARP) se viene effettuata un'ARP Request, questo perchè il mittente sta cercando di individuare l'indirizzo fisico del destinatario, di cui conosce solo l'ip, perciò utilizza ARP.

Nel momento della ARP request non so l'indirizzo ethernet del destinatario, ma so solo l'indirizzo ip, quindi ho tutti i byte del Ethernet address del target a 0.

Manderò quindi in broadcast(quindi multicast: a tutti i dispositivi della rete)(linee non tratteggiate) una richiesta a tutti i computer della rete, aspettando che il computer con l'indirizzo ip corrispondente a quello del target mi invii in unicast(linea tratteggiata) con un altro pacchetto ARP(arp reply, op - 2).

A questo punto il computer che ha inviato inizialmente l'ARP request riceve le info necessarie per poter inviare il pacchetto al destinatario.

### Esempio ARP in reti diverse

IMPORTANTE: IL TCP(livello 4) si disinteressa di questo percorso, opera come se A e B fossero già collegati(end to end).

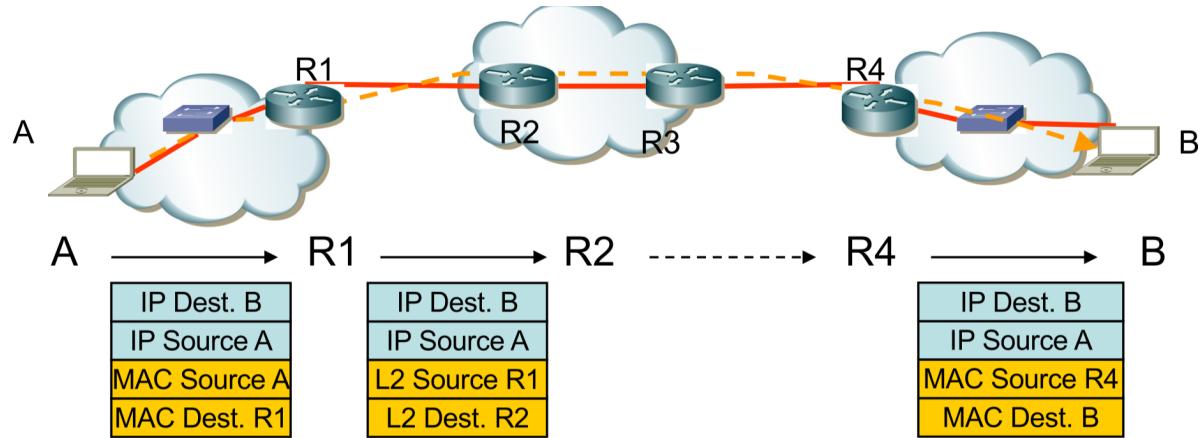


Figure 4.12: Esempio di funzionamento del protocollo ARP tra reti diverse

**Default gateway e tabelle di routing** per scoprire se la rete del destinatario è la stessa da cui mando la richiesta arp: applico la mia maschera di rete all'ip del destinatario (la mia maschera di rete è un'informazione che posseggo). quindi se la rete del destinatario non corrisponde a quella del mittente, per "uscire" e connettermi all'altra rete devo passare per il router. gli indirizzi di livello 3, quindi l'ip del destinatario e l'ip del mittente sono info che non variano tra un passaggio da un router all'altro, mentre quelli in giallo, gli indirizzi fisici(mac) variano tra router all'altro. quindi il protocollo arp verrà applicato ad ogni passaggio tra un router all'altro cambiando quindi i mac source e mac destinatario, inserendo i mac dei dispositivi(router) tra cui avviene questo passaggio d'informazione. fino ad arrivare al dispositivo con l'indirizzo ip contenuto nel frame (ip dest B) Il primo router a cui ci appoggiamo per l'inolitramento del pacchetto viene chiamato default gateway.

I router servono a connettere reti diverse, quindi se il destinatario non è nella mia rete allora mi appoggio al router, che ha un indirizzo ip di default, che è il default gateway.

Le tabelle di routing sono delle tabelle che i router utilizzano per sapere come inoltrare i pacchetti verso le reti di destinazione. Queste tabelle contengono informazioni sulle reti raggiungibili, i loro indirizzi IP e fisici.

A fine processo B legge i livelli 2 e 3 e riconosce i propri indirizzi MAC e IP, a quel punto può leggere il payload del pacchetto, che contiene i dati che A voleva inviare a B.

**ARP nella stessa rete** Se nel processo in cui cerco di capire se il destinatario sta nella mia rete o no scopro che sta nella mia rete allora non mi serve il router e posso inviare direttamente il pacchetto al destinatario, ma prima devo scoprire il suo indirizzo fisico(mac), quindi applico arp come fatto inizialmente.

#### 4.1.5 Dynamic Host Configuration Protocol (DHCP) - configurazione automatica degli indirizzi IP tramite server DHCP

Un indirizzo ip può essere statico o dinamico, servers e router hanno tipicamente indirizzi statici.

Gli ind statici si configurano manualmente, quelli dinamici invece sono configurati tramite DHCP, un server che va configurato dall'utente ed è utile perché fornisce:

- indirizzo ip del dispositivo
- netmask
- indirizzo ip del router
- indirizzo ip del server DNS
- lease time

Il lease time è il tempo in cui l'indirizzo ip è riservato al dispositivo, dopo di che il dispositivo deve richiedere un nuovo indirizzo ip; è anche possibile ottenere un indirizzo IP riservato, sulla base dell'indirizzo fisico MAC.

**DHCP relay - inoltro delle richieste DHCP** Se nella rete locale non è disponibile un DHCP server, allora la richiesta può essere inoltrata verso un'altra rete che dispone della funzione DHCP relay.

**DHCP - funzionamento** Questo protocollo fa uso dei UDP per la comunicazione tra client e server, e funziona in modo simile a ARP.

L'host che desidera un indirizzo IP si affida a questo protocollo per ottenerlo, inviando un messaggio DHCP discover, incapsulandolo in un pacchetto UDP, che viene inviato in broadcast a tutti i dispositivi della rete locale.

All'interno del pacchetto UDP, il mittente inserisce i seguenti indirizzi IP:

- IP source address: l'indirizzo IP del mittente, che non è ancora stato assegnato, viene impostato momentaneamente a 0.0.0.0
- IP destination address: l'indirizzo IP del server DHCP, viene impostato a 255.255.255.255

Tutti gli eventuali server DHCP presenti nella rete rispondono in broadcast con un messaggio DHCP offer, che contiene le informazioni richieste dal client, come l'indirizzo IP assegnato, la netmask ecc...

Il client riceve le offerte dai server DHCP e sceglie una di esse, inviando un messaggio DHCP request al server scelto.

Infine il server DHCP conferma l'assegnazione dell'indirizzo IP con un messaggio DHCP ack.

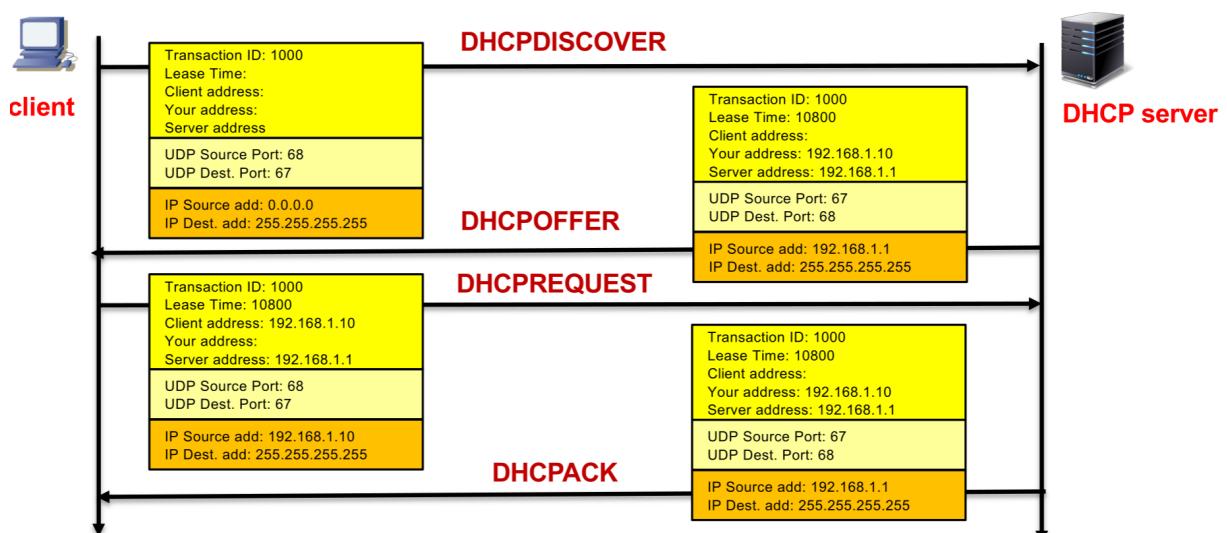


Figure 4.13: Funzionamento del protocollo DHCP

#### 4.1.6 Zero-configuration networking

Il zero-configuration networking è la configurazione della rete in assenza di server e amministratori, è ideale solo per reti piccole e semplici. Non è quindi un protocollo alternativo a DHCP.

**Link-Local Address** La IANA ha riservato a questo scopo un Link-Local Address: 169.254.0.0/16

Gli indirizzi IP dei dispositivi nella rete vengono assegnati automaticamente, all'interno di un range specifico: [169.254.1.1 - 169.254.254]

**ARP probe** Quando un dispositivo si connette alla rete, invia un ARP probe per verificare se l'indirizzo IP scelto è già in uso. Se non riceve risposta, assume che l'indirizzo sia disponibile e lo utilizza. Se un altro host ha lo stesso indirizzo, sceglie casualmente un altro indirizzo IP all'interno dello stesso range.

**ARP announcement** Dopo aver scelto un indirizzo IP, il dispositivo invia un ARP announcement per informare gli altri dispositivi della rete del suo nuovo indirizzo.

Questi indirizzi sono di tipo riservato e non possono essere usati per comunicare fuori dalla rete locale.

#### 4.1.7 Network Address Translation (NAT) - da IP privato a IP pubblico

Il NAT è un software eseguito dal router di frontiera (ha il compito di far uscire dalla rete locale i pacchetti destinati ad indirizzi ip di altre reti).

Quindi è quel servizio che converte l'indirizzo privato del router di frontiera, in uno pubblico, così da comunicare con "l'esterno". Perciò i router di frontiera delle altre reti non vedono l'indirizzo privato ma quello pubblico, convertito grazie al NAT.

Più precisamente, il NAT è una famiglia di tecniche/protocolli.

**Funzionamento del NAT e tabelle di traduzione** Tutti i datagram IP che viaggiano da (verso) la LAN hanno il medesimo indirizzo IP mittente (destinazione), poiché il router di frontiera ha un solo indirizzo IP pubblico, che viene utilizzato per tutte le comunicazioni in uscita (entrata) dalla rete locale(LAN).

Il NAT utilizza una tabella di traduzione per mappare gli indirizzi IP privati della rete locale con l'indirizzo IP pubblico del router di frontiera; utilizzando le porte come ulteriore identificativi.

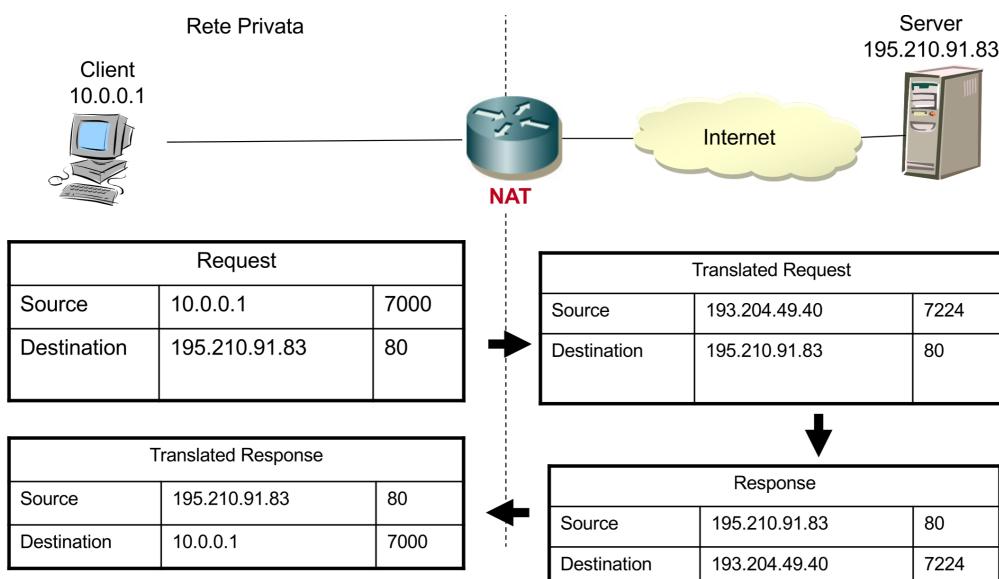


Figure 4.14: Funzionamento del NAT: traduzione degli indirizzi IP privati in indirizzi IP pubblici

#### 4.1.8 Internet control message protocol (ICMP) - diagnostica della rete

I messaggi ICMP si dividono messaggi di errore e messaggi di richiesta; vengono inviati attraverso i pacchetti IP(protocol type 1); questo protocollo serve a segnalare informazioni di controllo relative al livello di rete, un messaggio ICMP è composto dai campi type e code:

- type: indica il tipo di messaggio ICMP
- code: dettagli sul tipo di messaggio

ICMP type	Code	Description
0	0	Echo replay – risposta al messaggio di eco (ping)
3	0	Destination network unreachable
3	1	Destination host unreachable
3	2	Destination protocol unreachable
3	3	Destination port unreachable
3	6	Destination network unknown
3	7	Destination host unknown
4	0	Source quench – riduzione data rate
8	0	Echo request
9	0	Router advertisement
10	0	Router discovery
11	0	TTL expired
12	0	IP header bad

Figure 4.15: Principali comandi ICMP e loro utilizzo

**Esempio utilizzo di ICMP - ping** Ad esempio, echo request e echo reply sono alla base del comando ping. Type 8 corrisponde a echo request, mentre type 0 corrisponde a echo reply.

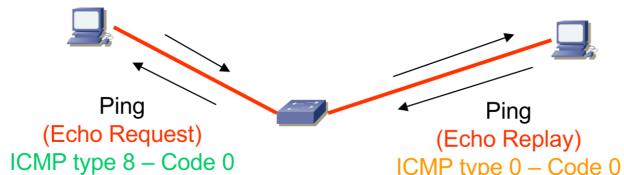


Figure 4.16: Esempio di funzionamento di ICMP tramite il comando ping

**Esempio utilizzo di ICMP - traceroute** Tramite il traceroute( utilizzo dei ICMP type 11) scopro il percorso che sta seguendo il pacchetto comandi: traceroute sitoweb

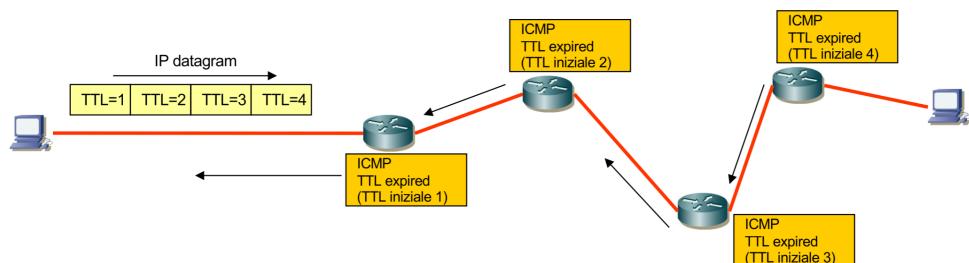


Figure 4.17: Esempio di funzionamento di ICMP tramite il comando traceroute

## 4.2 Algoritmi di routing

### 4.2.1 Introduzione agli algoritmi di routing - tipologie

Gli algoritmi di routing sono utilizzati dai router per determinare il percorso migliore che i pacchetti devono seguire per raggiungere la loro destinazione.

Questi algoritmi possono essere classificati in base al loro uso o meno delle tabelle di routing o se sono gerarchici.

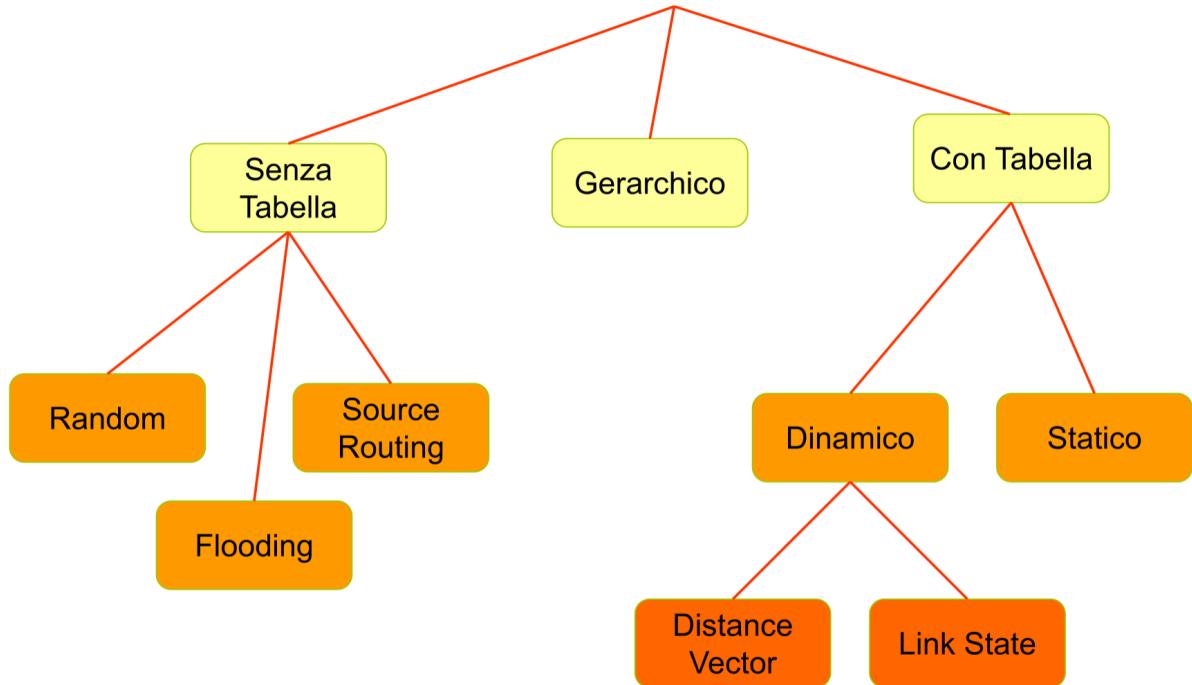


Figure 4.18: Tipologie di algoritmi di routing

- Con tabella: il router conserva in memoria una tabella per capire qual è la strada migliore, contiene dati per scegliere la strada migliore. Come costo si intende il numero di hop, la banda, l'affidabilità,

Indirizzo Rete Destinazione	Costo	Linea di Uscita/Next hop
192.168.59.0/24	....	L1
....	....	....
Default Route	....	L2

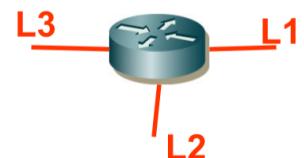


Figure 4.19: Esempio di tabella di routing

il ritardo o anche un valore inserito manualmente; dipende strettamente dal metodo che si vuole utilizzare.

- Senza tabella: includono il routing random, il flooding e il source routing
- Gerarchici: i router affidano le scelte di routing ad altri router "più intelligenti"?

!Instradare significa scegliere il percorso migliore per raggiungere la destinazione, il forwarding invece è l'azione di inviare il pacchetto per la strada scelta!

#### 4.2.2 Metriche per il costo del percorso

Il costo del percorso viene calcolato sulla base di diverse metriche:

- **Bandwidth**: la capacità di un link (es. generalmente un link a 10 Mbps è preferibile rispetto ad una linea a 64 kbps)
- **Delay**: il tempo necessario ad ogni pacchetto per andare dalla sorgente alla destinazione
- **Load**: il carico di lavoro degli elementi della rete come i router o i link
- **Reliability**: l'affidabilità, generalmente riferita al tasso di errore di ogni singolo link
- **Hop count**: il numero di router che un pacchetto deve attraversare per raggiungere la destinazione
- **Ticks**: il ritardo di un collegamento dati in multipli di un *IBM PC clock tick* (circa 55 ms)
- **Cost**: un valore arbitrario, generalmente basato sulla banda, sul costo economico di un link, o su altre misure stabilite dall'amministratore di rete

**Cosa fa il router?** Il router è un dispositivo fondamentale nelle reti di comunicazione, in quanto si occupa di instradare i pacchetti dati tra reti diverse. Il suo compito principale è ricevere i pacchetti in ingresso su una delle sue interfacce, esaminare l'indirizzo IP di destinazione e, in base alle informazioni contenute nella propria tabella di routing, decidere su quale interfaccia inoltrare ciascun pacchetto affinché raggiunga la destinazione finale nel modo più efficiente possibile. Il router aggiorna periodicamente la propria tabella di routing tramite protocolli specifici e può adattare le proprie decisioni in base ai cambiamenti della topologia di rete. In sintesi, il router permette la comunicazione tra reti diverse, ottimizzando il percorso dei dati e garantendo che i pacchetti raggiungano correttamente il destinatario.

**Funzionamento della tabella di routing** Quando un router riceve un datagramma, esegue i seguenti passaggi per determinare dove inoltrarlo:

1. Cerca nella tabella di routing una entry con lo stesso indirizzo di rete del destinatario del datagramma ricevuto.
2. Se la entry corrisponde a una rete direttamente connessa al router, viene individuata la sottorete e il datagramma è inoltrato direttamente.
3. Se la entry corrisponde a una rete remota, il datagramma è inoltrato (forwarding) verso il prossimo hop specificato nella tabella.
4. Se nessuna entry è trovata, viene utilizzato il *default route*

#### 4.2.3 Instradamento gerarchico - protocolli intra e inter AS

Al fine di comprendere il routing è indispensabile introdurre il concetto di Autonomous System(AS).

Un AS è l'insieme di router amministrati dallo stesso gestore; è possibile distinguere i vari protocolli di routing proprio sulla base di questo.

Infatti i protocolli che lavorano all'interno di un AS vengono chiamati intra AS, mentre quelli che operano tra Autonomous Systems differenti vengono detti inter AS.

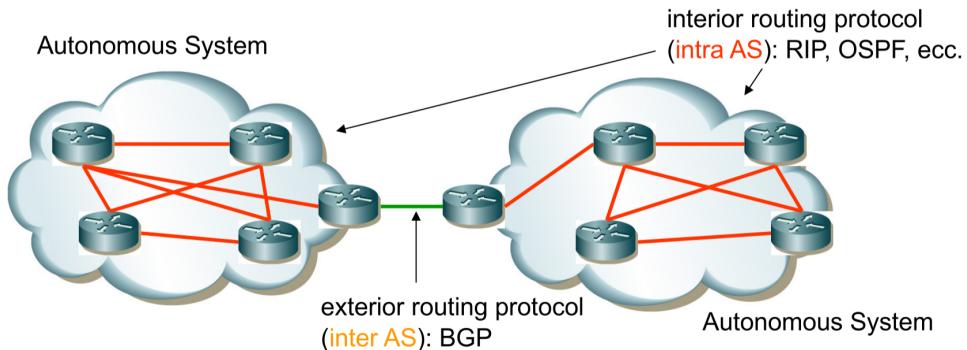


Figure 4.20: Esempio di gerarchia tra Autonomous Systems (AS)

#### 4.2.4 Distance Vector

È una famiglia di algoritmi di routing in cui ogni router scambia informazioni con i vicini, usa l'algoritmo di Bellman-Ford per calcolare il percorso più breve. Un algoritmo di routing che fa uso di tabelle dinamiche è il distance vector:

ogni router prende decisioni in modo autonomo; ogni router invia in modo periodico ai router vicini i "vettori delle distanze" (distance vector).

Le colonne della tabella che interessano particolarmente questo algoritmo sono quella dei costi e quella delle destinazioni note.

In questo modo tra i router ci si scambiano informazioni utili al routing stesso, prendendo informazioni dai router vicini

Il vettore distanza quindi comunica ai vicini i percorsi che conosce e il costo di tali percorsi (percorsi che passano dal router che condivide tali percorsi) le tabelle sono dinamiche perché la topologia della rete può cambiare nel tempo.

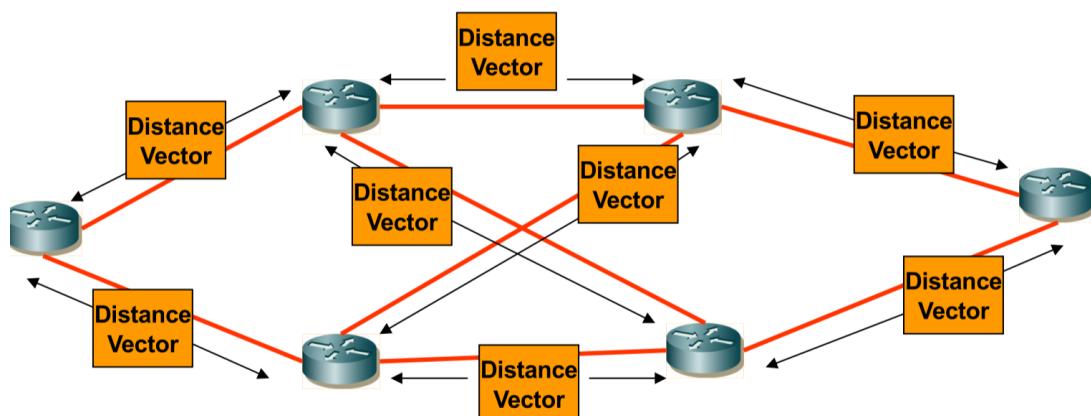


Figure 4.21: Esempio di funzionamento dell'algoritmo Distance Vector

#### 4.2.5 Algoritmo Bellman-Ford

Il funzionamento dell'algoritmo di routing distance-vector porta ad una soluzione che si otterrebbe equivalentemente applicando l'algoritmo di Bellman-Ford per la ricerca del cammino a costo minimo in un grafo.

I router non applicano questo algoritmo, ma la soluzione finale cui si giunge è identica.

Questo algoritmo lavora su una rappresentazione globale del grafo, cosa che nei router reali non accade poiché non conoscono tutta la rete, ma solo le info sui e dei vicini diretti.

I protocolli distance vector sono una versione distribuita del Bellman-Ford, cioè: l'algoritmo non è eseguito da un singolo punto centrale, ma ogni nodo della rete (es. ogni router) esegue una parte dell'algoritmo!

#### Albero dei cammini a costo minimo

##### Variabili

- $s$  è il router da cui parte l'algoritmo e di cui interessa calcolare il costo minimo
- $h$  indica il numero dei salti
- $j$  indica un nodo
- $i$  indica un nodo differente da  $j$
- $D_j^{(h)}$  è il costo del cammino minimo tra la sorgente  $s$  e il nodo  $j$  con massimo  $h$  salti
- $d_{ij}$  è il costo del collegamento diretto tra i nodi  $i$  e  $j$ ; questo costo è infinito se i nodi non sono collegati direttamente

#### Fasi dell'algoritmo

1. Inizializzazione:  $h = 0$

- La distanza del nodo da se stesso  $D_s^{(0)} = 0$
- La distanza tra  $s$  e i nodi non diretti è infinita  $D_j^{(0)} = \infty$  per tutti  $j \neq s$

2. Iterazione: per ogni  $h > 0$

Hop successivo  $h = h + 1$

Calcolo dei cammini al salto successivo  $D_j^{(h)} = \min_i [D_i^{(h-1)} + d_{ij}^{(h-1)}]$

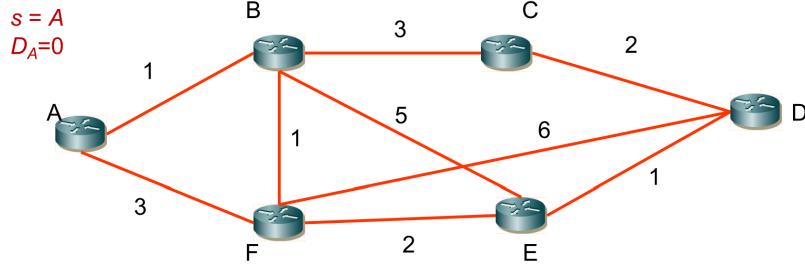
3. Verifica: tramite questo controllo capisco se il cammino  $D_j^{(h)}$  calcolato è uguale al precedente, se così fosse allora non ha senso continuare e si setta come  $h_{max}$  quello precedente, si ferma l'algoritmo; in caso contrario l'algoritmo prosegue dal passaggio due.

If  $D_j^h = D_j^{h-1} \quad \forall j \neq s$

then  $h_{max} = h - 1$ ; stop

else go to 2

**Esempio** Il primo passo è calcolare i costi dei cammini quando  $h = 0$ , ricordando che quando non c'è collegamento diretto avrà costo infinito.



$$D_B = D_C = D_D = D_E = D_F = \infty$$

Figure 4.22: Esempio di applicazione dell'algoritmo di Bellman-Ford - Passo 1

Poi si calcolano i costi dei cammini quando  $h = 1$ , quindi con al massimo un salto.

- $D_B^{(1)} = \min[D_A^{(0)} + d_{AB}, D_C^{(0)} + d_{CB}, D_D^{(0)} + d_{DB}, D_E^{(0)} + d_{EB}, D_F^{(0)} + d_{FB}, D_B^{(0)}] = \min[0 + 1, \infty + 3, \infty + \infty, \infty + 5, \infty + 1, \infty] = 1$
- $D_C^{(1)} = \min[D_A^{(0)} + d_{AC}, D_B^{(0)} + d_{BC}, D_D^{(0)} + d_{DC}, D_E^{(0)} + d_{EC}, D_F^{(0)} + d_{FC}, D_C^{(0)}] = \min[0 + 3, \infty + 3, \infty + \infty, \infty + 2, \infty + \infty, \infty] = 3$
- $D_D^{(1)} = \min[D_A^{(0)} + d_{AD}, D_B^{(0)} + d_{BD}, D_C^{(0)} + d_{CD}, D_E^{(0)} + d_{ED}, D_F^{(0)} + d_{FD}, D_D^{(0)}] = \min[0 + \infty, \infty + \infty, \infty + \infty, \infty + 1, \infty + 5, \infty] = \infty$
- $D_E^{(1)} = \min[D_A^{(0)} + d_{AE}, D_B^{(0)} + d_{BE}, D_C^{(0)} + d_{CE}, D_D^{(0)} + d_{DE}, D_F^{(0)} + d_{FE}, D_E^{(0)}] = \min[0 + \infty, \infty + 5, \infty + 2, \infty + 1, \infty + \infty, \infty] = \infty$
- $D_F^{(1)} = \min[D_A^{(0)} + d_{AF}, D_B^{(0)} + d_{BF}, D_C^{(0)} + d_{CF}, D_D^{(0)} + d_{DF}, D_E^{(0)} + d_{EF}, D_F^{(0)}] = \min[0 + \infty, \infty + 1, \infty + \infty, \infty + 5, \infty + \infty, \infty] = \infty$

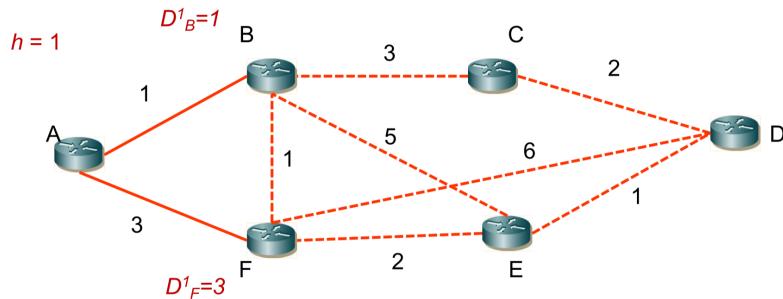


Figure 4.23: Esempio di applicazione dell'algoritmo di Bellman-Ford - Passo 2

- $D_B^{(2)} = \min[0 + 1, 3 + 3, \infty + \infty, \infty + 5, \infty + 1, 1] = 1$
- $D_C^{(2)} = \min[0 + 3, 1 + 3, \infty + \infty, \infty + 2, \infty + \infty, 3] = 3$
- $D_D^{(2)} = \min[0 + \infty, 1 + \infty, 3 + \infty, \infty + 1, \infty + 5, \infty] = \infty$
- $D_E^{(2)} = \min[0 + \infty, 1 + 5, 3 + 2, \infty + 1, \infty + \infty, \infty] = 5$

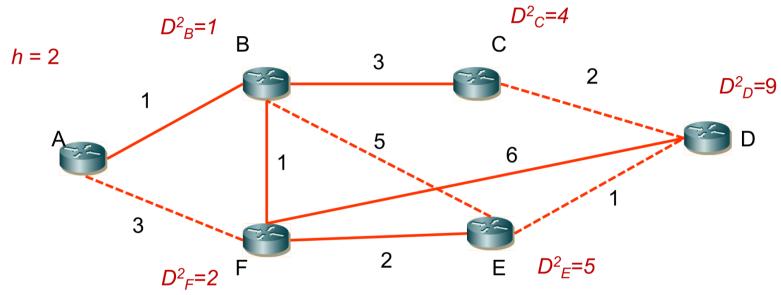


Figure 4.24: Esempio di applicazione dell'algoritmo di Bellman-Ford - Passo 3

- $D_F^{(2)} = \min[0 + \infty, 1 + 1, 3 + \infty, \infty + 5, \infty + \infty, \infty] = 2$
- $D_B^{(3)} = \min[0 + 1, 3 + 3, \infty + \infty, 5 + 5, 2 + 1, 1] = 1$
- $D_C^{(3)} = \min[0 + 3, 1 + 3, \infty + \infty, 5 + 2, 2 + \infty, 3] = 3$
- $D_D^{(3)} = \min[0 + \infty, 1 + \infty, 3 + \infty, 5 + 1, 2 + 5, \infty] = 6$
- $D_E^{(3)} = \min[0 + \infty, 1 + 5, 3 + 2, \infty + 1, 2 + \infty, 5] = 5$
- $D_F^{(3)} = \min[0 + \infty, 1 + 1, 3 + \infty, \infty + 5, 5 + \infty, 2] = 2$

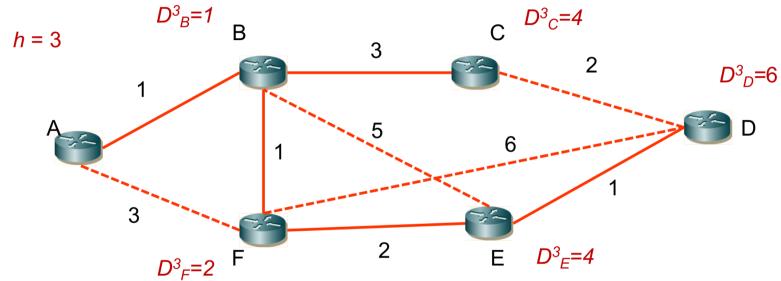


Figure 4.25: Esempio di applicazione dell'algoritmo di Bellman-Ford - Passo 4

Poi si calcolano i costi dei cammini quando  $h = 4$

- $D_B^{(4)} = \min[0 + 1, 3 + 3, 6 + \infty, 5 + 5, 2 + 1, 1] = 1$
- $D_C^{(4)} = \min[0 + 3, 1 + 3, 6 + \infty, 5 + 2, 2 + \infty, 3] = 3$
- $D_D^{(4)} = \min[0 + \infty, 1 + \infty, 3 + \infty, 5 + 1, 2 + 5, 6] = 6$
- $D_E^{(4)} = \min[0 + \infty, 1 + 5, 3 + 2, 6 + 1, 2 + \infty, 5] = 5$
- $D_F^{(4)} = \min[0 + \infty, 1 + 1, 3 + \infty, 6 + 5, 5 + \infty, 2] = 2$

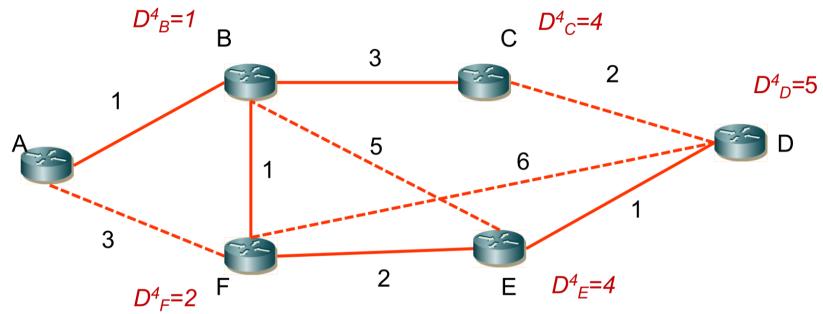


Figure 4.26: Esempio di applicazione dell'algoritmo di Bellman-Ford - Passo 5

Poi si calcolano i costi dei cammini quando  $h = 5$ , quindi con al massimo un salto.

- $D_B^{(5)} = \min[0 + 1, 3 + 3, 6 + \infty, 5 + 5, 2 + 1, 1] = 1$
- $D_C^{(5)} = \min[0 + 3, 1 + 3, 6 + \infty, 5 + 2, 2 + \infty, 3] = 3$
- $D_D^{(5)} = \min[0 + \infty, 1 + \infty, 3 + \infty, 5 + 1, 2 + 5, 6] = 6$
- $D_E^{(5)} = \min[0 + \infty, 1 + 5, 3 + 2, 6 + 1, 2 + \infty, 5] = 5$
- $D_F^{(5)} = \min[0 + \infty, 1 + 1, 3 + \infty, 6 + 5, 5 + \infty, 2] = 2$

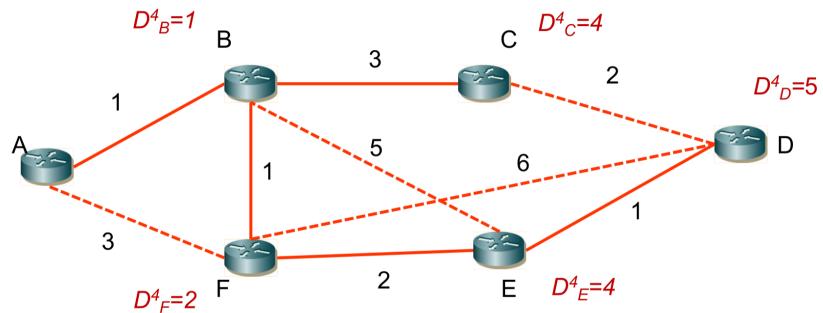


Figure 4.27: Esempio di applicazione dell'algoritmo di Bellman-Ford - Passo 6

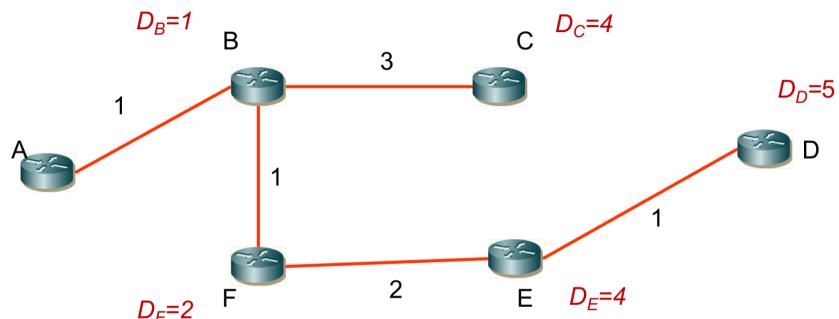


Figure 4.28: Albero dei cammini a costo minimo

#### 4.2.6 Link State e Dijkstra

L'algoritmo link-state rappresenta un'alternativa al vistance vector, si utilizza per reti di dimensioni maggiori.

Il presupposto su cui si basa questo algoritmo è che i router conoscano la topologia della rete, non come avveniva nel distance vector; infatti nel distance vector i router conoscevano solamente le info sui vicini, nel link state invece i router conoscono l'intera topologia della rete.

**Flooding** Si sfrutta il concetto di flooding: il flooding è un protocollo di instradamento usato dai router che inoltrano un pacchetto in ingresso su tutte le linee ad eccezione di quella da cui proviene. Ogni pacchetto in arrivo viene inoltrato su ogni linea di uscita eccetto quella da cui è arrivato; quindi tutti i router della rete comunicano in flooding tutte le informazioni dei vicini, con relativi costi. Inoltre il link state periodicamente ritrasmette l'informazione.

La topologia della rete è cosa nota a tutti e quindi si possono applicare strategie proprie della teoria dei grafi, più nello specifico applichiamo l'algoritmo di Dijkstra (complessità non lineare,  $n \log(n)$ , ossia con l'aumentare dei nodi la complessità di questo algoritmo aumenta alla  $n \log(n)$ ): permette di trovare la strada nel modo più efficiente possibile (considerando tempo e costo: migliore end to end).

#### Algoritmo di Dijkstra

A differenza dell'algoritmo di Bellman-Ford, quello di Dijkstra viene applicato dai router della rete; ad ogni cambiamento della topologia (o dopo un certo tempo) vengono calcolati i cammini minimi.

Considerazioni preliminari:

- $s$  è la sorgente, ossia il router da cui si calcolano i cammini minimi usando Dijkstra
- $N$  è l'insieme dei nodi della rete
- $M$  è l'insieme dei nodi dell'albero corrente
- $V(M)$  è l'insieme dei nodi raggiungibili con un solo salto dai nodi dell'albero  $M$
- $D_j$  è il costo del cammino a costo minimo, da  $s$  al nodo  $j$
- $d_{ij}$  è il costo del collegamento diretto tra i nodi  $i$  e  $j$ ;  
se il collegamento non è diretto allora questo costo è infinito

Steps dell'algoritmo:

1. Inizializzazione: considero l'insieme  $M$  pari ad  $s$ ;

nel caso in cui il nodo di cui desidero calcolare il cammino sia collegato direttamente ad  $s$ , quindi  $j$  fa parte dell'insieme  $V(M)$ , allora il costo del cammino minimo sarà:

$$M = \{s\} \quad D_j = \begin{cases} d_{sj}, & \forall j \in V(s) \\ \infty, & \text{altrimenti} \end{cases}$$

2. prendo in considerazione i nodi che fa parte dell'insieme  $V(M)$ , perciò i nodi vicini a  $M$ , raggiungibili con un solo salto; scelgo un  $k$  nodo che sia il più vicino all'albero  $M$  rispetto gli altri nodi vicini, "il più vicino tra i vicini": aggiungo  $k$  all'albero  $M$  e ne calcolo la distanza per arrivare al nodo  $j$ , e la aggiungo nell'insieme con i cammini  $D_j$

$$\text{Select } k \in V(M) \mid D_k = \min_{i \in V(M)} \{D_i\} \quad M = M \cup \{k\}$$

$$D_j = \min\{D_j, D_k + d_{kj}\} \quad \forall j \in V(M)$$

3. si prosegue con il passaggio svolto in precedenza (ricerca del vicino da aggiungere all'albero M) fino a quando non ho ottenuto tutti i cammini minimi, perciò fino a quando l'albero M non sia pari alla rete stessa:

If  $M = N$  stop,  
altrimenti vai al passo 2.

### Esempio Dijkstra

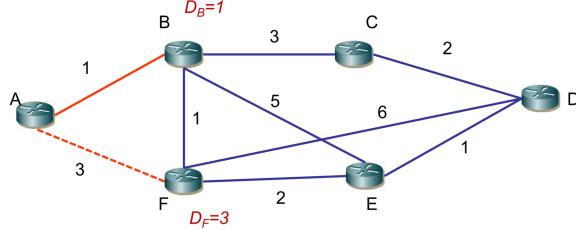


Figure 4.29: Esempio Link State

### Passo 1 dell'algoritmo di Dijkstra

$$M = \{A\}, \quad V(M) = \{B, F\}$$

$$D_B = D_A + d_{AB} = 0 + 1 = 1$$

$$D_F = D_A + d_{AF} = 0 + 3 = 3$$

B è il nodo diretto più vicino:

$$\min\{D_B, D_F\} = \min\{1, 3\} = 1 = D_B \implies$$

$$M = M \cup B; \quad V(M) = \{C, E, F\}$$

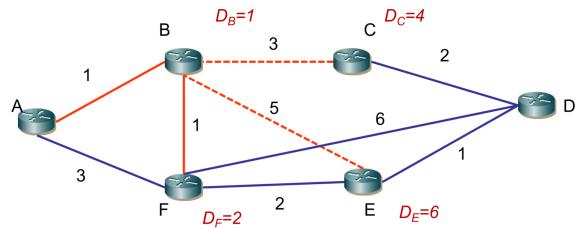


Figure 4.30: Esempio Link State - Passo 2

### Passo 2 dell'algoritmo di Dijkstra

Ora  $M = \{A, B\}$ ,  $V(M) = \{C, E, F\}$ .

$$D_C = \min(D_C, D_B + d_{BC}) = \min(\infty, 1 + 3) = 4$$

$$D_E = \min(D_E, D_B + d_{BE}) = \min(\infty, 1 + 5) = 6$$

$$D_F = \min(D_F, D_B + d_{BF}) = \min(3, 1 + 1) = 2$$

Il nodo più vicino è  $F$  ( $D_F = 2$ ), quindi  
 $M = \{A, B, F\}$ .

### Passo 3 dell'algoritmo di Dijkstra

$$M = \{A, B, F\}, \quad V(M) = \{C, D, E\}$$

$$D_C = \min\{D_C, D_F + d_{FC}\} = \min\{4, 2 + \infty\} = 4$$

$$D_D = \min\{D_D, D_F + d_{FD}\} = \min\{\infty, 2 + 6\} = 8$$

$$D_E = \min\{D_E, D_F + d_{FE}\} = \min\{6, 2 + 2\} = 4$$

Il nodo più vicino è  $C$  o  $E$  (entrambi con costo 4). Si può scegliere uno dei due, ad esempio  $E$ .

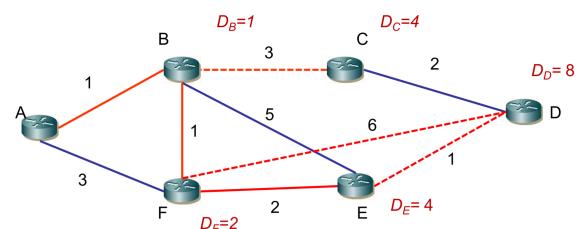


Figure 4.31: Esempio Link State - Passo 3

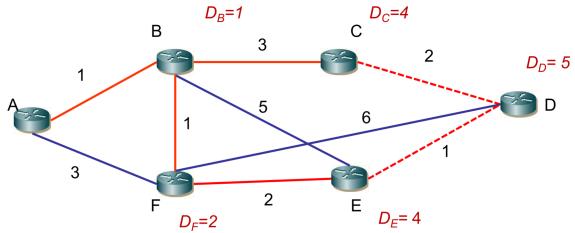


Figure 4.32: Esempio Link State - Passo 4

#### Passo 4 dell'algoritmo di Dijkstra

Ora  $M = \{A, B, F, E\}$ ,  $V(M) = \{C, D\}$ .

$$D_C = \min\{D_C, D_E + d_{EC}\} = \min\{4, 4 + \infty\} = 4$$

$$D_D = \min\{D_D, D_E + d_{ED}\} = \min\{8, 4 + 1\} = 5$$

Il nodo più vicino è  $C$  ( $D_C = 4$ ), quindi

$M = \{A, B, C, E, F\}$ .

#### Passo 5 dell'algoritmo di Dijkstra

Ora  $M = \{A, B, C, E, F\}$ ,  $V(M) = \{D\}$ .

$$D_D = \min\{D_D, D_C + d_{CD}, D_E + d_{ED}\} =$$

$$\min\{5, 4 + 2, 4 + 1\} = 5$$

L'albero dei cammini  $M$  è stato completato:  $M = N$

$$M = \{A, B, C, D, E, F\}, \quad V(M) = \emptyset$$

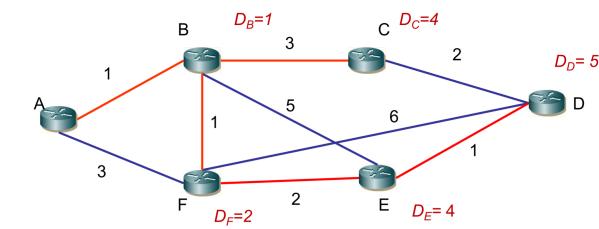


Figure 4.33: Esempio Link State - Passo 5

### Albero dei cammini

Il risultato è identico a quello ottenuto con Bellman-Ford

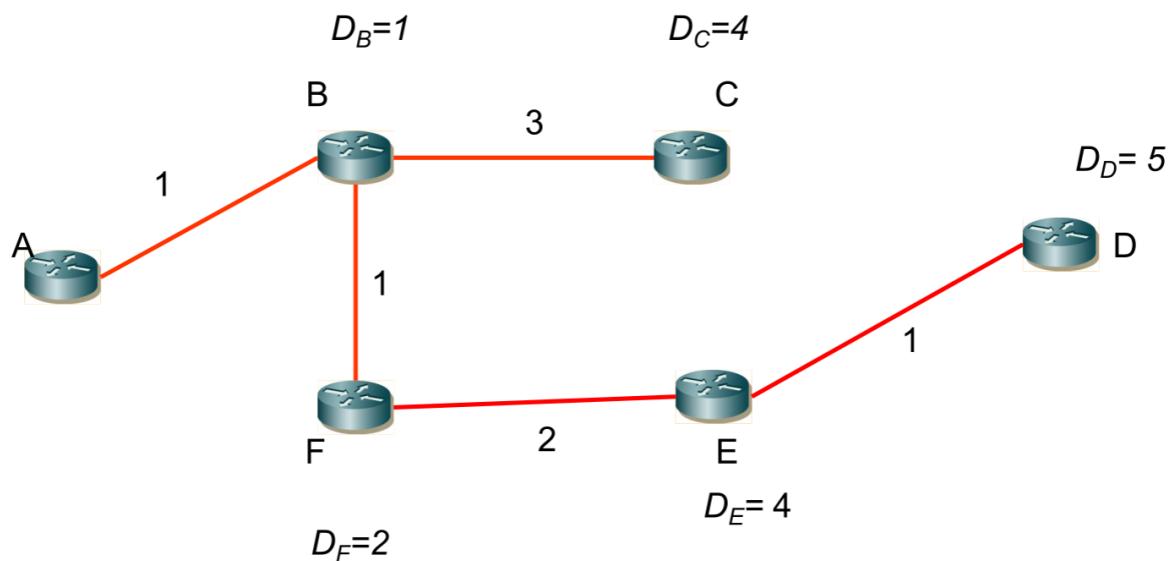


Figure 4.34: Albero dei cammini

#### 4.2.7 Problemi di routing - routing loop

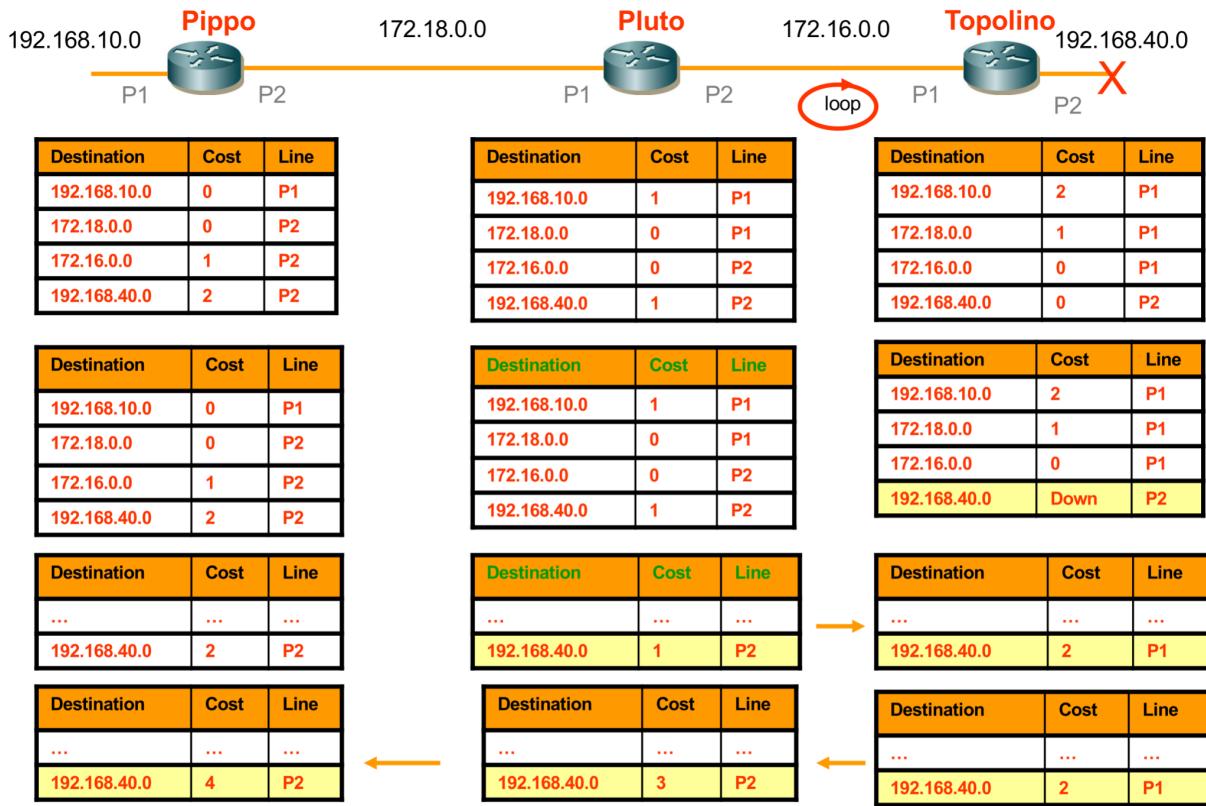


Figure 4.35: Esempio di routing loop

#### Soluzioni: split horizon, poison reverse, hold-down

- Tramite split horizon decido che i router non condividono con i vicini le informazioni ottenute da altri vicini
- split horizon con poisoned reverse, invece di non comunicare ai vicini le info di altri vicini, glielo comunico con costo infinito, metrica “infinity”, così che non vengano mai scelte poiché irraggiungibili
- trigger update: ogni volta che avviene un cambiamento nella topologia della rete allora vengono aggiornate le tabelle in base a questo cambiamento le reti vengono dichiarate irraggiungibili non nell'esatto momento in cui non sono più raggiungibili, ma allo scadere di un time “hold-down timer”

#### 4.2.8 Protocolli di routing intra AS (RIP - OSPF)

##### RIP (Distance Vector)(UDP)

**Caratteristiche principali** Il RIP è il protocollo di routing più semplice ed è adatto a reti di piccole dimensioni; usa il Distance Vector.

Come metrica del costo utilizza l'hop count, ossia valuta il costo dei percorsi in base al numero dei santi che vengono effettuati. Il valore massimo di hop è 15, perciò quando questo valore arriva a 16 per un certo percorso, questo viene indicato come irraggiungibile (per evitare il counting tot infinity). Il routing update (aggiornamento delle tabelle contenenti le informazioni per il routing) avviene ogni 30s oppure ad ogni cambiamento topologico.

**RIPv1 e RIPv2** Nel RIPv1 l'indirizzamento IP era classful, perciò i pacchetti scambiati non contenevano informazioni sulla maschera di rete; quindi come destinazione si scriveva solamente l'ind. IP.

RIPv2 invece ha integrato anche la riga Subnet Mask, così da permettere l'uso degli indirizzi classless.

I messaggi RIP vengono incapsulati all'interno di UDP (porta 520), perciò tramite datagrammi. Ognuno di questi pacchetti può contenere al massimo 25 destinazioni.

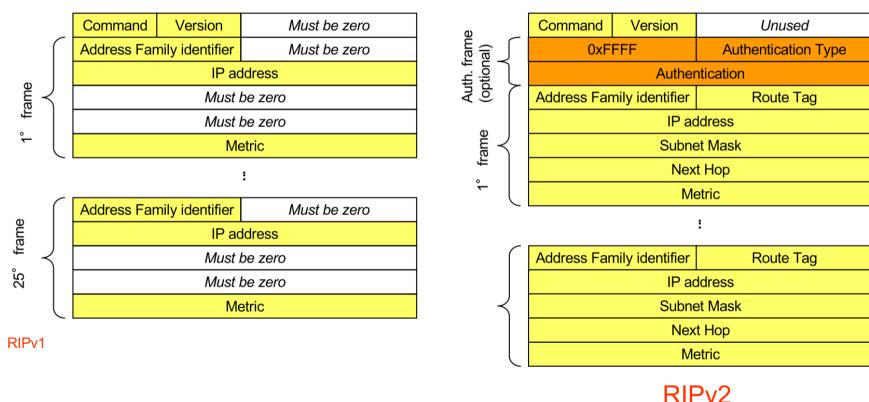


Figure 4.36: Confronto tra RIPv1 e RIPv2

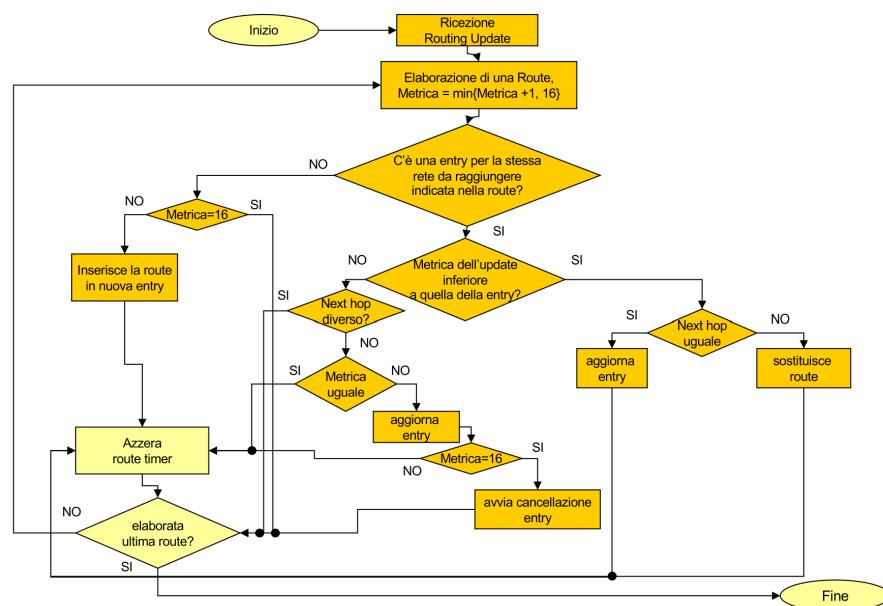


Figure 4.37: Processo di aggiornamento in RIP

## OSPF (Link State) e Link State Advertisement (LSA)

Un protocollo che utilizza il link state come alg di routing è il OSPF(open shortest path first); in OSPF si può specificare quale sia il costo(banda, affidabilità, ritardo, anche un valore inserito manualmente).

Quando i router diventano tanti allora OSPF non fa altro che dividere gli AS in aree, ognuna delle quali definita da 4 ottetti, in modo gerarchico;

**Area border routers** i router al confine con due aree vengono detti “area border router” e devono avere almeno un’interfaccia verso la backbone area(rete 0.0.0.0) questa backbone area(area 0) fa da ponte tra le aree. L’idea è quella di usare l’algoritmo dijkstra in ogni singola area e non sul totale, così da richiedere meno dispendio compilativo.

**LSA e pacchetti LSP** I Link State Packet: sono pacchetti generati dal protocollo OSPF, contengono i vari nodi a cui il router è collegato, informazione contenuta nel campo LSA(Link State Advertisement).

Nella rete locale LAN è importante definire un router che inoltri i pacchetti LSP, questo router viene chiamato “designated” router.

Per ridurre l’overhead(risorse richieste in sovrappiù rispetto a quelle strettamente necessarie) da segnalazione, l’LSP viene inviato ogni 30 minuti o ad ogni qual volta ci sia un cambiamento dei collegamenti della rete.

**Tipologie di pacchetti LSP** I pacchetti LSP sono utilizzati per scambiare informazioni tra i router OSPF e sono di tre tipi:

- link state request: richiesta di invio di uno o più LSA
- link state update: ogni 30 min o quando avviene un cambiamento topologico
- link state acknowledgment: all'avvenuta ricezione del LSA

**Tipologie di LSA** In OSPF esistono cinque tipologie di LSA, che vengono utilizzate per scambiare informazioni sulla topologia della rete tra i router OSPF:

- router LSA
- Network LSA
- Summary LSA
- AS Boundary
- router LSA
- AS external LSA

! un LSA “vecchio” di circa un’ora viene rimosso

Ogni router conserva gli LSA più recenti

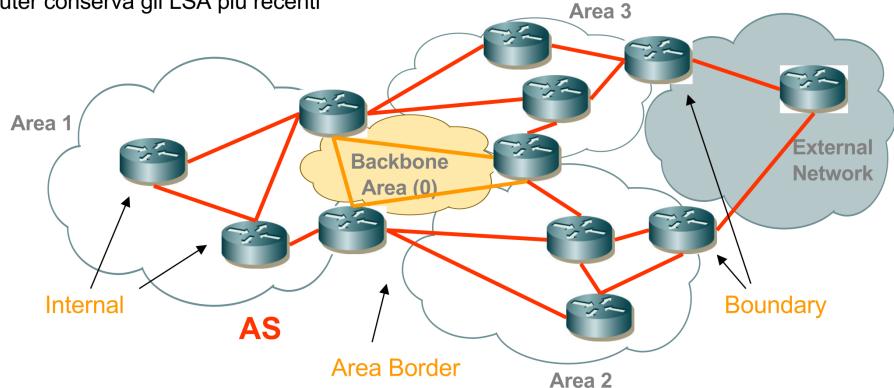


Figure 4.38: Backbone della rete OSPF

**Differenza tra LSA e pacchetto OSPF** LSA è informazione vera e propria dello stato dei link. Il pacchetto OSPF invece trasporta una o più LSP tra router che utilizzano OSPF.

**Autenticazione** OSPF supporta l'autenticazione dei pacchetti, per garantire che i pacchetti OSPF siano inviati da router autorizzati e non siano stati manomessi. Si distinguono due tipi di autenticazione:

- **Autenticazione del peer:** verifica che il pacchetto sia inviato da un router autorizzato
- **Autenticazione del messaggio:** verifica l'integrità del messaggio stesso, assicurandosi che non sia stato alterato durante la trasmissione

**Pacchetto OSPF** Il pacchetto OSPF è encapsulato in pacchetti IP(type 59, in esadecimale).

Del pacchetto OSPF mi serve sapere: version, Router ID(IP address del router), Area ID, link state checksum.

**Utilizzo di Dijkstra in diverse aree** Ad esempio, nell'area 1 uso il dijkstra tra quei 4 router, i router di bordo area avranno sicuramente dei collegamenti con le altre aree, però durante il calcolo dijkstra in area 1 questi altri collegamenti “terminano” sui router di bordo area. questo ragionamento viene applicato ad ogni area, anche alla backbone area, che mi va a risolvere il fatto che i router di bordo area precedentemente non integravano nel dijkstra i collegamenti fuori dalla propria area. così ho calcolato le strade più efficienti di ogni area, il che porta a vantaggi anche nel quadro globale della cosa. ci sono anche router detti boundary router che si collegano con “internet”, con il “mondo esterno”, infatti tutte le aree precedenti fanno parte della stessa rete, è divisa solo in aree. così abbasso il costo computazionale dell'algoritmo dijkstra perchè non lo applico a tutta la rete, ma a tutte le aree che compongono la rete, poichè è un algoritmo la cui complessità è  $n \log(n)$ , perciò meglio usarlo più volte con un numero inferiore di nodi( $n$ ) che una sola volta con un numero elevato di  $n$

## Pacchetti OSPF

version	packet type	Packet Length
Router ID (IP address)		
Area ID		
Checksum	Auth. Type	
Authentication Data		
Link state age	options	Link state type
Link state ID		
Advertising router		
Link State sequence number		
Link state checksum	Length	
Link state Data		

Figure 4.39: Pacchetto OSPF

#### 4.2.9 Protocolli di routing inter-AS - (BGP)

Il BGC(border gateway protocol) è il protocollo che gestisce il routing tra AS diversi, è montato sui router di frontiera degli AS.

Esistono tre tipologie di Autonomous Systems:

- Stub AS: ha un solo router di frontiera, comunica con un solo AS; trasporta solo traffico in uscita o in entrata, ma non entrambi
- Multihomed AS: può avere più router di frontiera, comunica con più AS, ma non fa da transit per altri AS
- transit AS: ha più router di frontiera, comunica con più AS e fa da transit per altri AS

Questo protocollo prevede che i router comunichino a coppie tra di loro utilizzando il TCP(porta 179); il router di bordo della AS comunica con gli altri router di bordo delle altre AS.

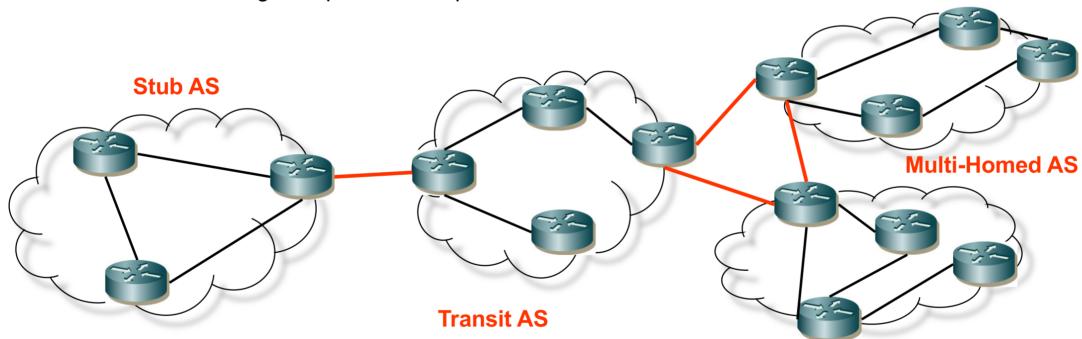


Figure 4.40: Tipologie di Autonomous Systems

Lo scambio di informazioni BGP può avvenire anche tra router interni dello stesso AS; è possibile distinguerli:

- **EBGP**: External BGP, scambio di informazioni tra router di frontiera di AS diversi
- **IBGP**: Internal BGP, scambio di informazioni tra router di frontiera dello stesso AS

Ogni AS ha un identificativo unico, l'Autonomous System Number(ASN).

#### Approccio Path Vector

Il BGP utilizza un approccio path vector, in cui ogni router mantiene una tabella di routing che contiene le informazioni sui percorsi verso le destinazioni e i rispettivi attributi. Queste informazioni vengono scambiate tra i router BGP per costruire una visione globale della rete e determinare il percorso migliore per raggiungere le destinazioni.

**Attributi BGP** Gli attributi BGP sono informazioni associate ai percorsi che vengono utilizzate per prendere decisioni di routing. Alcuni degli attributi più comuni includono:

- **AS Path**: elenca gli Autonomous Systems attraversati dal percorso
- **Next Hop**: indica il prossimo router da raggiungere per il percorso

Tra le diverse rotte si sceglie quella con AS-PATH più breve; in caso di rotte con stesso costo, si sceglie quella con NEXT-HOP più vicino; in ultima istanza ci si basa sugli altri attributi in relazione alla policy adottata.

Inoltre un router può accettare o meno gli annunci che gli giungono in base alle sue policy.

# Chapter 5

## Livello 2 Data link

### 5.1 Strategie di controllo errore

I pacchetti di livello 2 sono chiamati frame/trame, sono divisi in header e trailer; nel trailer sono presenti i campi utili alle strategie di controllo degli errori.

È possibile distinguere due tipi principali di strategie:

1. FEC (Forward Error Correction): il mittente invia dati ridondanti, in modo che il destinatario possa correggere gli errori senza richiedere una ritrasmissione.
2. ARQ (Automatic Repeat-reQuest): il mittente invia i dati e il destinatario richiede la ritrasmissione dei dati corrotti.

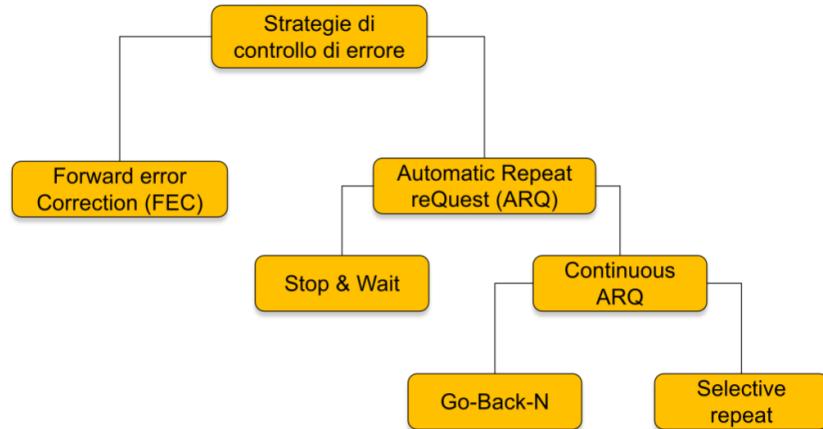


Figure 5.1: Strategie di controllo degli errori: confronto tra FEC e ARQ

#### 5.1.1 Protocolli ARQ (Automatic Repeat-reQuest)

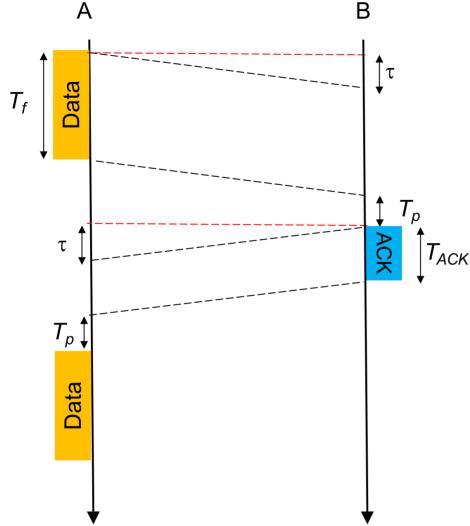
Nei sistemi di telecomunicazioni, Automatic Repeat-reQuest è una strategia di controllo di errore, che svolge il compito di rivelare un errore (ma non di correggerlo). I pacchetti corrotti vengono scartati e viene richiesta la loro ritrasmissione. I protocolli di tipo ARQ più comuni sono:

- Stop-and-Wait
- Go-Back-N
- Selective Repeat

## Stop-and-Wait

Lo stop-and wait è un protocollo di tipologia Continous ARQ, per cui è ammessa la trasmissione continua di più frame numerati.

In questo caso specifico, i segnali di riscontro negativi sono cumulativi.



Il protocollo Stop-and-Wait prevede che il mittente attenda il riscontro del destinatario per ogni frame inviato, prima di procedere con il frame successivo.

Il mittente invia una trama, quindi si ferma (stop) e attende (wait) il riscontro (Ack) da parte del ricevente prima di trasmettere una nuova trama.

- $T_f$ : tempo di trasmissione di una trama [s]
- $T_{ACK}$ : tempo di trasmissione di un riscontro [s]
- $T_p$ : tempo di elaborazione di una trama [s]
- $\tau$ : tempo di propagazione [s]

Figure 5.2: Schema Stop-and-Wait

### 5.1.2 Efficienza di stop and wait in assenza di errori

Prima di definire l'efficienza del protocollo, è necessario definire il tempo necessario a trasmettere una trama:

$$T_{tot} = T_f + T_{ACK} + 2T_p + 2\tau \quad (5.1)$$

$T_p$  come già detto è il tempo di elaborazione della trama, dipende dalla velocità di elaborazione e quindi non dovrebbe essere precisamente uguale in trasmissione e ricezione, ma per semplicità lo consideriamo uguale. Inoltre  $T_f$ , il tempo di trasmissione della trama, è anche definito come il tempo utile, poiché tutti gli altri tempi che consideriamo sono relativi a operazioni di controllo, elaborazione e invio del pacchetto. L'efficienza ( $\eta$ ) è perciò definita come:

$$\eta = \frac{T_f}{T_{tot}} = \frac{T_f}{T_f + T_{ACK} + 2T_p + 2\tau} \quad (5.2)$$

Si possono effettuare alcune semplificazioni, considerando che  $T_{ACK}$  e  $T_p$  sono molto più piccoli di  $T_f$  e  $\tau$ , quindi possiamo considerare:

$$T_{tot} = T_f + 2\tau \quad (5.3)$$

da cui si ottiene l'efficienza:

$$\eta = \frac{T_f}{T_f + 2\tau} = \frac{1}{1 + 2\frac{\tau}{T_f}} = \frac{1}{1 + 2\alpha} \quad (5.4)$$

dove  $\alpha = \frac{\tau}{T_f}$  è il rapporto di propagazione normalizzato

- il throughput è pari al prodotto tra la frequenza di cifra C (capacità del canale) e l'efficienza:  $\eta C$
- anche in assenza di errori, l'efficienza non è mai pari a 1.

### 5.1.3 Efficienza di stop and wait in presenza di errori

I problemi che possono verificarsi sono:

- il frame o l'ACK arrivano errati
- il frame non arriva

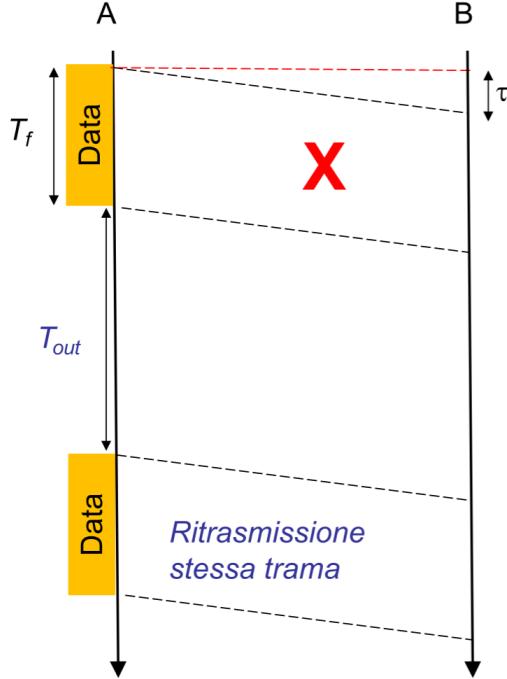


Figure 5.3: Errore nel frame: quando si verifica un errore nel frame, il mittente si accorge del problema grazie alla scadenza del timer avviato durante la trasmissione. In tal caso, il frame viene ritrasmesso. Questo processo garantisce che i dati vengano ricevuti correttamente, ma può ridurre l'efficienza del protocollo a causa delle ritrasmissioni necessarie.

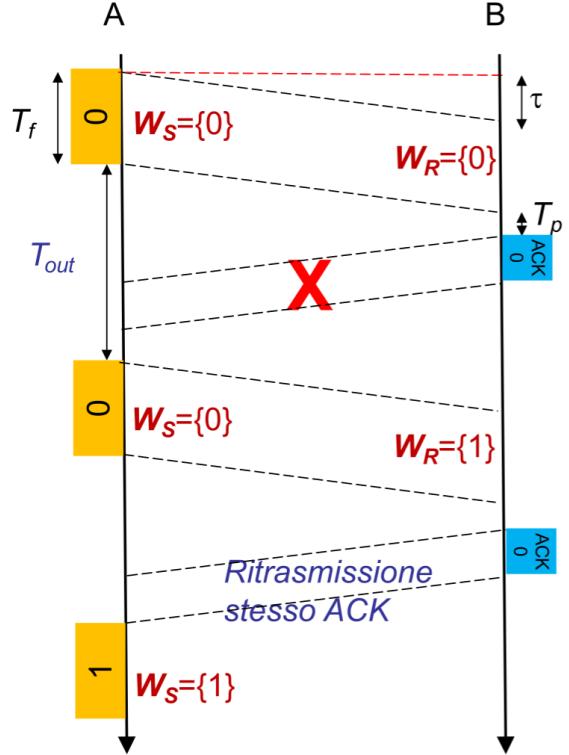


Figure 5.4: Errore nell'ACK: quando si verifica un errore nell'ACK, il mittente non riceve il riscontro atteso entro il tempo stabilito dal timer. Anche in questo caso, il frame viene ritrasmesso.

Come calcolo il  $T_{out}$ (time che avvio nel momento nella trasmissione del frame)?

Deve valere almeno il tempo di invio del ACK( $T_{ACK}$ ), più il tempo di propagazione(andata e ritorno) più il tempo di elaborazione del frame( $T_p$ , in ricezione e trasmissione):

$$T_{out} \geq T_{ACK} + 2\tau + 2T_p \quad (5.5)$$

Il numero di frame ritrasmessi per la corretta ricezione del frame è detto  $N_S$  ed è in media considerabile come una variabile aleatoria;

inoltre il numero totale di frame trasmessi( $N_R$ ) è dato dal numero di frame ritrasmessi più il frame inviato con successo:

$$N_R = N_S + 1 \quad (5.6)$$

Per calcolare il tempo totale necessario per la trasmissione si deve considerare che per ogni ritrasmessione si perde un tempo pari a  $T_f + T_{out}$  cui si deve aggiungere il tempo necessario per la trasmissione che avviene con successo:

$$T_{tot} = (N_S - 1)(T_f + T_{out}) + T_f + T_{ACK} + 2T_p + 2\tau \quad (5.7)$$

**Calcolo di  $\overline{N_S}$**  la variabile  $N_S$  è aleatoria, in quanto dipende dagli errori sul canale di trasmissione; si calcola l'efficienza media del protocollo SW in presenza di errori considerando il valor medio di  $N_S$

$$\begin{aligned} N_S = 1 & \quad pr\{N_S = 1\} = 1 - P \\ N_S = 2 & \quad pr\{N_S = 2\} = P(1 - P) \\ N_S = 3 & \quad pr\{N_S = 3\} = P^2(1 - P) \\ \vdots & \quad \vdots \\ N_S = i & \quad pr\{N_S = i\} = P^{i-1}(1 - P) \end{aligned}$$

Figure 5.5: Probabilità di successo e insuccesso nella trasmissione di un frame in Stop-and-Wait

Definiamo il frame error rate (FER) come la probabilità che un frame venga ricevuto con errore, che sarà il valore di  $P$  nel calcolo successivo. Se  $P$  è la probabilità che un frame sia errato, la probabilità che un frame sia trasmesso correttamente è  $1 - P$ .

La variabile aleatoria  $N_S$  rappresenta il numero di tentativi necessari affinché un frame sia ricevuto correttamente.

La sua distribuzione è di tipo geometrico:

$$P(N_S = k) = P^{k-1}(1 - P)$$

dove  $k$  è il numero di tentativi (ritrasmissioni più il primo invio). Il valor medio di  $N_S$  è:

$$\overline{N_S} = \frac{1}{1 - P}$$

$$\overline{N_S} = E[N_S] = \sum_{i=1}^{+\infty} iP^{i-1}(1 - P) = \frac{1}{(1 - P)^2}(1 - P) = \frac{1}{1 - P}$$

**Spiegazione** La formula sopra mostra il calcolo del valor medio  $\overline{N_S}$  per una variabile aleatoria geometrica, che rappresenta il numero medio di tentativi necessari affinché un frame venga trasmesso correttamente.

La distribuzione geometrica è appropriata perché ogni trasmissione è indipendente e la probabilità di successo (ricezione corretta) è costante e pari a  $1 - P$ , dove  $P$  è la probabilità di errore del frame. La somma  $\sum_{i=1}^{+\infty} iP^{i-1}(1 - P)$  rappresenta la media pesata di tutti i possibili numeri di tentativi, ciascuno moltiplicato per la probabilità che siano necessari esattamente  $i$  tentativi.

Sviluppando la somma, si ottiene che il valor medio è  $\frac{1}{1-P}$ : questo significa che, in media, il numero di tentativi cresce all'aumentare della probabilità di errore  $P$ . Se  $P$  è vicino a zero, bastano pochi tentativi; se  $P$  aumenta, il numero medio di ritrasmissioni cresce rapidamente.

Perciò, ripetendo, se si vuole ottenere il valore medio di una variabile aleatoria bisogna effettuare uno studio probabilistico; questa probabilità aleatoria sarà in media pari alla moltiplicazione tra due probabilità:

- probabilità di successo ( $1-P$ ): frame trasmesso correttamente
- $P^{i-1}$  rappresenta la sequenza di fallimenti che avvengono prima della trasmissione avvenuta con successo, che calcolo per un valore di  $i$  infinito (serie geometrica), caso peggiore (benchè irreale)

$$\Pr(N_S = i) = \underbrace{P^{i-1}}_{\# \text{ fallimenti}} \cdot \underbrace{(1 - P)}_{\text{successo finale}}$$

La distribuzione di probabilità è:

$$\Pr(N_S = i) = P^{i-1}(1 - P)$$

Il valore atteso si ottiene come:

$$E[N_S] = \sum_{i=1}^{\infty} i \cdot \Pr(N_S = i)$$

## Errore sul bit ed errore sul frame

**Frame error rate (FER)** è possibile definire le probabilità per le quali ci sia un errore sui bit(BER, bit error rate) all'interno di un range oppure, di maggiore interesse a questo livello di rete, la probabilità di errore su un range di frame, quindi il FER(frame error rate).

Per calcolarlo ci interessa definire:

- $L_f$ : lunghezza in bit del frame
- $L_{ACK}$ : lunghezza in bit del ACK

e sarà pari al valore  $P$  utilizzato nel calcolo probabilistico precedente, perciò:

$$P = FER = 1 - (1 - p)^{(L_f + L_{ACK})} \quad (5.8)$$

## Calcolo dell'efficienza

L'efficienza  $\eta$  del protocollo Stop and wait è data da:

L'efficienza  $\eta$  del protocollo Stop-and-Wait in presenza di errori si calcola come rapporto tra il tempo utile di trasmissione e il tempo totale medio necessario per trasmettere correttamente un frame, considerando anche le ritrasmissioni dovute agli errori:

$$\eta = \frac{T_f}{T_{tot}} \quad (5.9)$$

dove:

- $T_f$  è il tempo di trasmissione del frame (dato utile)
- $T_{tot}$  è il tempo totale per ogni tentativo di trasmissione (inclusi tempi di ACK, propagazione, elaborazione)

ricordando la formula di  $T_{tot}$

$$T_{tot} = (N_S - 1)(T_f + T_{out}) + T_f + T_{ACK} + 2T_P + 2\tau \quad (5.10)$$

è possibile effettuare qualche semplificazione, trascurando termini piccoli (come  $T_{ACK}, T_P, \tau$ ) rispetto al tempo di trasmissione; considerando inoltre il tempo minimo per  $T_{out}$  (paria  $2\tau$ ) (timer pari al tempo di propagazione di andata/ritorno del frame) per cui:

$$\eta \simeq \frac{T_f}{(N_S - 1)(T_f + 2\tau) + T_f + 2\tau} = \frac{T_f}{N_S(T_f + 2\tau)} = \frac{(1 - P)T_f}{T_f + 2\tau} \quad (5.11)$$

ricordando il rapporto di propagazione normalizzato  $\alpha$  ( $\alpha = \frac{\tau}{T_f}$ ), ottengo che l'efficienza:

$$\eta = \frac{1 - P}{1 + 2\alpha} \quad (5.12)$$

**Dipendenza dell'efficienza dal ritardo di propagazione normalizzato** Fissato il FER (dipendente dalla rumorosità del canale), l'efficienza dipende dal valore del ritardo di propagazione normalizzato

Considerando la distanza  $d$  e la velocità di propagazione  $v$  delle onde elettromagnetiche nel mezzo:

$$\alpha = \frac{\tau}{T_f} = \frac{d/v}{L_f/C} = \frac{dC}{vL_f}$$

Il protocollo Stop-and-Wait è efficiente (quasi  $1 - P$ ) solo per valori piccoli di  $\alpha$ , cioè:

- canali lenti ( $C$  basso)
- collegamenti a distanze ridotte ( $d$  basso)
- trame sufficientemente grandi ( $L_f$  alto, ma non troppo per non aumentare  $P$ )

### 5.1.4 Protocolli Continuous ARQ

I protocolli continuous ARQ, a differenza dello stop-and-wait, prevede l'invio di un'insieme di pacchetti(frame/trame) numerati, tramite sliding window. Vengono definite:

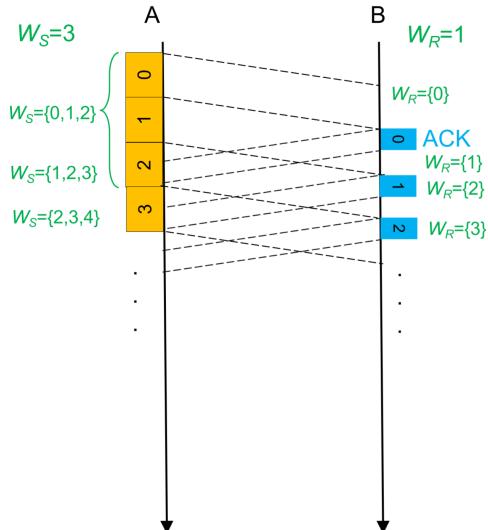
- finestra di trasmissione  $W_s$ : definisce la grandezza della finestra di ricezione, perciò quanti pacchetti posso inviare "consecutivamente"(on the fly, in attesa di riscontro) e da cui posso aspettarmi un riscontro (ACK) cumulativo
- finestra di ricezione  $W_R$ : definisce la grandezza della finestra di ricezione,

Se il campo dedicato alla numerazione è costituito da  $b$  bit, si potranno numerare in modo differente fino a  $N_b$  trame.

#### Protocollo Go-Back-N

**Senza errori - efficienza massima** Il protocollo Go-Back-N è un protocollo di tipo Continuous ARQ, che consente la trasmissione continua di più frame numerati. In questo caso specifico, i segnali di riscontro negativi sono cumulativi.

Nel caso in cui ci siano errori allora questo protocollo prevede di ritornare indietro nel momento della trasmissione in cui c'è stato l'errore(alla trama N), perciò go back N.



Nel seguente esempio la finestra di trasmissione è 3 e quella di ricezione 1; vengono riscontrati gli ack in modo corretto nei tempi previsti dalla finestra, che nel frattempo ad ogni ack corretto ricevuto scorre di posizione.

Questo vuol dire che nel caso in cui non ci siano errori con l'invio delle trame e con il riscontro degli ack, il seguente protocollo ha efficienza massima.

Figure 5.6: Go-Back-N senza errori

## Go back N - con errori

Si possono riscontrare errori durante la ricezione del pacchetto e durante il riscontro degli ack; a seconda della tipologia dell'errore avremo soluzioni differenti.

**Errore trasmissione del frame** utilizzo del negative ACK (NACK)

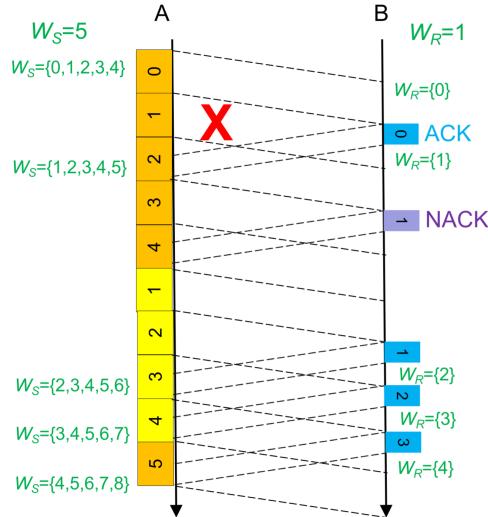


Figure 5.7: Go-Back-N con errori in trasmissione

Quando si verifica un errore in un frame, il destinatario scarta il frame errato e tutti i frame successivi.

Il mittente, una volta rilevato l'errore (tramite la ricezione di un NACK (negative ACK) inviato dal destinatario), ritrasmette il frame errato e tutti i frame successivi.

**Errore ACK non ricevuto ( $T_{out}$ )** caso in cui l'ACK non venga inviato o si sia perso

L'unico modo con il quale il mittente può evitare che il sistema si blocchi in caso di mancato riscontro (neanche NACK), è quello di avviare un timer ad ogni frame inviato, entro il quale va ricevuto un riscontro.

Se questo timer ( $T_{out}$ ) scade allora la finestra torna indietro e ritrasmette da dove è stato riscontrato il problema.

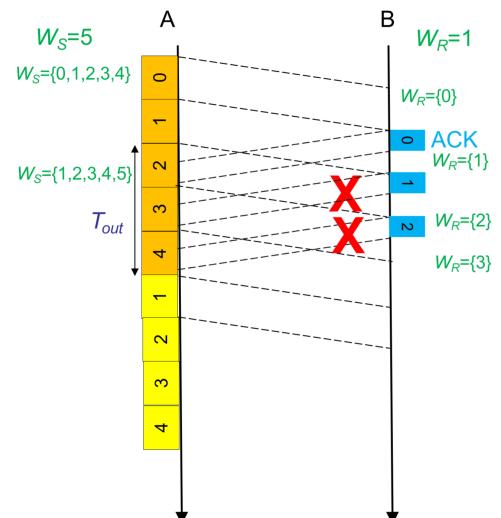


Figure 5.8: Go-Back-N con errore nell'ACK

## Go back N - ACK equivocato( $T_{out}$ )

con conseguente ricezione di duplicati

**Analisi del problema** Quando si verifica un equivoco nell'ACK, ossia viene riscontrato erroneamente un frame, il mittente interpreta erroneamente il riscontro ricevuto e potrebbe ritrasmettere frame già ricevuti correttamente dal destinatario. Questo equivoco può portare a un aumento delle ritrasmissioni e a una riduzione dell'efficienza del protocollo. L'unico ACK corretto è l'ACK-0, quelli successivi sono errati; nel frattempo la finestra di ricezione ha continuato a scorrere, richiedendo i nuovi 1, 2, etc... Il problema è che i pacchetti 1, 2, 3 sono già stati ricevuti, l'errore c'è stato solo negli ACK, per i quali il trasmittente sta reinviando 1, 2, 3 precedenti, che però risulteranno come una copia (il ricevente non lo sa) di quelli trasmessi correttamente prima ma riscontrati in modo errato.

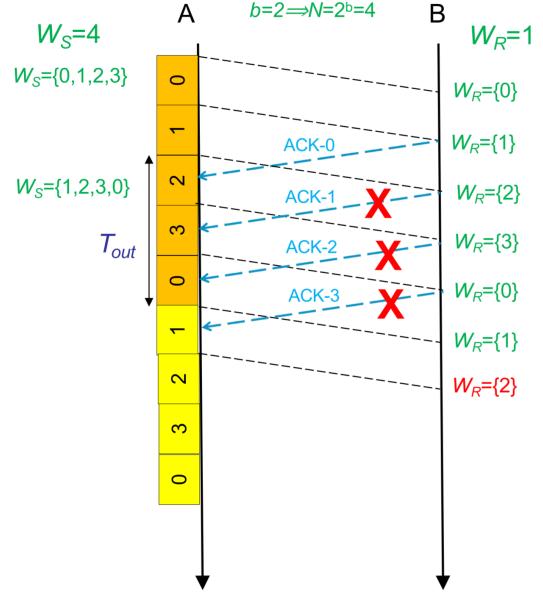


Figure 5.9: Go-Back-N con equivoco nell'ACK

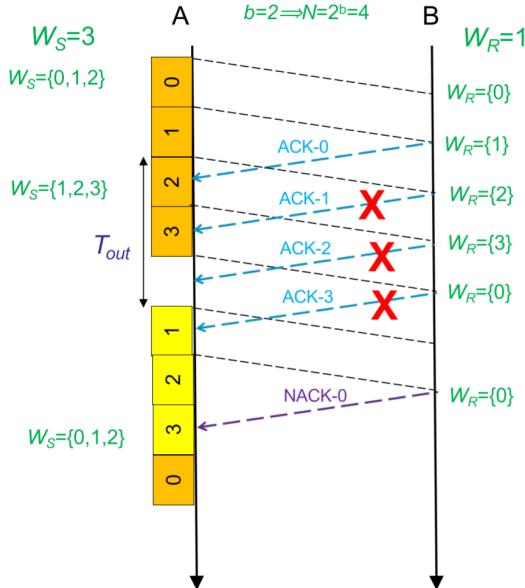


Figure 5.10: Go-Back-N con equivoco nell'ACK

**Soluzione** In questo esempio l'unico frame riscontrato correttamente è 0, come nel caso precedente; il problema dei duplicati si risolve poiché il primo frame che non ha avuto un riscontro corretto, ACK-1, ha avviato un timer entro il quale desidera ricevere un riscontro. Al termine di quest'ultimo il trasmittitore ritrasmette dal primo frame che non ha ricevuto riscontro, quindi da 1.

Il ricevente riceve 1, ma la sua finestra intata ha slideato come se non ci fossero problemi fino a 0; quindi il ricevente si accorge del problema poiché riceve 1 al posto di 0; a questo punto invia un NACK-0 per dire al trasmittitore che quelli ricevuti precedentemente erano corretti, ma riscontrati erroneamente, quindi desidera continuare la comunicazione dal frame 0, senza ricevere duplicati. ( $W_S + W_R \leq N$ ) Questo è possibile perché la finestra di trasmissione è grande  $N-1$ , dove  $N$  è il numero di frame.

## Protocollo selective repeat

Il protocollo Selective Repeat è un protocollo di tipo Continuous ARQ che consente la ritrasmissione selettiva dei soli frame ricevuti con errore, evitando di ritrasmettere quelli ricevuti correttamente.

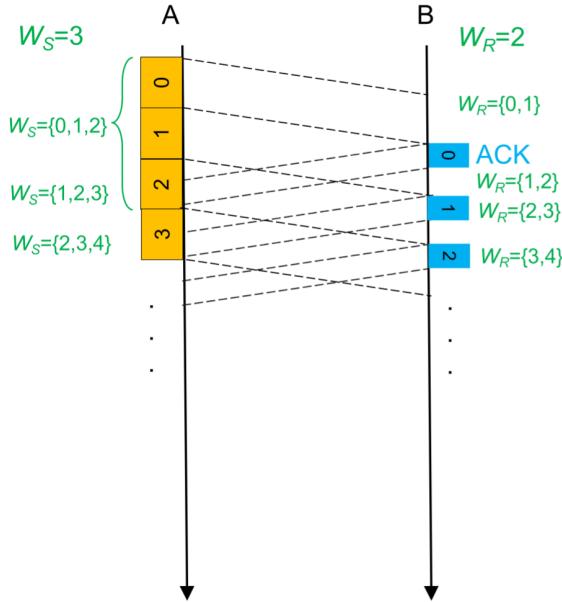


Figure 5.11: Selective Repeat

**In assenza di errori** Questo approccio migliora l'efficienza rispetto al Go-Back-N, poiché la finestra di ricezione può essere maggiore di 1 ( $W_R > 1$ ).

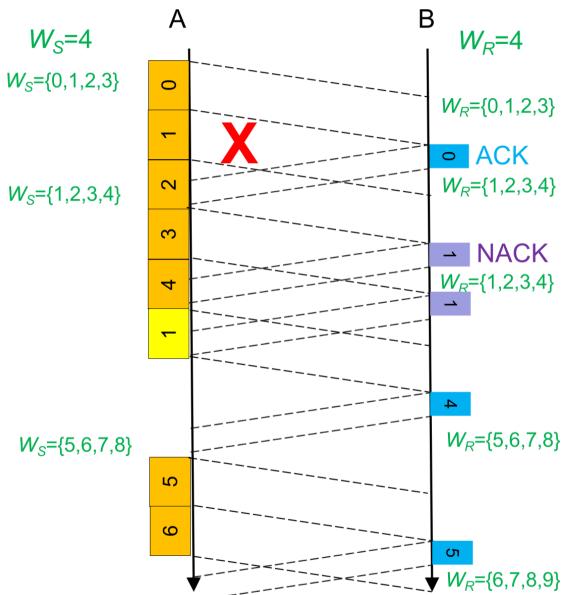


Figure 5.12: Selective Repeat con errori sulla trama

**In presenza di errori sul frame** Quando si verifica un errore in una trama, il protocollo Selective Repeat consente di ritrasmettere solo la trama errata (il ricevente invia un NACK in cui indica quale frame è andato perso), senza dover ritrasmettere tutte le trame successive come nel Go-Back-N. Questo approccio riduce il numero di ritrasmissioni necessarie e migliora l'efficienza complessiva del protocollo, specialmente in presenza di errori frequenti.

### Selective repeat - equivocazione dei riscontri e soluzione ( $T_{out}$ )

**Problema equivocazione ACK** Il problema sussiste quando i frame vengono trasmessi correttamente, la finestra di ricezione scorre correttamente, ma gli ACK inviati sono errati.

Perciò come avveniva erroneamente negli altri protocolli, vengono ricevuti dei frame duplicati perché il trasmettitore riceve degli ACK errati e quindi invia nuovamente i frame rispettivi (alla fine del  $T_{out}$ ) a questi ACK; ma questi frame risulteranno a tutti gli effetti dei duplicati di quelli ricevuti correttamente dal ricevitore.

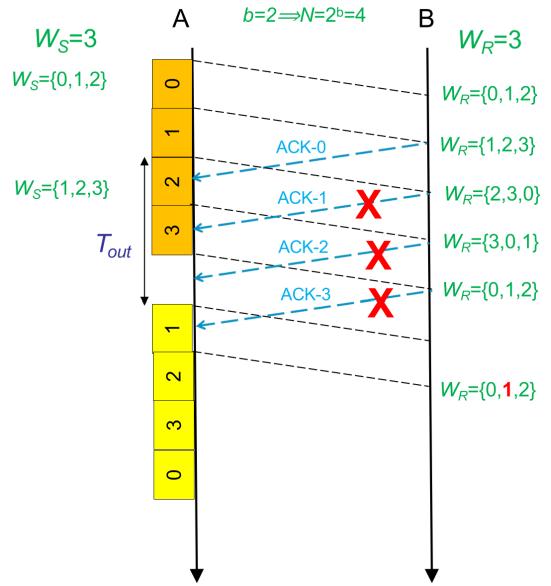


Figure 5.13: Selective Repeat con errori sull'ACK

**Soluzione** Questo problema viene risolto diminuendo la grandezza delle finestre ( $W_S + W_R \leq N$ ), secondo i criteri discussi nel capoverso successivo.

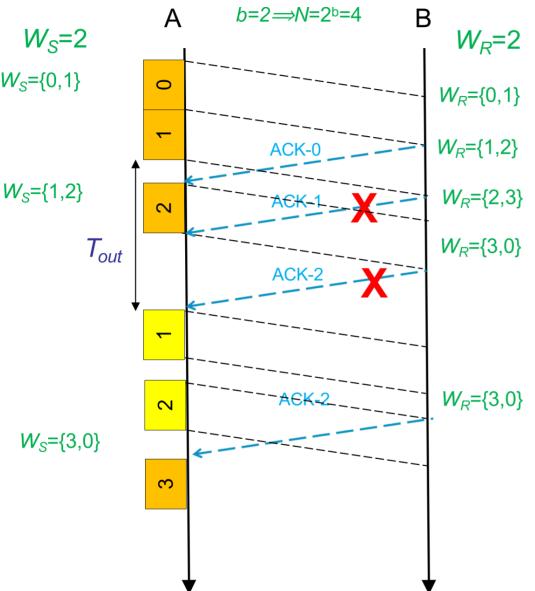


Figure 5.14: Selective Repeat con equivocazione risolta

### 5.1.5 Dimensione delle finestre di trasmissione/ricezione

Con la visione degli esempi riguardanti l'equivocazione, si può affermare che in entrambi i protocolli continuous ARQ, Go-Back-N e Selective Repeat, le finestre di trasmissione e di ricezione non possono essere pari a o maggiori di N in quanto si avrebbe equivocazione:

$$\begin{cases} W_S \leq N \\ W_R \leq N \end{cases} \quad W_S + W_R \leq N$$

Inoltre le finestre non devono sovrapporsi, poiché in caso di equivoco e di scorrimento completo della finestra di ricezione, si potrebbe riscontrare il problema di ricezione di duplicati; anche questo si risolve imponendo una condizione sulla grandezza di tali finestre:

#### Dimensione finestre GBN ed efficienza massima

Nel caso di GBN, la finestra di ricezione è necessariamente unitaria. Per cui la finestra di trasmissione deve essere al più N-1; poiché non conviene fissarla ad un valore inferiore al massimo (riduzione del throughput non giustificata), la scelta ottimale è fissare:

$$\begin{cases} W_S = N - 1 \\ W_R = 1 \end{cases}$$

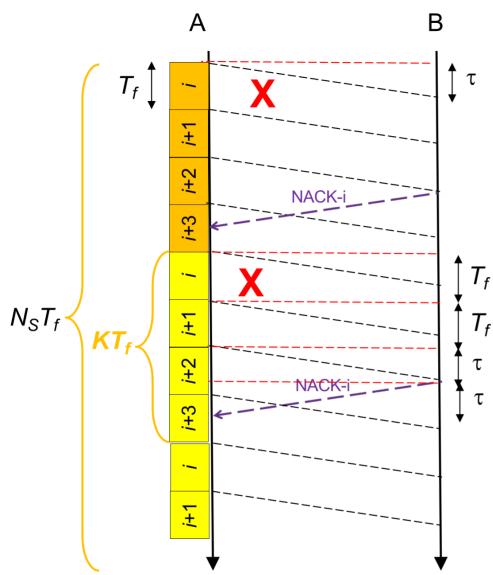
#### Dimensione finestre SR ed efficienza massima

Nel caso del selective repeat vanno considerati più fattori:

- non ha senso scegliere la finestra di trasmissione maggiore di quella di ricezione, perché si trasmetterebbero più trame di quante ne può ricevere e ordinare il ricevente
- non ha senso scegliere la finestra di ricezione maggiore di quella di trasmissione perché si ridurrebbe il throughput trasmettendo meno trame di quelle che si possono ricevere

Pertanto la soluzione ottimale è:  $W_S = W_R = \frac{N}{2}$

### 5.1.6 Efficienza GBN & SR



**Efficienza Go-Back-N** Il trasmittitore si accorge dell'errore sulla trama  $i$  solo dopo  $i+x$ , dove  $x$  dipende dalla finestra di trasmissione e dal tempo in cui impiega ad arrivare il NACK- $i$  relativo al frame errato.

Il costo della trasmissione del frame corretto è  $N_S T_f$ , dove  $N_S$  è il numero di frame che sono stati trasmessi e ritrasmessi per la corretta trasmissione di quel frame, pari a:

$$N_S = ("numero\_errori" * K) + 1 \quad (5.13)$$

K è il numero di trame che vengono ritrasmesse ad ogni errore, il + 1 il frame corretto.

Figure 5.15: Efficienza del protocollo Go-Back-N

Considerando il valore medio delle trame trasmesse in totale,  $N_S$ , il tempo totale equivale a:

$$T_{tot} = \overline{N_S} * T_f \quad (5.14)$$

da cui l'efficienza  $\eta$  è pari al reciproco di  $\overline{N_S}$ :

$$\eta = \frac{T_f}{\overline{N_S} \cdot T_f} = \frac{1}{\overline{N_S}} \quad (5.15)$$

$$\begin{array}{ll} N_S = 1 & pr\{N_S = 1\} = 1 - P \\ N_S = K + 1 & pr\{N_S = K + 1\} = P(1 - P) \\ N_S = 2K + 1 & pr\{N_S = 2K + 1\} = P^2(1 - P) \\ \vdots & \vdots \\ N_S = iK + 1 & pr\{N_S = iK + 1\} = P^i(1 - P) \end{array}$$

Figure 5.16: Calcolo probabilità di  $N_S$

Nella prima riga si illustra il caso in cui non ci siano errori, per cui  $numero\_errori * K + 1$  vale 1; nel secondo rigo l'errore è uno, nel terzo rigo gli errori sono due ( $numero\_errori = 2$ ) e così via, fino al calcolo generale di  $N_S = iK + 1$

Inoltre si calcola la probabilità pr

**Calcolo del valor medio di  $N_S$**  Il valor medio si otterrà sommando i valori assunti dalla variabile "i" per la probabilità di assumerli:

$$\begin{aligned} \overline{N_S} &= E[N_S] = \sum_{i=0}^{+\infty} (iK + 1)P^i(1 - P) = (1 - P) \left[ \sum_{i=0}^{+\infty} iKP^i + \sum_{i=0}^{+\infty} P^i \right] = \\ &= (1 - P) \left[ KP \sum_{i=1}^{+\infty} iP^{i-1} + \frac{1}{1 - P} \right] = (1 - P) \left[ \frac{KP}{(1 - P)^2} + \frac{1}{1 - P} \right] = \frac{1 + P(K - 1)}{1 - P} \end{aligned}$$

### Quanto vale K

Dal grafico sull'efficienza si può individuare il valore di K approssimato:

$$KT_f \simeq 2T_f + 2\tau \implies K = 2 + 2\alpha \quad (5.16)$$

sostituendo K nella formula dell'efficienza ottengo:

$$\eta = \frac{1 - P}{1 + P(K - 1)} = \frac{1 - P}{1 + P(1 + 2\alpha)} \quad (5.17)$$

### 5.1.7 Efficienza GBN & SR

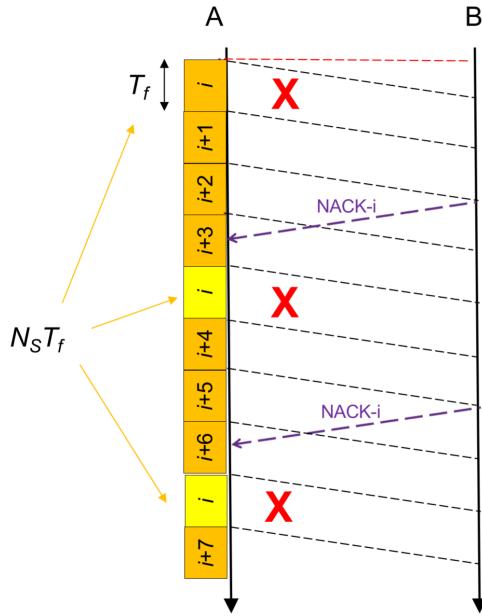


Figure 5.17: Efficienza del protocollo Selective Repeat

Per calcolare  $T_{tot}$  ho bisogno di individuare il valore medio di  $N_S$ :

$$\begin{aligned}
 N_S = 1 & \quad pr\{N_S = 1\} = 1 - P \\
 N_S = 2 & \quad pr\{N_S = 2\} = P(1 - P) \\
 N_S = 3 & \quad pr\{N_S = 3\} = P^2(1 - P) \\
 \vdots & \quad \vdots \\
 N_S = i & \quad pr\{N_S = i\} = P^{i-1}(1 - P)
 \end{aligned}$$

Figure 5.18: Calcolo del valor medio di  $N_S$  per Selective Repeat

**Efficienza Selective Repeat** Nel protocollo Selective Repeat, grazie alla ritrasmissione selettiva dei soli frame errati, l'efficienza risulta superiore rispetto al Go-Back-N, soprattutto in presenza di errori frequenti.

Per valutarne l'efficienza si suppone che:

- i riscontri arrivano all'interno della finestra di trasmissione
- l'eventuale trama sbagliata sia sempre la stessa
- si trascurano i tempi di elaborazione e di trasmissione degli ack( $\tau, T_P$ )

l'efficienza la calcoliamo allo stesso modo:

$$\eta = \frac{T_f}{T_{tot}} \quad (5.18)$$

$$\eta = \frac{T_f}{\overline{N_S} \cdot T_f} = \frac{1}{\overline{N_S}} \quad (5.19)$$

va capito quanto vale  $T_{tot}$

Nel protocollo Selective Repeat, il valore medio di  $N_S$  (numero di trasmissioni per frame corretto) si calcola considerando che ad ogni errore viene ritrasmesso solo il frame errato. La variabile aleatoria  $N_S$  segue una distribuzione geometrica come nel caso Stop-and-Wait, quindi:

$$\overline{N_S} = E[N_S] = \sum_{i=1}^{+\infty} i P^{i-1} (1 - P) = \frac{1}{1 - P}$$

dove  $P$  è la probabilità di errore sul frame. L'efficienza risulta quindi:

$$\eta = 1 - P$$

Si vede chiaramente che il SR ha l'efficienza più elevata tra tutti i protocolli di linea, ma è quello con la maggiore complessità (e quindi costi più alti) per la presenza dei due buffer (in trasmissione e ricezione) e per la necessità di gestire il riordinamento.

#### Go-Back-N

$$\eta_{GBN} = \frac{1 - P}{1 + P(1 + 2\alpha)}$$

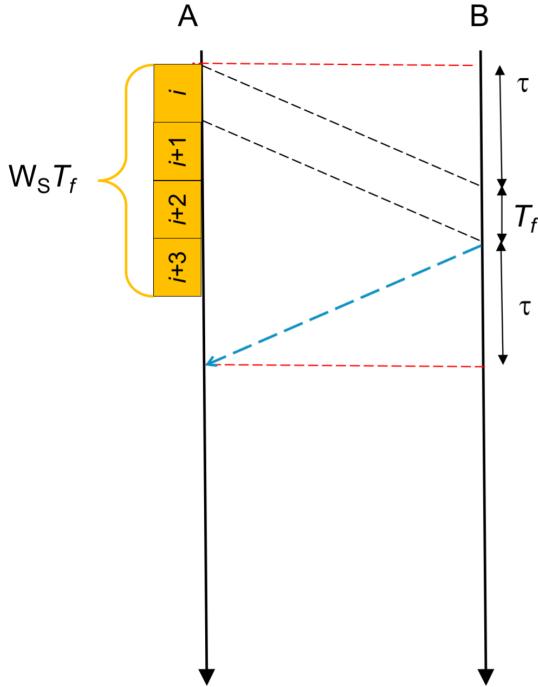
Quando nel caso ideale non si considerano gli errori ( $P=0$ ), GBN e SR hanno stessa efficienza pari a 1 (la massima teorica).

Si noti però che per valori piccoli del ritardo di propagazione normalizzato a tutti i protocolli hanno efficienza comparabile e pari a  $1-P$  per cui, in tal caso, sono equivalenti.

#### Selective Repeat

$$\eta_{SR} = 1 - P$$

### 5.1.8 Efficienza con riscontri fuori dalla finestra di trasmissione $W_s$



Fino ad ora ho studiato l'efficienza di questi protocolli nel caso in cui i riscontri arrivino all'interno della finestra di trasmissione; se però ciò non accade, quindi i riscontri arrivano fuori dalla finestra, l'efficienza sarà inferiore.

Il tempo che impiega A a trasmettere tutti i frame delle finestra è necessariamente minore del tempo che impiega il primo frame a raggiungere B e ad essere processato(da cui calcolo la finestra):

$$W_s T_f \leq T_f + 2\tau \implies W_s \leq 1 + 2\alpha \quad (5.20)$$

Il primo ACK arriverà al di fuori di questa finestra, quindi impiegherà almeno  $T_f + 2\tau$

Figure 5.19: Riscontri fuori dalla finestra di trasm.  $W_s$

Pertanto l'efficienza degli schemi visti prima, va moltiplicata per un fattore( $w$ ) di efficienza che pesa l'efficienza finale e che rappresenta il valore limite per SR e GBN nel caso di assenza di errori e riscontri fuori dalla finestra di trasmissione. Tale valore è dato dal rapporto tra tempo utile di occupazione del canale e tempo di attesa del primo riscontro:

$$\eta_w = \frac{W_s T_f}{T_f + 2\tau} = \frac{W_s}{1 + 2\alpha} \quad (5.21)$$

Per cui le efficienze si SR e GBN sono:

$$\eta_{GBN} = \frac{1 - P}{1 + P(W_s - 1)} \cdot \frac{W_s}{1 + 2\alpha} \quad \eta_{SR} = (1 - P) \cdot \frac{W_s}{1 + 2\alpha}$$

Nel GBN invece, il valore K è proprio  $W_s$  in quanto in caso di errore si ritrasmette l'intera finestra. Se si ipotizza stesso valore di  $N$  per SR e GBN, considerando il valore ottimale delle finestre, si ha:

$$\begin{aligned} \eta_{GBN} &= \frac{1 - P}{1 + P(N - 2)} \cdot \frac{N - 1}{1 + 2\alpha} \\ \eta_{SR} &= (1 - P) \cdot \frac{N/2}{1 + 2\alpha} \end{aligned}$$

### 5.1.9 Confronto tra protocolli di linea

	Stop&Wait	Go-back-N	Selective Repeat
$W_S \leq 1 + 2a$	$\frac{1 - P}{1 + P(N - 2)}$	$\frac{N - 1}{1 + 2a}$	$(1 - P) \frac{N/2}{1 + 2a}$
$W_S > 1 + 2a$	$\frac{1 - P}{1 + 2a}$	$\frac{1 - P}{1 + P(1 + 2a)}$	$1 - P$

Figure 5.20: Confronto dell'efficienza tra i protocolli Stop-and-Wait, Go-Back-N e Selective Repeat

**Confronto tra protocolli** Il grafico mostra l'andamento dell'efficienza dei protocolli Stop-and-Wait, Go-Back-N e Selective Repeat al variare del ritardo di propagazione normalizzato  $\alpha$  e della probabilità di errore  $P$ .

La scelta del protocollo di linea dipende dalla situazione:

- caso in cui i riscontri siano all'interno della finestra di ricezione: SR è sempre più efficiente, ma per bassi valori di  $\alpha$  tutti hanno efficienza simile e pari a  $1-P$
- caso in cui i riscontri siano fuori della finestra di trasmissione: il diverso valore della finestra di trasmissione può rendere più efficiente GBN rispetto al SR,
- canali poco rumorosi( $P$  basso) e  $\alpha$  elevato: si può avere un'efficienza maggiore del GBN: infatti, è vero che con il GBN si ritrasmette l'intera finestra, ma questo evento accade raramente, mentre durante la trasmissione senza errori si trasmettono più frame di continuo (es. link satellitari)
- canali molto rumorosi( $P$  alto): sarà nuovamente più efficiente il SR poiché si ritrasmette spesso per cui conviene ritrasmettere la singola trama piuttosto che l'intera finestra

La scelta del protocollo dipende dal compromesso tra efficienza, complessità(\$) e condizioni del canale.

## 5.2 Livello 2 Frame DHL

### 5.2.1 Struttura del frame

Flag 01111110	Address	Control	Information	FCS	Flag 01111110
8 bit	$8 \times n$ bit	8/16 bit	Variable ( $8 \times m$ bit)	16/32 bit	8 bit

Figure 5.21: Struttura del frame HDLC

- flag: sono posti ad inizio e fine frame così da delimitare il frame e far capire a chi lo riceve quando inizia e quando finisce il frame, questi flag sono SEMPRE 8 bit in questa sequenza: 01111110, ossia 7E in esadecimale
- address: indirizzo del trasmettitore, ricevitore o broadcast(se tutti 1)
- control: indica il tipo di frame
- information: dati del livello superiore, payload
- FCS(frame check sequence):

### 5.2.2 Bit stuffing

A proposito dei flag usati nel frame(01111110) che succede se questa sequenza invece fa parte della informazione utile del frame e non come flag delimitatore?

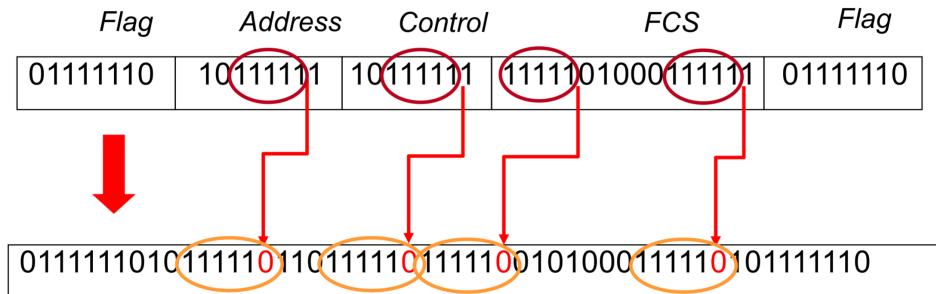


Figure 5.22: Esempio di bit stuffing

Risolvo inserendo uno 0 ogni cinque 1 ripetuti; questo fa perdere la molteplicità di 8 del frame.

### 5.2.3 Rivelazione dell'errore con CRC

HDLC numera i frame e usa il protocollo di linea, utilizzando 3 o 7 bit, usando una tecnica di piggybacking per gestire il flusso che viene dalla direzione opposta(finestre di trasmissione e ricezione).

Il CRC è un metodo per il calcolo di somme di controllo (checksum). Il nome deriva dal fatto che i dati d'uscita sono ottenuti elaborando i dati di ingresso i quali vengono fatti scorrere ciclicamente in una rete logica.

Sfrutta l'algebra dei campi finiti ed è utile per l'individuazione di errori casuali nella trasmissione dati (a causa di interferenze, rumore di linea, distorsione), il CRC non è invece affidabile per verificare la completa correttezza dei dati contro tentativi intenzionali di manomissione.

**Polinomio generatore** Un codice CRC è definito dal suo polinomio generatore di ordine r:  
esempio r = 3

$$G(x) = x^3 + x^2 + 1 \Rightarrow 1101 \quad (5.22)$$

### Cosa avviene in trasmissione

Con una scelta opportuna del polinomio generatore è possibile rilevare errori sul singolo bit etc... ;  
setup del CRC per la trasmissione:

Traduco i bit che compongono header e payload  
in un polinomio  $P(x)$  di coefficienti d; perciò di  
grado d-1.

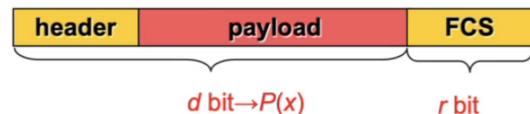


Figure 5.23: Schema del calcolo di CRC,  $P(x)$

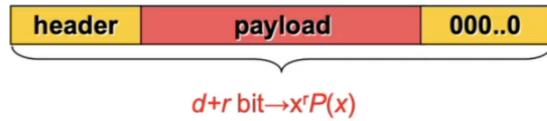


Figure 5.24: Schema del calcolo di CRC,  $x^r P(x)$

Si moltiplica  $P(x)$ , ricavato qui sopra, per  $x^r$  (shift di r posizioni, le stesse del polinomio generatore), ottendo il polinomio  $x^r P(x)$  di grado d+r-1, con coefficienti d+r.

In seguito allo shift, il campo FCS del frame sarà costituito da r zeri.

**Calcolo e verifica del CRC in trasmissione** In trasmissione si divide il polinomio  $x^r P(x)$  per  $G(x)$ , il resto di questa operazione [il polinomio  $R(x)$ ] sono i bit del CRC, da inserire nel FCS del frame. Vedi operatore XOR appendice basi.

$$\frac{x^r P(x)}{G(x)} = Q(x) \oplus \frac{R(x)}{G(x)} \quad (5.23)$$

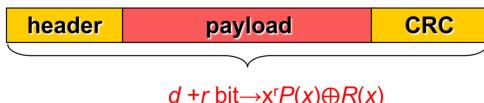


Figure 5.25: Inserimento bit in FCS

Devo quindi inserire i bit del polinomio  $R(x)$  all'interno del FCS, effettuando l'operazione:

$$P^{Tx} = x^r P(x) \oplus R(x) \quad (5.24)$$

## Verifica CRC

C'è la necessità di verificare se il CRC calcolato ed inserito nel FCS sia corretto.

**Polinomio relativo** è necessario calcolare il polinomio relativo  $P^{Rx}(x)$  per verificare CRC; questo polinomio non è altro che la trama di d+r bit ricevuti in ricezione.

**Condizione necessaria** Infine per verificare CRC c'è da tenere a mente la condizione necessaria affinché il CSC sia corretto, ossia che il resto della divisione tra polinomio relativo [ $P^{Rx}(x)$ ] e polinomio generatore [G(x)] sia nullo:

$$\frac{P^{Rx}(x)}{G(x)} = Q'(x) \oplus R'(x) \quad (5.25)$$

Può accadere che il resto sia nullo con trama ricevuta errata (la condizione è necessaria, non sufficiente).

Non si può quindi essere certi della correttezza della trama (c'è sempre una probabilità non nulla di falso positivo).

Se il resto NON è nullo, invece, c'è la certezza che la trama sia errata perché la condizione necessaria è stata violata

**Dimostrazione condizione necessaria** Nel caso in cui la trama ricevuta sia corretta, il polinomio che costruiamo da questa trama deve essere uguale al polinomio originale (trama trasmessa), cioè:

$$P^{Rx}(x) = P^{Tx}(x) \quad (5.26)$$

Se effettuiamo la divisione per il polinomio generatore:

$$\frac{P^{Rx}(x)}{G(x)} = \frac{P^{Tx}(x)}{G(x)} = \frac{x^r P(x) \oplus R(x)}{G(x)} = \frac{x^r P(x)}{G(x)} \oplus \frac{R(x)}{G(x)} \quad (5.27)$$

$$\frac{P^{Rx}(x)}{G(x)} = \frac{x^r P(x)}{G(x)} \oplus \frac{R(x)}{G(x)} = Q(x) \oplus \frac{R(x)}{G(x)} \oplus \frac{R(x)}{G(x)} = Q(x) \quad (5.28)$$

## Esempio calcolo CRC

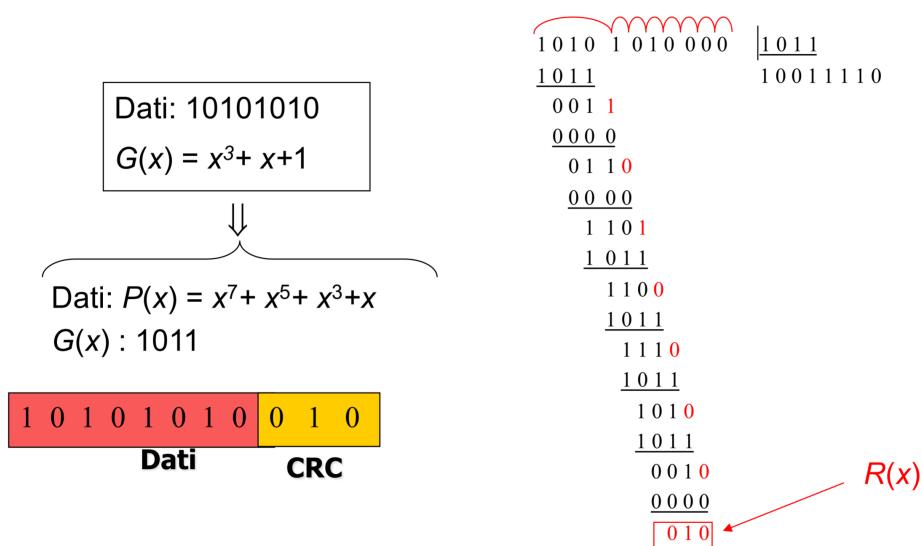


Figure 5.26: Esempio di calcolo CRC

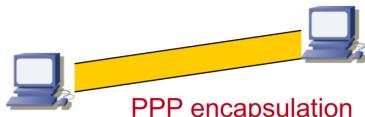
#### 5.2.4 Protocollo point to point (PPP)

IETF RFC 1661, 1662

protocollo di livello 2

punto-punto

diversi protocolli di livello superiore



Flag	Address	Control	Protocol	Information	FCS	Flag
8 bit	$8 \times n$ bit	8/16 bit	16 bit	Variable ( $8 \times m$ bit)	16/32 bit	8 bit

Figure 5.27: Protocollo Point to Point

Il protocollo Point to Point è comunemente usato nello stabilire connessioni dirette tra due nodi.

Il frame è quasi identico a quello del HDLC, ha un campo aggiuntivo, quello protocol: specifica il protocollo di livello 3 che stiamo trasportando.

**Network bit order - Big Endian e Little Endian** A differenza dello standard HDLC, big-endian, il protocollo PPP non è endianness specifico.

L'ordine dei byte (conosciuto anche come big-endian, little-endian o middle-endian a seconda dei metodi differenti), in informatica, indica modalità differenti usate dai calcolatori per immagazzinare all'interno della memoria dati di dimensione superiore al byte; dipende essenzialmente dall'architettura hardware usata.

"I termini big-endian e little-endian derivano dai nomi di due popolazioni che abitavano nelle favolose isole di Lilliput e Blefuscu nel romanzo I viaggi di Gulliver di Jonathan Swift. Queste erano entrate in rivalità per il modo in cui aprivano le uova - rompendo la punta o il fondo: a Lilliput, per editto dell'imperatore il cui figlio una volta si tagliò aprendo un uovo dall'estremità più grande, fu ordinato di aprire le uova dall'estremità più piccola (little endians); a Blefuscu si rifugiarono gli oppositori che volevano conservare la tradizione di rompere le uova dall'estremità più grande (big endians). A causa di questa differenza e della sua legittimazione imperiale era scoppiata tra le due isole una guerra sanguinosa."

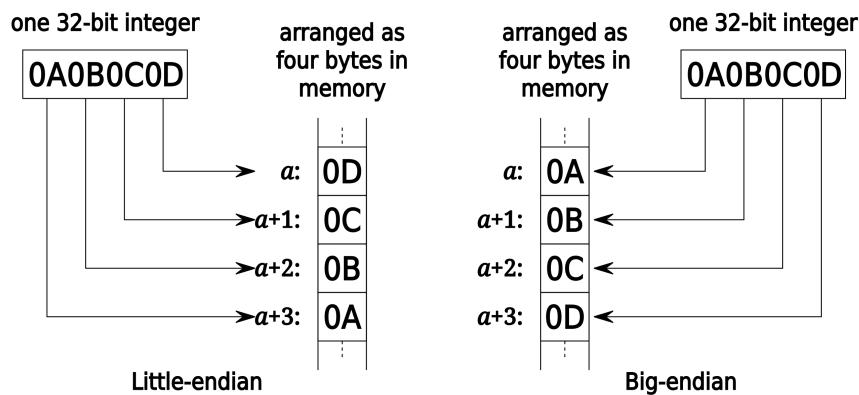


Figure 5.28: Big Endian e Little Endian

## Byte stuffing

Anche qui, vista la struttura del pacchetto molto simile a quella del frame HDLC, c'è il rischio di confondere il dato informativo con un flag; perciò si attua il byte stuffing.

**Sequenza di flag 0x7d** Se si vuole evitare la sequenza 0x7e (01111110), si antepone la sequenza di controllo 0x7d (01111101); al posto della sequenza 0x7e inseriamo la sua versione in XOR con la sequenza 0x20 (00100000).

A quel punto avremo individuato la sequenza "pericolosa" all'interno del frame, ossia 0x7e, avremo un identificativo, 0x7d, che ci dice dov'è situata nel frame e la sostituiremo con il suo valore in XOR con 0x20.

Quindi quando all'interno del frame leggo il byte 0x7d, il nuovo flag che indica che c'è stato byte stuffing, non leggiamo 0x7d ma il byte successivo, quello calcolato tramite il 0x7e XOR 0x20.

In ricezione se applico nuovamente XOR 0x20 al byte precedentemente modificato con lo XOR, allora otterrò nuovamente il byte iniziale, 0x7e.

Es.

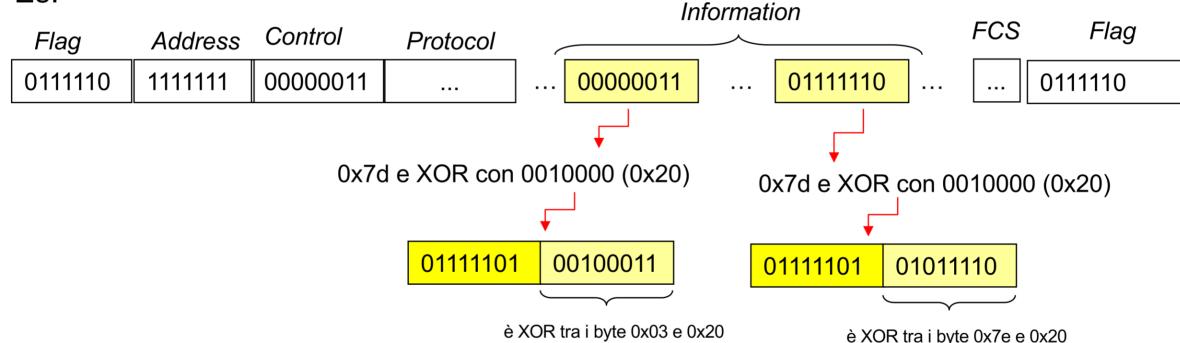
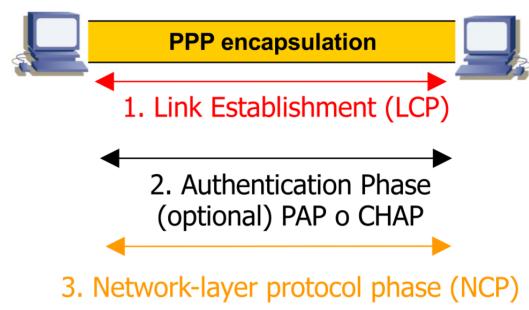


Figure 5.29: Esempio di byte stuffing

Nel caso in cui incappiamo con una "falsa" sequenza di flag del byte stuffing, ossia 0x7d, la trasmetteremo come 0x7d, 0x5d.

## Struttura PPP

Il PPP utilizza due protocolli per la gestione del collegamento tra nodi:



- LCP(Link Control Protocol): grazie al quale si apre la connessione, si ha la verifica della qualità del collegamento e rende possibile l'autenticazione(PAP o CHAP)
- NCP(Network Control Protocol): con cui è possibile scegliere e configurare uno o più protocolli di rete(es. IP)

Figure 5.30: Encapsulation PPP

## Autenticazioni

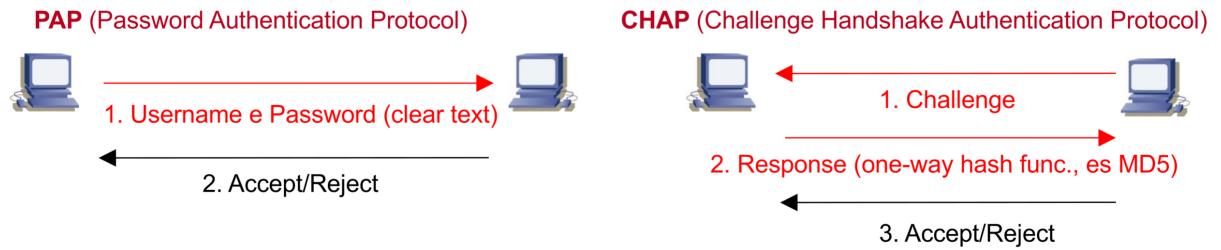


Figure 5.31: Autenticazione PAP e CHAP

**Autenticazione PAP** Password Authentication Protocol (PAP) invia username e password in chiaro. È semplice ma poco sicuro, poiché le credenziali possono essere intercattate(password e username sono solitamente criptate).

**Autenticazione CHAP** Challenge Handshake Authentication Protocol (CHAP) utilizza un meccanismo di challenge-response con hash per autenticare in modo più sicuro, evitando di trasmettere la password in chiaro.

## 5.3 Protocolli di accesso

A livello due viene svolto un ruolo importante dai protocolli di accesso, che possono essere di due tipologie:

- Accesso ordinato:
  - Ogni nodo ha un turno prestabilito per trasmettere, evitando collisioni.
  - Utilizzato in sistemi come il polling o il token passing.
  - Garantisce un utilizzo equo del canale, ma può introdurre ritardi se un nodo non ha dati da trasmettere.
- Accesso casuale:
  - senza rilevazione del canale:
    - \* I nodi trasmettono senza verificare se il canale è libero.
    - \* Può portare a collisioni frequenti, come nel protocollo ALOHA puro.
  - con rilevazione del canale:
    - \* senza rivelazione di collisioni:
      - I nodi verificano se il canale è libero prima di trasmettere.
      - Non rilevano collisioni, quindi i dati persi devono essere ritrasmessi dopo un timeout.
      - Esempio: Carrier Sense Multiple Access (CSMA).
    - \* con rilevazione di collisioni:
      - I nodi verificano se il canale è libero e rilevano eventuali collisioni durante la trasm.
      - In caso di collisione, interrompono la trasm. e ritentano dopo un intervallo casuale.
      - Esempio: CSMA/CD (utilizzato in Ethernet).

### 5.3.1 Troughput di rete

Per valutare le prestazioni di una rete vengono considerati parametri come ... ; il Troughput, tra questi, è uno dei più rilevanti: è definito come il numero di bit al secondo che giungono corretti; maggiore è questo valore, meglio è per le prestazioni della rete.

Questo valore è alto quando gli errori nei pacchetti che giungono a destinazione sono pochi.

Si parla anche di goodput, ossia il rateo bit al secondo che giungono corretti a destinazione; il Troughput considerava tutti i pacchetti/bit che arrivano a destinazione, anche quelli errati.

#### Calcolo del Troughput

Per il calcolo del Troughput è fondamentale definire alcuni concetti:

- frequenza di cifra  $C$ : è un paramentro tipico del livello 2 (datalink), rappresenta la capacità del canale su cui sta viaggiando l'informazione, quindi la quantità di bit che possono fisicamente muoversi all'interno del canale
- Troughput massimo: è pari alla frequenza di cifra  $C$
- $\Lambda_s$ : è il numero medio di pacchetti emessi dalla sorgente
- $\pi$ : la probabilità che il pacchetto arrivi errato o che non arrivi proprio
- $\Lambda_T$ : Troughput medio

**Troughput medio packet/s** Il Troughput medio  $\Lambda_T$ , misurato in pacchetti al secondo, si calcola come:

$$\Lambda_T = \Lambda_s(1 - \pi) \quad [\text{packet/s}] \quad (5.29)$$

**Troughput medio bit/s** se desiderassi calcolare il Troughput medio misurato in bit al secondo, ho bisogno di definire il parametro  $L$ , ossia la lunghezza media dei pacchetti, allora il Troughput vale:

$$\Gamma_T = \Lambda_s L (1 - \Pi) \quad [\text{bit/s}] \quad (5.30)$$

**Troughput adimensionale** è anche possibile definire un valore adimensionale del Troughput, che varia da un valore utile che va da 0 ad uno massimo di 1; lo si ottiene normalizzando  $\Gamma_s$  alla capacità del canale  $C$ ; inoltre definiamo:

- $T = \frac{L}{C}$  è il tempo medio di trasmissione di un singolo pacchetto
- $G = \Gamma_s T$  è il traffico medio normalizzato emesso dalla sorgente

$$S = \frac{\Lambda_s L}{C} (1 - \Pi) = \frac{\Lambda_T}{C} = G(1 - \Pi) \quad (5.31)$$

se  $S$  è maggiore di 1 significa che il mittente sta ricevendo più bit di quelli che il canale può supportare.

### 5.3.2 Protocollo ALOHA puro

ALOHA è un protocollo di accesso multiplo casuale senza rilevazione del canale.

Questo protocollo non prevede vincoli all'invio di dati e quindi all'occupazione della banda. Quando una postazione ha dati da trasmettere, li trasmette.

Poiché ogni stazione agisce indipendentemente dalle altre, il successo è determinato unicamente dalla mancata collisione con altre trasmissioni da parte di altre stazioni. Poiché i canali broadcast danno la possibilità di verificare (feedback) se il frame trasmesso è stato ricevuto correttamente oppure se si sono verificate collisioni, la stazione trasmittente ascolta il canale e determina il successo o l'insuccesso della trasmissione. Qualora non sia possibile ascoltare il canale le stazioni si mettono in attesa di un riscontro (ack) da parte del ricevente. Se ci sono collisioni (o se l'ack non arriva entro un tempo di attesa stabilito), i frame corrotti vengono distrutti. Ciò indipendentemente dal livello di corruzione dei dati; quando un frame è stato interessato da collisione, viene eliminato. In questo caso la postazione mittente reinvia il frame dopo un'attesa casuale e si rimette in ascolto sul canale (o attende un ack) fino a quando non stabilisce che il frame è stato ricevuto correttamente.

#### Collisioni nel canale condiviso

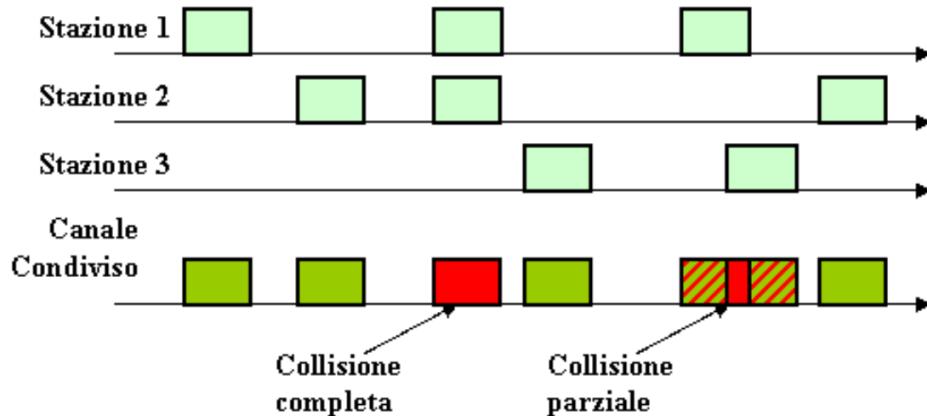


Figure 5.32: Collisioni nel protocollo ALOHA

Le stazioni trasmittenti inviano i loro messaggi lungo il canale condiviso. Se l'intervallo di tempo durante il quale due o più stazioni trasmettono si sovrappone, allora si avrà una collisione; quindi le stazioni coinvolte non riceveranno alcun ack e dovranno perciò ritentare la trasmissione in un istante successivo.

Per evitare che la collisione si ripeta indefinitamente è opportuno che le stazioni coinvolte tentino la loro ritrasmissione in tempi distinti, in modo da ridurre la probabilità di nuove sovrapposizioni fra i due periodi di trasmissione.

## Calcolo del Troughput di ALOHA puro

Si ipotizza che le stazioni trasmettano sul canale secondo un processo di Poisson, quindi ognuna di esse è un evento indipendente dagli altri; le stazioni trasmettono pacchetti nel canale indipendentemente da ciò che fanno le altre stazioni.

Si suppone anche che ci sia un numero di stazioni molto elevato, che trasmettano tutte nel canale condiviso di interesse;

ricordando la formula del Troughput medio espresso in pacchetti al secondo:

$$\Lambda_T = \Lambda_s(1 - \pi) \quad [\text{packet/s}] \quad (5.32)$$

ci interessa capire quanto può valere  $(1 - \pi)$  nel protocollo ALOHA puro, quindi la probabilità che i pacchetti arrivino a destinazione considerando solamente l'effetto della collisione tra pacchetti, non altri fenomeni, poiché si vuole valutare solo l'effetto del protocollo ALOHA sul Troughput.

Per fare ciò considero un nuovo parametro, il periodo di vulnerabilità.

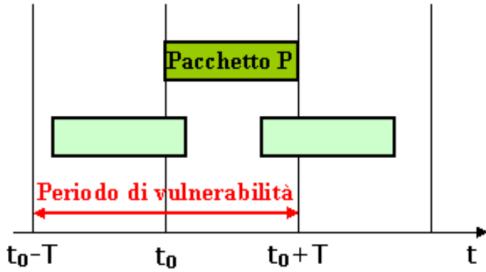


Figure 5.33: Periodo di vulnerabilità

Il periodo di vulnerabilità rappresenta l'intervallo di tempo durante il quale una trasmissione(pacchetto verde chiaro) può essere soggetta a collisioni(pacchetto P collide con la trasmissione se rientra nella finestra di  $2T$ ). Nel caso del protocollo ALOHA puro, questo periodo è pari a  $2T$ , dove  $T$  è il tempo di trasmissione di un singolo pacchetto. Durante questo intervallo, se un'altra stazione inizia a trasmettere, si verifica una collisione.

Il valore  $(1 - \pi)$  rappresenta quindi la probabilità per la quale nell'intervallo di trasmissione di un pacchetto questo arrivi correttamente(infatti  $1 - \pi$ , dove  $1$  è la probabilità massima per la quale non ci sia collisione,  $\pi$  invece rappresenta la probabilità di collisione, più è alta e più andrà a diminuire il valore ideale  $1$ ).

Dall'espressione di Poisson( $p_0$  rappresenta la probabilità che zero eventi (collisione di pacchetti) si verifichino in un certo intervallo di tempo) si calcola che la probabilità di non avere eventi di collisione nell'intervallo di vulnerabilità  $2T$  è la seguente:

$$p_k = \frac{(\Lambda t)^k}{k!} e^{-\Lambda t} \Rightarrow p_0 = \frac{(\Lambda_s 2T)^0}{0!} e^{-\Lambda_s 2T} = e^{-2\Lambda_s T} = (1 - \pi) \quad (5.33)$$

**Troughput in packet/s** sostituendolo nella formula di  $\Lambda_T$ :

$$\Lambda_T = \Lambda_s(1 - \Pi) = \Lambda_s p_0 = \Lambda_s e^{-2\Lambda_s T} \quad \text{packet/s} \quad (5.34)$$

**Troughput in bit/s**

$$\Gamma_T = \Lambda_s L(1 - \pi) = \Lambda_s L(e^{-2\Lambda_s T}) \quad [\text{bit/s}] \quad (5.35)$$

**Troughput adimensionale**

$$S = \frac{\Lambda_s L}{C}(1 - \Pi) = \frac{\Lambda_T}{C} = G(1 - \Pi) = G(e^{-2\Lambda_s T}) \quad (5.36)$$

## Prestazioni ALOHA puro

**Andamento funzione  $G(e^{-2\Lambda_s T})$**  analizzando la funzione

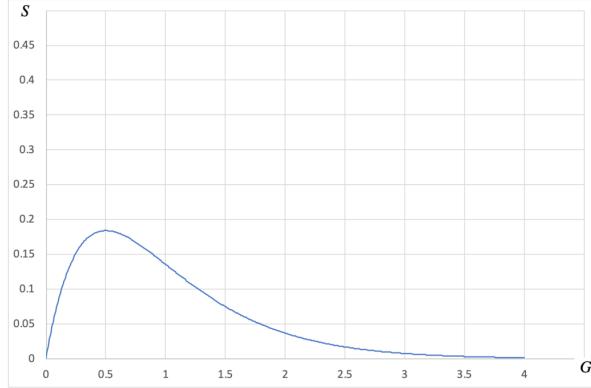


Figure 5.34: Andamento Troughput normalizzato ALOHA puro

La funzione  $S = Ge^{-2G}$  ha un massimo per  $G = 0.5$ , che corrisponde a  $S = 0.184$ .

Questo significa che il massimo Troughput normalizzato ottenibile con ALOHA puro è circa il 18.4% della capacità del canale, un valore molto basso dovuto all'elevata probabilità di collisione.

Le prestazioni basse sono dovute al fatto che le stazioni trasmettono senza alcun controllo sul canale.

## Appendice matematica - distribuzione di Poisson

La distribuzione di Poisson  $\mathcal{P}_\lambda(n)$  è una distribuzione di probabilità discreta data da

$$\mathcal{P}_\lambda(n) = \frac{\lambda^n e^{-\lambda}}{n!} \quad \text{per ogni } n \in N, \quad (5.37)$$

dove  $\lambda$  è il numero medio di eventi per intervallo di tempo, mentre  $n$  è il numero di eventi per intervallo di tempo (lo stesso col quale si misura  $\lambda$ ) di cui si vuole la probabilità.

**Esempio** Un call center riceve in media 4 chiamate all'ora. Qual è la probabilità che riceva esattamente 2 chiamate in un'ora? ( $\lambda = 4$ ,  $n = 2$ )

$$\mathcal{P}_4(2) = \frac{4^2 \cdot e^{-4}}{2!} = \frac{16 \cdot e^{-4}}{2} \approx \frac{16 \cdot 0.0183}{2} \approx \frac{0.2928}{2} \approx 0.1464$$

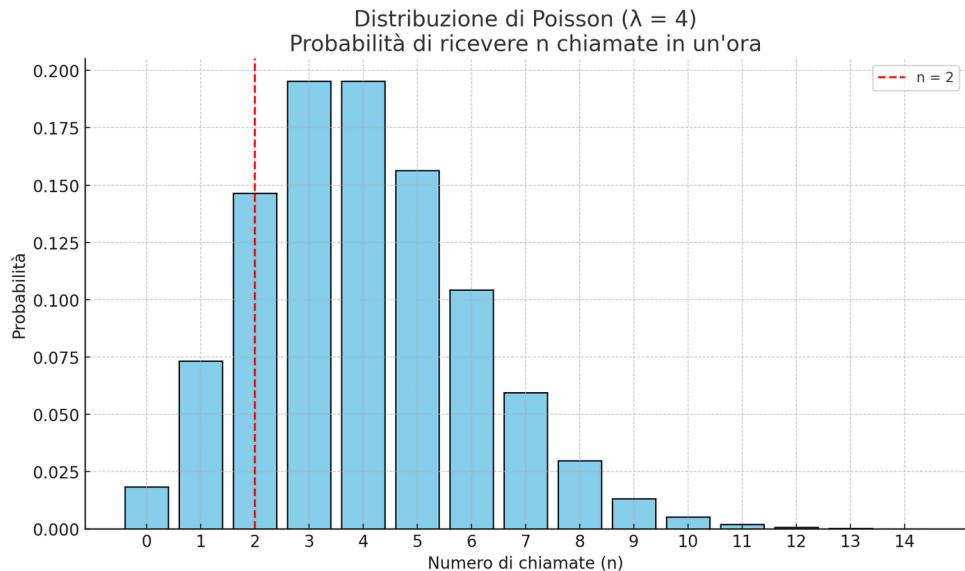


Figure 5.35: Grafico della distribuzione di Poisson

### 5.3.3 Protocollo ALOHA slotted - prestazioni

Il protocollo Slotted Aloha aggiunge al protocollo Aloha (da cui deriva) un’ulteriore caratteristica, ovvero la suddivisione del tempo in intervalli discreti chiamati slot.

Ogni stazione è vincolata a cominciare la propria trasmissione all'inizio di uno slot temporale.

Se una stazione ad un certo istante è pronta a trasmettere dovrà attendere necessariamente l'inizio del successivo slot. La conseguenza di tale caratteristica è che due trasmissioni o collidono completamente all'interno dello stesso slot oppure non collidono affatto; il problema delle collisioni parziali non c'è.

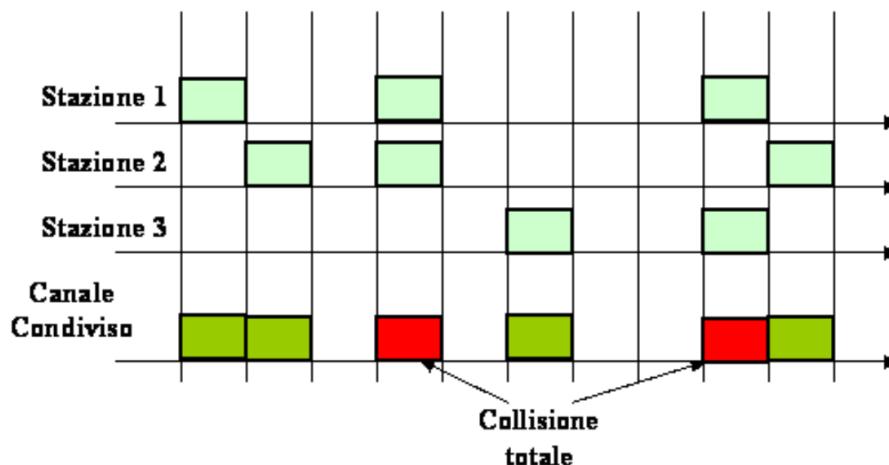


Figure 5.36: Protocollo ALOHA slotted

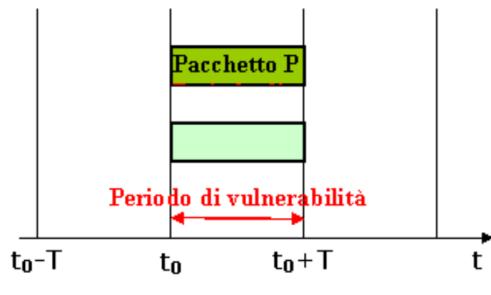


Figure 5.37: Protocollo ALOHA slotted

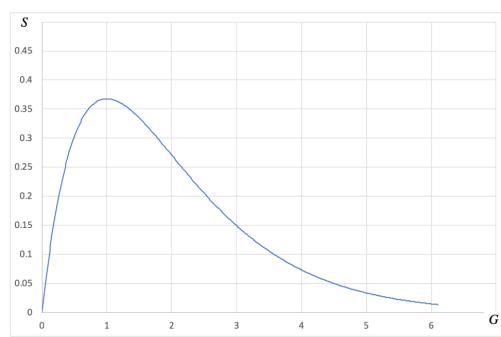


Figure 5.38: Protocolo ALOHA slotted

**Probabilità di collisione** Il modello è identico a quello visto per ALOHA, ma stavolta il tempo di vulnerabilità si è dimezzato per cui la probabilità di non avere tentativi di trasmissione in  $T$

$$p_0 = \frac{(\Lambda_s T)^0}{0!} e^{-\Lambda_s T} = e^{-\Lambda_s T} \quad (5.38)$$

### Troughput normalizzato

$$S = \Lambda_s \frac{L}{C} p_0 = \Lambda_s \frac{L}{C} e^{-\Lambda_s T} = G e^{-\Lambda_s T} \quad (5.39)$$

Il protocollo Slotted Aloha ha come conseguenza il dimezzamento del periodo di vulnerabilità, che in tal caso è pari a  $T$ . L'efficienza massima risulta conseguentemente raddoppiata, pari quindi al 36,8%.

### 5.3.4 Protocolli Carrier Sense Multiple Access - CSMA (rilevazione canale)

I protocolli CSMA sono quei protocolli di accesso che rilevano il canale prima di trasmettere, per migliorare l'efficienza ed evitare collisioni con altre trasmissioni.

Se rilevando il canale, risulta occupato, allora la stazione può attuare diverse strategie:

- 1-persistent: continuare a "sentire" il canale finché non lo rileva libero per trasmettere subito dopo
- no-persistent: rinviare la trasmissione ad un momento successivo, generalmente scegliendo un intervallo di attesa casuale
- p-persistent: scegliere di comportarsi con probabilità  $p$  in una delle due modalità precedenti

"Può accadere di rilevare il canale come libero, ma nel tempo in cui l'ho rilevato e ho iniziato ad utilizzarlo qualcun'altro potrebbe iniziare a sua volta la comunicazione, c'è quindi da considerare il tempo che impiega l'onda elettromagnetica a viaggiare nel mezzo  $v$  e noi a rilevarla  $\tau$ ".

#### CSMA/CD - CSMA with collision detection

Per realizzare il CSMA/CD è necessario che la stazione possa a livello fisico realizzare una comunicazione full duplex (per essere in grado a livello fisico di trasmettere e ricevere contemporaneamente, per permettere il listening while talking spiegato successivamente).

La stazione che intende trasmettere verifica che il canale sia libero prima di trasmettere (sensing del canale del CSMA);

**Listening while talking** se il canale è libero si trasmette la trama e durante la trasmissione della stessa (tra l'invio del primo e dell'ultimo bit) si continua ad ascoltare il canale (Listening while talking)(ascolta se stessa).

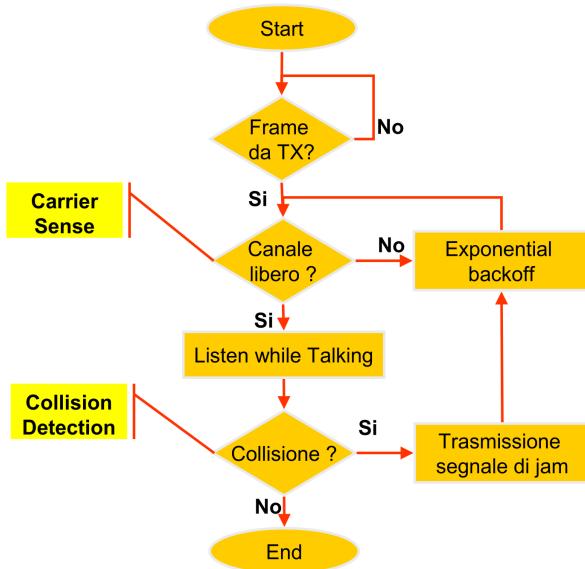


Figure 5.39: Esempio CSMA/CD

**Rilevazione collisione** Se il segnale "ascoltato" (confronto tra le forme d'onda a livello fisico) è diverso da quello trasmesso, si rileva la collisione (altra stazione che ha iniziato a trasmettere sentendo il canale libero).

In tal caso si può:

- interrompere la trasmissione
- inviare un segnale di jam per informare chi sta ascoltando il canale di scartare nel buffer di ricezione tutti i bit della trama parziale ricevuta

**Exponential backoff** Il tempo di attesa per la rilevazione del canale in seguito ad una collisione viene calcolato in un intervallo  $T_{slot} * [0; 2^m - 1]$  tramite un algoritmo di exponential backoff, per il quale  $m = \min(n; 10)$ ,  $m$  è il numero di ritrasmissioni,  $n$  è il numero di collisioni consecutive,  $T_{slot}$  è il tempo necessario a trasmettere un frame di lunghezza minima(512 bit).

### Dimostrazione funzionamento collision detection

Utilizzando il CSMA/CD è necessario che i frame abbiano una dimensione superiore ad un valore minimo per garantire il funzionamento del collision detection.

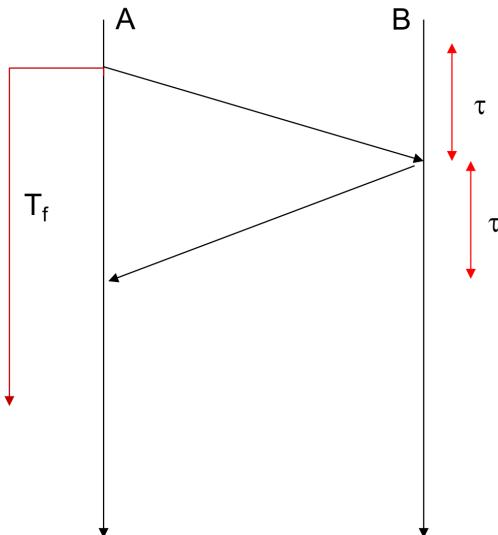


Figure 5.40: Dimostrazione collision detection

A e B comunicano nei tanti  $\tau$  (valore massimo, caso peggiore); il primo bit inviato da A arriva a B dopo  $\tau$  secondi.

Nel caso peggiore B ascolta il canale un istante( $\epsilon$ ) prima che il bit di A arrivi a B, perciò B vedrà che il canale è libero, non sapendo che un'istante dopo arriveranno i bit di A, causando quindi una collisione. B si accorge della collisione e manda un jam, impiegando  $\tau$  per "avvisare" A.

A può aprire che il jam che riceve è dovuto ad una collisione con la sua trasmissione solo se la sua trasmissione ( $T_f$ ) dura almeno quanto il tempo che ha impiegato il suo primo bit di trasmissione ed il jam di B, perciò:  $T_f \geq 2\tau$

$$T_f = \frac{L_f}{C} \geq 2\tau = \frac{2d}{v} \implies L_f \geq \frac{2dC}{v} \quad (5.40)$$

### CSMA/CA - CSMA with collision avoidance

**Duration** L'obiettivo del CSMA/CA è diminuire il numero di collisioni.

In questa versione di CSMA, in cui c'è rilevazione del canale prima di trasmettere, se il canale è libero inizia la trasmissione inserendo la durata stimata della trasmissione del singolo frame (considerando anche i tempi di propagazione e la trasmissione del frame di riscontro) in un campo "duration" dell'header del frame.

**Network allocation vector(NAV) - virtual sensing** Tutte le altre stazioni che condividono il canale ascoltano il frame e settano un loro orologio interno, detto NAV, pari al valore del duration letto.

Finchè il NAV non si azzera non iniziano altre trasmissioni poiché si sa che il canale è occupato.

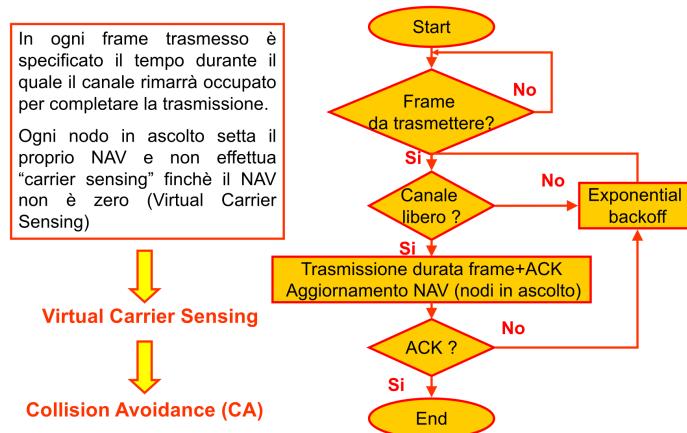


Figure 5.41: CSMA/CA

## 5.4 Il Progetto IEEE 802

Il progetto **IEEE 802** ha l'obiettivo di definire gli **standard per le reti locali (LAN)**, sia cablate che wireless, ed è stato successivamente esteso a:

- **MAN** (Metropolitan Area Network), es. **IEEE 802.16** (WiMAX)
- **WPAN** (Wireless Personal Area Network), es. **IEEE 802.15** (Bluetooth)

### Struttura della pila ISO/OSI nei livelli inferiori

IEEE 802 si occupa dei **primi due livelli** del modello ISO/OSI:

1. **Livello fisico (Physical Layer)**
2. **Livello di collegamento dati (Data Link Layer)**

Il livello Data Link viene ulteriormente suddiviso in:

- **LLC (Logical Link Control)** – definito dallo standard **IEEE 802.2**, è comune a tutte le tecnologie e fornisce un'interfaccia standard ai livelli superiori.
- **MAC (Media Access Control)** – specifico per ciascuna tecnologia, definisce come accedere al mezzo trasmissivo.

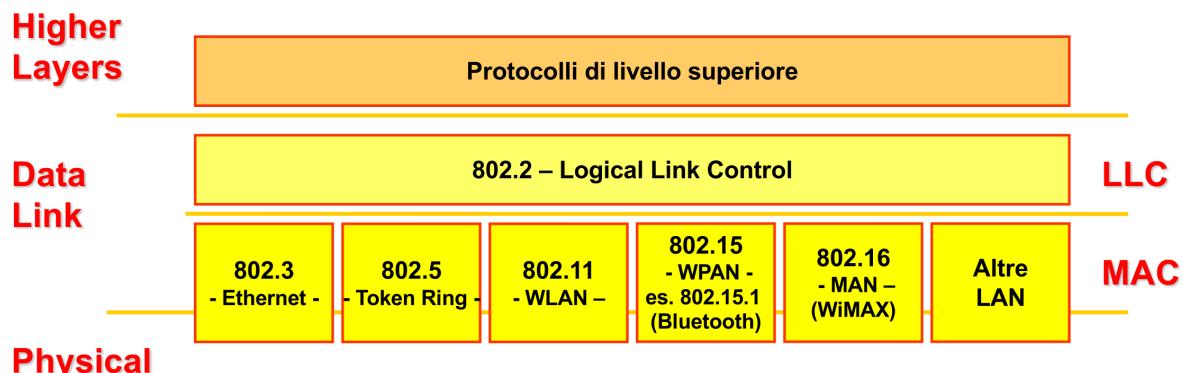


Figure 5.42: Struttura della pila ISO/OSI secondo IEEE 802

## IEEE 802.1

Lo standard IEEE 802.1 definisce le **caratteristiche generali del progetto**, comprese le specifiche per la struttura degli **indirizzi MAC**.

### Vantaggi della suddivisione LLC/MAC

- Favorisce la **modularità** e l'interoperabilità tra diversi tipi di rete.
- I protocolli di livello superiore (es. IP, TCP, UDP) possono utilizzare una singola interfaccia LLC, indipendentemente dallo standard MAC sottostante.
- Facilita l'**evoluzione tecnologica**, permettendo l'introduzione di nuovi standard MAC senza modificare gli strati superiori.

## 5.5 Reti wireless

Il wireless o senza fili indica una comunicazione tra dispositivi elettronici che non fa uso di cavi.

Generalmente il wireless utilizza onde radio a bassa potenza; tuttavia la definizione si estende anche ai dispositivi, meno diffusi, che sfruttano la radiazione infrarossa o il laser.

Ogni sistema di comunicazione wireless è composto da un trasmettitore, un ricevitore e dagli elementi deputati alla radiazione elettromagnetica ovvero alla trasduzione elettro-elettromagnetica e viceversa ovvero le antenne, i laser, i fotorivelatori.

### 5.5.1 Wireless LAN - WLAN

WLAN o wireless LAN è una rete locale (LAN) che sfrutta la tecnologia wireless, invece di una connessione cablata. In altre parole: con la sigla WLAN si indicano genericamente tutte le reti locali di computer che non utilizzano dei collegamenti via cavo per connettere fra loro gli host della rete; è una tecnologia definita da IEEE 802.11

- Pregi delle WLAN:
  - Facilità di installazione
  - Basso costo
  - Assenza di cablaggio
  - Mobilità
- Difetti delle WLAN:
  - Inaffidabilità mezzo trasmittivo
  - Sensibilità alle interferenze
  - Privacy

#### Infrastruttura WLAN

”Il sistema è composto da un insieme di BSS interconnessi con un Distribution System (DS) che collega i diversi Access Point (AP)”

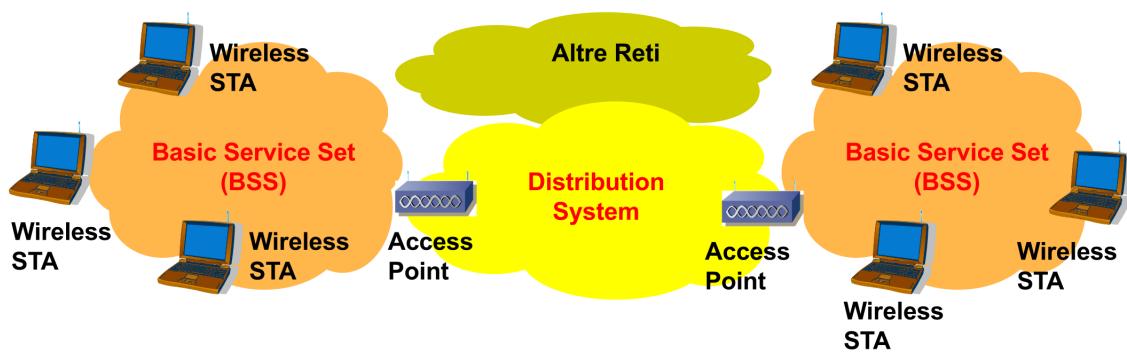


Figure 5.43: Infrastruttura WLAN

Nelle aree arancioni, dette Basic service set (BSS), le wireless station comunicano nello stesso canale radio; ciò che permette la comunicazione tra questa area con altre sono gli access point (AP).

**Access point** tramite il dispositivo access point è possibile che le tecnologie ethernet 802.3 e le tecnologie wireless 802.11 possano scambiarsi informazioni, fungendo da ponte tra rete cablata e rete wireless.

Funge tra ponte anche tra reti wireless con canali radio differenti, poiché il dato passa da un access point all'altro, utilizzando ethernet.

La famiglia 802.11x è uno standard IEEE che definisce le caratteristiche di una WLAN.

Lo standard 802.11x definisce una serie di tecniche di modulazione radio per segnali half-duplex che utilizzano di base lo stesso formato di dati. In particolare, la trasmissione si basa sul controllo di collisioni CSMA/CA per cui una emittente su uno specifico canale è in grado di determinare se su quello stesso canale ci siano altre entità che stanno trasmettendo (anche su protocolli diversi da quelli definiti da 802.11x) e trasmette le sue trame dati solo se il canale in quel momento è libero.

Protocollo	Frequenza	Data rate massimo (teorico)	Range (outdoor)
802.11a	5 GHz	54 Mbps	50 m
802.11b	2.4 GHz	11 Mbps	50 m
802.11g	2.4 GHz	54 Mbps	125 m
802.11n (WiFi 4)	2.4 GHz e/o 5 GHz	600 Mbps (4 data streams)	500 m
802.11ac (WiFi 5)	2.4 GHz e/o 5 GHz	6.77 Gbps (8 antenne, 160MHz)	200 m
802.11ax (WiFi6)	2.4 GHz e/o 5 GHz e/o 6GHz (WiFi6e)	circa 9 Gbps	100-200
802.11be (WiFi7)	2.4 GHz e/o 5 GHz e/o 6GHz	circa 24 Gbps	100-200

Figure 5.44: 802.11x

### Famiglia 802.11x

#### 5.5.2 Struttura dei canali 802.11b

Nello standard 802.11b la banda radio viene suddivisa in 14 canali, le "onde" rappresentano l'ampiezza di banda di ogni canale wireless.

Seguendo queste ampiezze si possono individuare le terne di canali che non interferiscono tra di loro, ad esempio 1, 6, 11; a conti fatti ogni canale ha dopo 5 canali quello con cui non interferisce.

Queste terne coprono l'area di interesse della rete wireless senza avere interferenze.

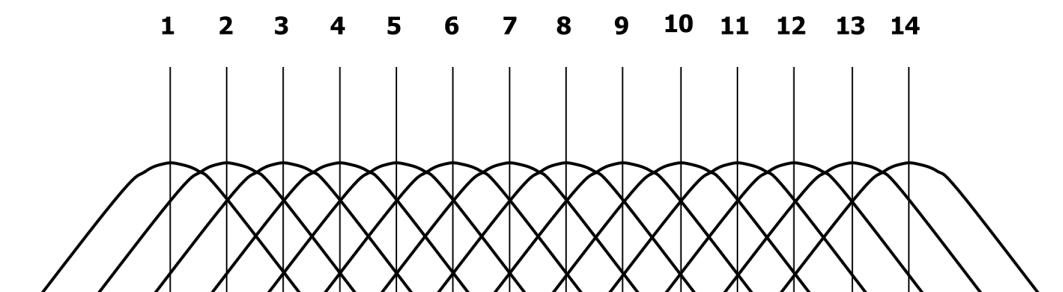
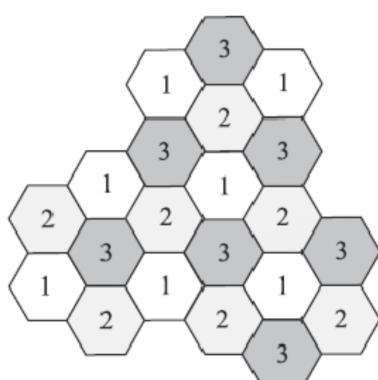


Figure 5.45: Struttura canali 802.11b



**Cluster** In questa rappresentazione ogni esagono rappresenta una cella o area di copertura di un access point (AP). I numeri all'interno degli esagoni indicano il canale utilizzato da ciascun AP.

Questa configurazione permette di minimizzare le interferenze tra le celle adiacenti, ottimizzando le prestazioni della rete wireless.

Figure 5.46: Cluster di 3 celle (1,2,3)

### Distributed Coordination Function (DCF)

La WLAN usa il CSMA/CA, la DCF è insieme di regole per la gestione dell'accesso al mezzo condiviso.

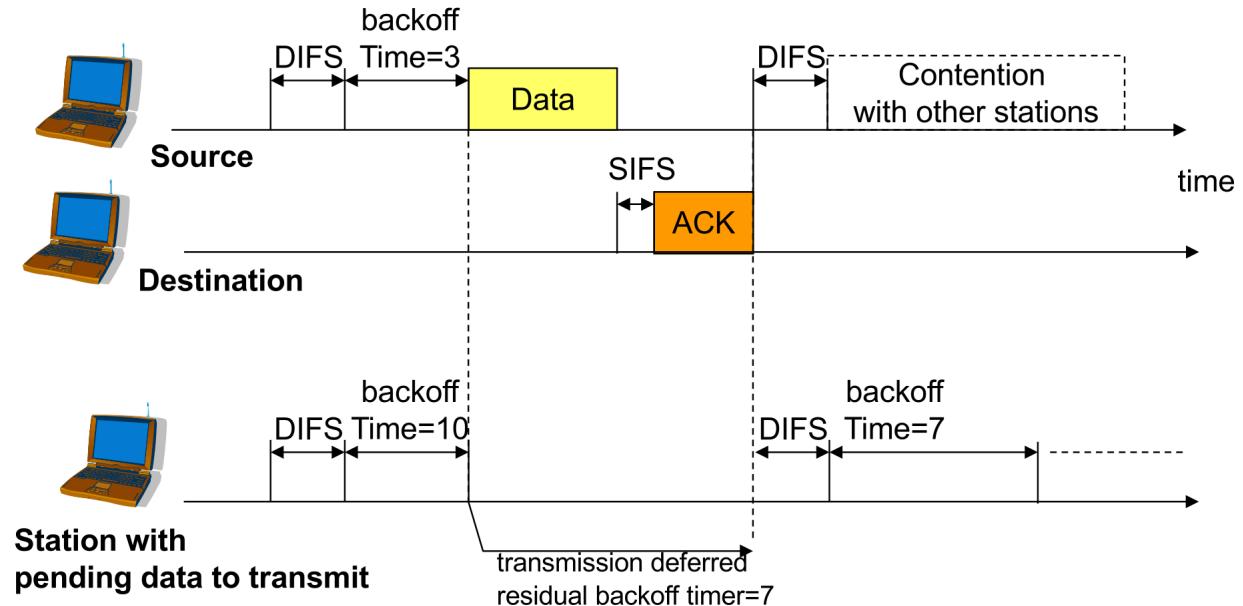


Figure 5.47: Distributed Coordination Function (DCF)

Source ha interesse di comunicare qualcosa(data) a destination, per farlo utilizza WNAL e perciò non può inviare direttamente l'informazione al destinatario ma lo fa attraverso un'ulteriore stazione, un access point(station with pending data to transmit).

Source ascolta il canale per un certo periodo prima di far partire la comunicazione(CSMA/CA), precisamente aspetta un tempo DIFS + backoff, rispettivamente:

- DIFS(DCF inter frame space): specifica la modalità di CSMA/CA utilizzata dal sistema, specifica per 802.11
- backoff time: tempo calcolato tramite algoritmo di backoff, per decidere il tempo di subentro in trasmissione, source e la stazione hanno dei backoff time diversi in questo caso

Nel momento in cui source inizia ad inviare i dati, il backoff time della stazione si interrompe(proseguirà solo al termine della comunicazione tra source e destination, poichè comunicano grazie alla station).

Source aspetta un riscontro(ACK) da parte di destination, che attenderà per un tempo SIFS(Short inter frame space), ossia un tempo predefinito che viene fornito al destination per assicurarsi di aver terminato correttamente i processi di ricezione, pronto per trasmettere l'ACK nel modo più corretto possibile. Una volta ricevuto l'ACK, la comunicazione è terminata.

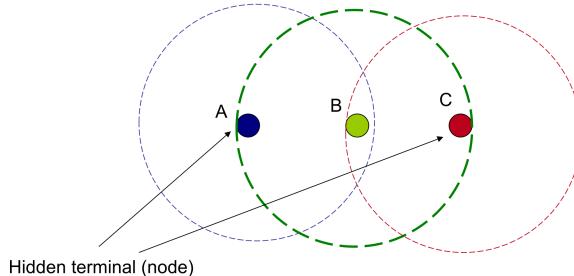
Quello che si vede successivamente nell'immagine, la "Contention with other stations", indica una possibile contesa con altre "source" per l'utilizzo della stazione per la trasmissione.

C'è sempre contesa tra gli apparati per poter comunicare con questa tecnologia.

### 5.5.3 Nodi nascosti ed esposti

Questi problemi esistono poichè i nodi condividono gli stessi canali per la condivisione.

**Nodo nascosto** I nodi “nascosti” (A e C) comunicano con B; i frame collidono nei pressi del nodo B.



**Nodo nascosto** Il nodo nascosto è un problema che si verifica nelle reti wireless quando due nodi non sono in grado di "vedersi" a vicenda a causa di ostacoli o distanza, ma entrambi possono comunicare con un terzo nodo. Ciò può portare a collisioni di pacchetti e perdita di dati.

Figure 5.48: Nodo nascosto

### Soluzione nodo nascosto

**Request to send RTS - 20 byte** I nodi che intendono trasmettere tramite un nodo intermedio devono inviare un piccolo frame di richiesta, detto Request to send (RTS).

Inoltre la probabilità che ci sia una collisione, quindi un problema, con l'invio di questi pacchetti è molto bassa poichè sono molto piccoli.

**Clear to send CTS - 14 byte** Il nodo destinatario del frame RTS, conferma ed autorizza la richiesta inviando a sua volta un pacchetto piccolo, il Clear o send (CTS), al nodo che ha inviato la richiesta.

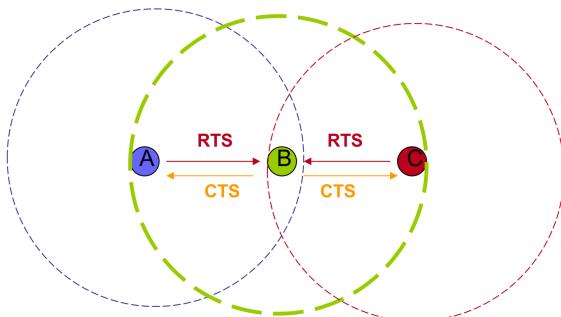


Figure 5.49: Soluzione nodo nascosto

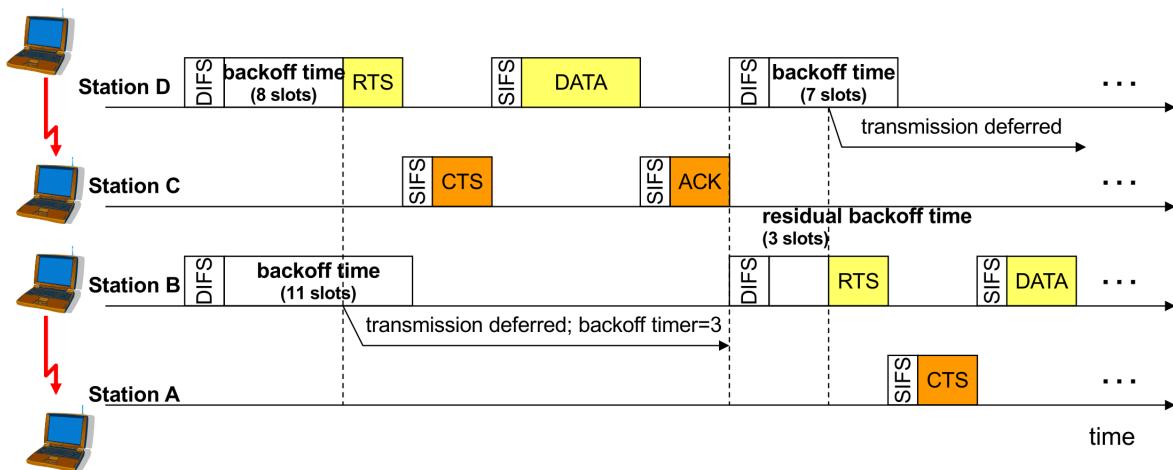
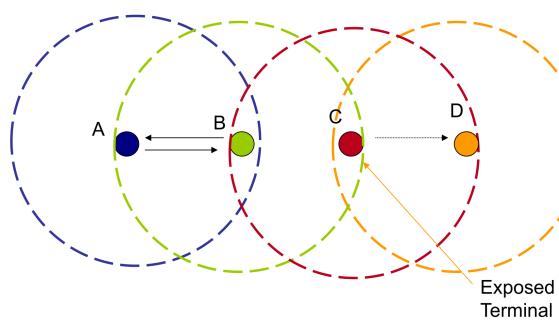


Figure 5.50: Handshake DCF per nodi nascosti

Esempio della trasmissione WLAN 802.11 con RTS e CTS. Di fatto cambia solo il tempo della duration, prima era composta da data, SIFS e ACK; adesso includo i campi RTS e CTS come in figura.



**Nodo esposto** A e B stanno comunicando ed occupano il canale; il nodo C vuole comunicare con il canale D ma sente, tramite rilevazione di canale, che questo è già occupato.

Questo si risolve avviando un backoff time(casuale, secondo un algoritmo) a fine comunicazione trasmessa con successo così che tra un pacchetto e l'altro, di quelli inviati da A e B, ci sia "spazio" per le comunicazioni di C e D, che rilevavano il canale occupato.

Figure 5.51: Nodo esposto

#### 5.5.4 Dispositivi wireless

##### Hub

L'hub è un dispositivo che riceve i dati da una stazione e li ritrasmette a tutte le altre stazioni collegate. In ambito wireless, l'hub non viene praticamente mai utilizzato perché non è in grado di gestire le collisioni e non offre alcuna forma di filtraggio o instradamento dei dati. In generale, l'hub lavora a livello fisico (livello 1 ISO/OSI).

##### Bridge

Il bridge è un dispositivo che collega due segmenti di rete, anche di tipo diverso (ad esempio una rete cablata e una wireless), filtrando il traffico in base agli indirizzi MAC. In ambito wireless, il bridge permette di collegare due reti wireless o una rete wireless e una cablata, inoltrando solo i frame destinati all'altro segmento. Opera a livello data link (livello 2 ISO/OSI).

##### Switch

Lo switch è un dispositivo che riceve i frame e li inoltra solo alla porta (o interfaccia) corretta, in base all'indirizzo MAC di destinazione. In ambito wireless, la funzione di switch è spesso integrata negli access point avanzati, che possono gestire più connessioni e segmentare il traffico tra i dispositivi collegati. Anche lo switch opera a livello data link (livello 2 ISO/OSI).

##### Modalità di funzionamento di uno switch

- **Store and Forward:**

Lo switch legge l'intero frame e verifica il campo FCS (scarta il frame se errato). Il frame viene memorizzato temporaneamente nel buffer associato alla porta d'uscita e trasmesso quando la linea della porta d'uscita è libera.

- **Cut and Through:**

Lo switch legge solo l'header del frame e inoltra immediatamente il frame senza memorizzazione intermedia. Il ritardo di elaborazione è indipendente dalla dimensione del frame.

- **Fragment Free:**

Lo switch legge solo i primi 64 byte del frame (dimensione minima frame Ethernet) per evitare di inoltrare frame che potrebbero essere coinvolti in collisioni. I frame di dimensione minore a 64 byte vengono scartati.