1. COMMUTAZIONE E PROTOCOLLI WIRELESS

1.1 Commutazione (Switching)

La commutazione è il processo mediante il quale i dati vengono trasferiti da un nodo all'altro in una rete. Esistono diverse tecniche di commutazione:

1.1.1 Commutazione di pacchetto

- I dati vengono suddivisi in pacchetti più piccoli
- Ogni pacchetto contiene intestazione (header) con informazioni di routing
- I pacchetti possono seguire percorsi diversi nella rete
- Vantaggi: utilizzo efficiente della larghezza di banda, resilienza ai guasti
- Svantaggi: possibile jitter (variazione del ritardo), ritardi variabili
- Utilizzata in Internet e nelle moderne reti di dati

1.1.2 Commutazione di circuito

- Viene stabilito un canale dedicato per l'intera durata della comunicazione
- La larghezza di banda è riservata anche quando non utilizzata
- Vantaggi: prestazioni costanti, nessun jitter
- Svantaggi: utilizzo inefficiente delle risorse
- Esempio classico: rete telefonica tradizionale

1.1.3 Commutazione di messaggio

- L'intero messaggio viene trasmesso da un nodo all'altro
- Ogni nodo intermedio riceve, memorizza e inoltra l'intero messaggio
- Vantaggi: adatto per messaggi brevi, tolleranza agli errori
- Svantaggi: elevata latenza, richiede molta memoria nei nodi
- Esempi: primi sistemi di posta elettronica, reti telegrafiche

1.2 Protocolli per LAN Wireless

Le reti wireless presentano problematiche specifiche che richiedono protocolli dedicati.

1.2.1 Problematiche delle LAN wireless

- **Stazione esposta**: un nodo si astiene dal trasmettere perché rileva una trasmissione in corso, ma non interferirebbe con essa

- **Stazione nascosta**: un nodo non rileva una trasmissione in corso e quindi trasmette, causando interferenze
- **Attenuazione del segnale**: diminuzione della potenza del segnale con la distanza
- **Interferenze**: disturbi causati da altre sorgenti di segnale
- **Sicurezza**: vulnerabilità legate alla propagazione del segnale in aria

1.2.2 MACA/MACAW (Multiple Access with Collision Avoidance)

- Risolve i problemi di stazione nascosta/esposta
- Utilizza pacchetti di controllo RTS (Request To Send) e CTS (Clear To Send)

RTS/CTS (Request To Send/Clear To Send)

- Il mittente invia un pacchetto RTS
- Il destinatario risponde con un pacchetto CTS
- Le altre stazioni che ricevono CTS attendono
- Riduce le collisioni causate da stazioni nascoste
- Sequenza:
  1. Il mittente invia RTS (inclusa durata prevista della trasmissione)
  2. Il destinatario risponde con CTS (inclusa durata)
  3. Le altre stazioni attendono per il tempo indicato
  4. Il mittente invia i dati

1.2.3 IEEE 802.11 (Wi-Fi)

- **802.11b**: 2.4 GHz, fino a 11 Mbps
- **802.11a**: 5 GHz, fino a 54 Mbps
- **802.11g**: 2.4 GHz, fino a 54 Mbps
- **802.11n**: 2.4/5 GHz, fino a 600 Mbps, MIMO
- **802.11ac**: 5 GHz, fino a 6.9 Gbps, MU-MIMO
- **802.11ax (Wi-Fi 6)**: 2.4/5/6 GHz, maggiore efficienza, OFDMA

1.3 Ethernet e codifica Manchester

Ethernet è uno standard per reti locali cablate, inizialmente sviluppato da Xerox e poi standardizzato come IEEE 802.3.

1.3.1 Codifica Manchester

- Tecnica di codifica usata in Ethernet
- Ogni bit è rappresentato da una transizione:
  - Da basso ad alto per bit 1
  - Da alto a basso per bit 0

- Vantaggi:
  - Sincronizzazione di clock incorporata
  - Rilevamento di errori
- Svantaggi: richiede il doppio della larghezza di banda

1.3.2 Algoritmo di Backoff in Ethernet

- Utilizzato quando viene rilevata una collisione
- Tempi di attesa casuali per evitare collisioni ripetute
- L'algoritmo di backoff esponenziale binario:
  1. Attendi K slot di tempo, dove K è un numero casuale tra 0 e (2^n-1)
  2. n è il numero di collisioni consecutive, fino a un massimo (10)
  3. Il tempo di attesa aumenta esponenzialmente con ogni collisione

1.3.3 Evoluzione di Ethernet

- **10Base5**: cavo coassiale spesso, 10 Mbps
- **10Base2**: cavo coassiale sottile, 10 Mbps
- **10Base-T**: doppino intrecciato, 10 Mbps
- **100Base-TX**: Fast Ethernet, 100 Mbps
- **1000Base-T**: Gigabit Ethernet, 1 Gbps
- **10GBase-T**: 10 Gigabit Ethernet, 10 Gbps

---

2. LIVELLO DI RETE (NETWORK LAYER)

2.1 Introduzione al Livello 3

Il livello di rete (layer 3) è responsabile del routing dei pacchetti dalla sorgente alla destinazione, attraversando potenzialmente diverse reti intermedie.

2.1.1 Funzioni principali

- Indirizzamento logico (IP)
- Routing
- Instradamento dei pacchetti
- Frammentazione e riassemblaggio
- Interconnessione di reti eterogenee

2.1.2 Posizionamento nel modello ISO/OSI

- Si trova tra il livello di collegamento dati (2) e il livello di trasporto (4)

- Il livello 3 è indipendente dalla tecnologia di trasmissione sottostante
- Si occupa della consegna end-to-end di pacchetti attraverso diverse reti

2.2 Tipi di Routing

2.2.1 Routing statico

- Le tabelle di routing sono configurate manualmente
- Nessun adattamento automatico ai cambiamenti della rete
- Vantaggi: overhead ridotto, maggiore sicurezza, prevedibilità
- Svantaggi: nessuna tolleranza ai guasti, richiede riconfigurazioni manuali
- Utilizzo: reti piccole, connessioni stabili, link di backup
- Esempio di configurazione in un router Cisco:

```
Router(config)# ip route 192.168.2.0 255.255.255.0 192.168.1.2
```

2.2.2 Routing dinamico

- Le tabelle di routing vengono create e aggiornate automaticamente
- Adattamento ai cambiamenti della topologia di rete
- Vantaggi: resilienza, scalabilità, adattabilità
- Svantaggi: overhead di calcolo e di traffico, convergenza
- Due approcci principali:
  - Distance Vector
  - Link State

2.2.3 Confronto tra routing statico e dinamico

Caratteristica	Routing Statico	Routing Dinamico
Configurazione	Manuale	Automatica
Adattabilità ai cambiamenti	Nessuna	Alta
Overhead di rete	Nessuno	Presente
Risorse richieste	Basse	Medio-alte
Scalabilità	Bassa	Alta
Complessità di configurazione	Bassa per reti piccole, alta per reti grandi	Media, indipendente dalla dimensione
Tempo di convergenza	Non applicabile	Variabile in base al protocollo

2.3 Algoritmi di Routing

2.3.1 Distance Vector (Bellman-Ford)

- Ogni router condivide la propria tabella di routing con i vicini
- Calcolo basato sulla "distanza" (numero di hop o costo)
- Memorizza solo il vettore distanza verso destinazioni note
- Vettore = [destinazione, distanza, next-hop]
- Aggiornamento periodico
- Vantaggi: semplice implementazione, basso overhead computazionale
- Svantaggi: convergenza lenta, count-to-infinity
- Protocolli che lo implementano: RIP, BGP (variante)

Count-to-infinity

- Problema del routing distance vector
- Quando un link fallisce, le informazioni errate possono propagarsi
- Soluzione parziale: Split horizon, Poisoned reverse

Split horizon

- Non annunciare le rotte apprese da un vicino indietro allo stesso vicino
- Impedisce loop semplici
- Non risolve completamente il problema count-to-infinity

Poisoned reverse

- Annuncia le rotte apprese da un vicino indietro allo stesso vicino ma con metrica infinita
- Più efficace del semplice split horizon
- Aiuta a risolvere loop più complessi

2.3.2 Link State (Dijkstra)

- Ogni router crea una mappa dell'intera rete
- Invia informazioni sullo stato dei propri link a tutti
- Calcola il percorso più breve usando l'algoritmo di Dijkstra
- Vantaggi: convergenza rapida, affidabilità, meno soggetto a loop
- Svantaggi: richiede più memoria e potenza di calcolo
- Protocolli che lo implementano: OSPF, IS-IS

Algoritmo di Dijkstra (pseudocodice)

```
function Dijkstra(Graph, source):
    // Inizializzazione
    for each vertex v in Graph:
        dist[v] := infinity
        prev[v] := undefined
        add v to Q
    dist[source] := 0

    // Algoritmo
    while Q is not empty:
        u := vertex in Q with min dist[u]
        remove u from Q

        for each neighbor v of u:
            alt := dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] := alt
                prev[v] := u

    return dist[], prev[]
```

Esempio di tabella di routing

```
Destinazione | Next Hop | Metrica | Interfaccia
-------------|----------|---------|------------
192.168.1.0  | -        | 0       | eth0
10.0.0.0     | 192.168.1.1 | 1    | eth0
172.16.0.0   | 192.168.1.254 | 2  | eth0
0.0.0.0      | 192.168.1.1 | 1    | eth0
```

Esempio di protocollo: OSPF (Open Shortest Path First)

- Protocollo link state
- Calcola percorsi più brevi con Dijkstra
- Diviso in aree gerarchiche per migliorare la scalabilità
- Scambia solo cambiamenti (LSA) e non l'intera tabella
- Supporta autenticazione e subnet mask variabili
- Calcola metriche basate sulla larghezza di banda

2.3.3 BGP (Border Gateway Protocol)

- Utilizzato tra sistemi autonomi (AS)

- Protocollo path vector (evoluzione del distance vector)
- Tiene traccia del percorso completo verso ogni destinazione
- Permette policy di instradamento basate su accordi economici
- Cruciale per il routing su Internet
- Distingue tra eBGP (tra AS diversi) e iBGP (all'interno dello stesso AS)
- Attributi principali: AS_PATH, NEXT_HOP, LOCAL_PREF, MED

2.3.4 ICMP (Internet Control Message Protocol)

- Protocollo di supporto per IP, non usato per trasporto di dati applicativi
- Funzioni principali:
  - Segnalazione errori (es. host irraggiungibile)
  - Diagnostica (es. ping, traceroute)
- Struttura pacchetto:
  - **Type**: tipo di messaggio (es. 0: Echo Reply, 8: Echo Request)
  - **Code**: sottotipo
  - **Checksum**: verifica integrità
  - **Data**: dipende dal tipo

Tipi di messaggi comuni:

Type	Code	Significato
0	0	Echo Reply (risposta al ping)
3	0-15	Destination Unreachable
8	0	Echo Request (ping)
11	0-1	Time Exceeded (usato da traceroute)
5	0-3	Redirect (cambia next-hop)

Comandi che utilizzano ICMP

- **ping**: verifica connettività verso un host

```
ping 192.168.1.1
```

- **traceroute** (Linux/macOS) o **tracert** (Windows): determina il percorso verso un host

```
traceroute google.com
```

2.4 Routing Mobile e Algoritmi di Congestione

2.4.1 Routing Mobile

- Gestisce la comunicazione con dispositivi in movimento
- Problematiche: handover, localizzazione, roaming
- Soluzioni:
  - Home Agent / Foreign Agent
  - Tunneling
  - Registrazione dinamica

Procedura di base in Mobile IP

1. Il dispositivo mobile contatta un Foreign Agent nella rete visitata
2. Il Foreign Agent contatta l'Home Agent nella rete di origine
3. L'Home Agent tunnelizza i pacchetti destinati al dispositivo mobile verso il Foreign Agent
4. Il Foreign Agent consegna i pacchetti al dispositivo mobile

2.4.2 Algoritmi di Congestione

Usati per prevenire e gestire la congestione nelle reti.

Leaky Bucket (Secchio che perde)

- Regola la velocità con cui i pacchetti vengono inoltrati
- Funzionamento:
  1. I pacchetti entrano nel "secchio" (buffer)
  2. Escono a velocità costante
  3. Se il secchio è pieno, i nuovi pacchetti vengono scartati
- Analogia: secchio con un foro sul fondo, acqua (pacchetti) esce a velocità costante
- Livella i burst di traffico
- Vantaggi: implementazione semplice, traffico regolare
- Svantaggi: limitazione della velocità anche con rete non congestionata

Token Bucket

- Più flessibile rispetto al Leaky Bucket
- Funzionamento:
  1. Token vengono generati a velocità costante
  2. Ogni pacchetto richiede un token per essere trasmesso
  3. Se non ci sono token, il pacchetto attende
  4. I token possono accumularsi fino a un massimo
- Permette burst controllati di traffico
- Vantaggi: permette picchi temporanei, efficiente
- Svantaggi: più complesso da implementare

Random Early Detection (RED)

- Evita la congestione scartando pacchetti in modo proattivo
- Funzionamento:
  1. Monitora la lunghezza media della coda
  2. Quando supera una soglia minima, inizia a scartare pacchetti con probabilità crescente
  3. Quando supera una soglia massima, scarta tutti i pacchetti
- Vantaggi: previene la sincronizzazione globale TCP, equità tra flussi
- Utilizzato nei router Internet moderni

---

3. GESTIONE DEGLI ERRORI A LIVELLO DATA LINK

La trasmissione a livello 2 include strategie per prevenire o correggere errori di bit nei frame e per controllare il flusso di dati.

3.1 Rilevamento degli errori

3.1.1 CRC (Cyclic Redundancy Check)

- Alta capacità di rilevazione, usato da Ethernet
- Basato su divisione polinomiale in campo binario
- Procedura:
  1. Aggiungere n bit zero ai dati (dove n è il grado del polinomio generatore)
  2. Dividere la stringa risultante per il polinomio generatore
  3. Il resto della divisione è il CRC
  4. Trasmettere i dati originali seguiti dal CRC
- Polinomi standard: CRC-16, CRC-32

3.1.2 Bit di parità

- **Parità semplice**: aggiunge un bit per rendere il numero totale di bit 1 pari (parità pari) o dispari (parità dispari)
- **Parità bidimensionale**: organizza i dati in una matrice e calcola la parità per ogni riga e colonna
- Vantaggi: semplicità
- Svantaggi: limitata capacità di rilevazione errori

3.1.3 Checksum

- Somma dei valori dei dati, eventualmente con complemento a uno

- Poco diffuso a livello data link, più comune a livello trasporto (TCP/UDP)
- Meno efficace del CRC per la rilevazione di errori

3.2 Correzione degli errori

3.2.1 FEC (Forward Error Correction)

- I bit di ridondanza permettono di correggere un numero limitato di errori senza ritrasmettere
- Esempi:
  - **Codici di Hamming**: possono correggere errori singoli
  - **Codici Reed-Solomon**: correggono burst di errori, usati in CD/DVD
  - **Codici convoluzionali**: usati in comunicazioni wireless
  - **Turbo codes**: alta efficienza, usati in 3G/4G/5G
- Adatto in contesti dove la ritrasmissione è costosa (es. collegamenti satellitari)

3.2.2 Hamming Distance

- Numero di bit che differiscono tra due sequenze
- Per rilevare d errori: serve una distanza minima d+1
- Per correggere d errori: serve una distanza minima 2d+1

3.3 Controllo di flusso

Il controllo di flusso è necessario per impedire che il mittente saturi i buffer del ricevitore.

3.3.1 Sliding Window

- Gestione dinamica di quanti frame possono essere inviati senza ACK
- Tipi:
  - **Stop-and-Wait**: invio di un solo frame, poi attesa ACK
  - **Go-back-N**: finestra di dimensione N, in caso di errore si ritrasmettono i frame non confermati
  - **Selective Repeat**: ritrasmette solo i frame effettivamente corrotti/persi

3.3.2 Stop-and-Wait

- Protocollo più semplice
- Invia un frame e attende conferma (ACK) prima di inviare il successivo
- Efficienza:

```
E = T_trasmissione / (T_trasmissione + RTT)
```

- Poco efficiente con RTT elevati
- Esempio: linea da 1 Mbps, frame da 1000 bit, RTT = 30 ms

```
E = 1ms / (1ms + 30ms) = 0.032 = 3.2%
```

3.3.3 Go-back-N

- Permette di inviare più frame prima di ricevere ACK
- In caso di errore, ritrasmette tutti i frame dall'ultimo confermato
- Dimensione finestra (N): in genere 2^m-1 dove m è il numero di bit per la numerazione
- Efficienza migliore con RTT elevati
- L'ACK k conferma tutti i frame fino a k-1

3.3.4 Selective Repeat

- Permette di inviare più frame prima di ricevere ACK
- In caso di errore, ritrasmette solo i frame persi
- Richiede buffering al ricevitore
- Efficienza massima
- Più complesso da implementare
- Dimensione finestra massima: 2^(m-1) per evitare ambiguità nei numeri di sequenza

3.4 Ritrasmissione e ACK/NACK

3.4.1 Timeout

- Se non arriva ACK entro un tempo limite, il frame è ritrasmesso
- Valore di timeout critico: troppo breve → ritrasmissioni inutili, troppo lungo → ritardi
- In genere: Timeout = RTT stimato + margine

3.4.2 ACK

- Conferma ricezione corretta
- Può essere cumulativo (conferma tutti i frame fino a un certo numero)
- Riduce l'overhead di rete

3.4.3 NACK

- Notifica un errore, sollecitando la ritrasmissione immediata
- Permette di ridurre i tempi di attesa per il timeout
- Non tutti i protocolli lo utilizzano

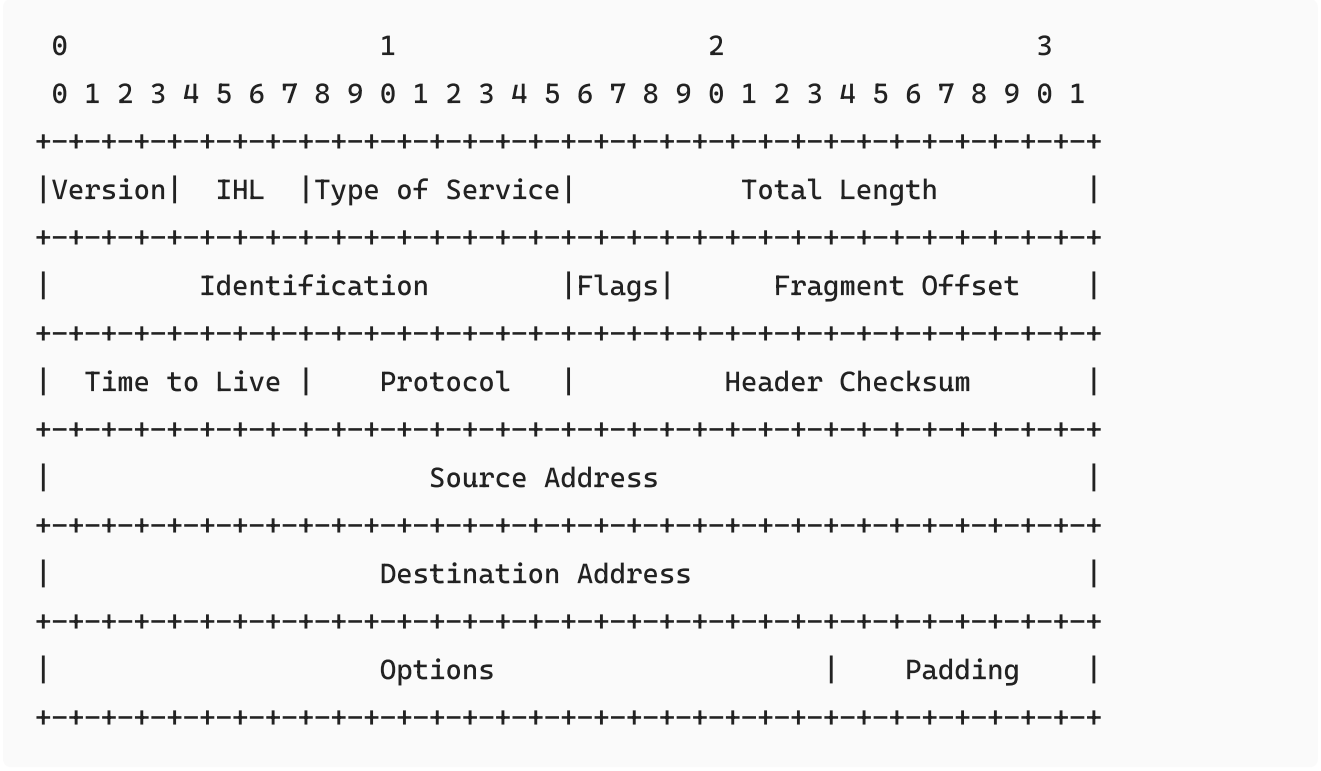
---

4. PROTOCOLLO IP (INTERNET PROTOCOL)

4.1 Struttura del pacchetto IP

4.1.1 Header IPv4

Un pacchetto IPv4 ha un header di 20-60 byte che include:



- **Version**: Versione del protocollo IP (4 per IPv4)
- **IHL** (Internet Header Length): Lunghezza dell'header in parole da 32 bit
- **Type of Service**: Priorità del pacchetto (oggi DSCP e ECN)
- **Total Length**: Lunghezza totale del pacchetto (header + dati)
- **Identification**: Identificatore per i frammenti appartenenti allo stesso datagramma
- **Flags**: Controllo frammentazione (DF: Don't Fragment, MF: More Fragments)
- **Fragment Offset**: Posizione del frammento nel datagramma originale
- **Time to Live (TTL)**: Numero massimo di hop prima di scartare il pacchetto
- **Protocol**: Protocollo di livello superiore (TCP=6, UDP=17, ICMP=1)
- **Header Checksum**: Verifica dell'integrità dell'header
- **Source Address**: Indirizzo IP sorgente (32 bit)
- **Destination Address**: Indirizzo IP destinazione (32 bit)
- **Options**: Opzioni (raramente usate)
- **Padding**: Byte di riempimento per allineare l'header a 32 bit

4.2 Classi di indirizzi IP

Gli indirizzi IPv4 sono divisi in classi basate sui primi bit:

4.2.1 Classe A

- Primo bit: 0
- Range: 0.0.0.0 - 127.255.255.255
- Maschera di rete: 255.0.0.0 (/8)
- Formato: N.H.H.H (N=Network, H=Host)
- 16,777,214 host per rete

4.2.2 Classe B

- Primi bit: 10
- Range: 128.0.0.0 - 191.255.255.255
- Maschera di rete: 255.255.0.0 (/16)
- Formato: N.N.H.H
- 65,534 host per rete

4.2.3 Classe C

- Primi bit: 110
- Range: 192.0.0.0 - 223.255.255.255
- Maschera di rete: 255.255.255.0 (/24)
- Formato: N.N.N.H
- 254 host per rete

4.2.4 Classe D (Multicast)

- Primi bit: 1110
- Range: 224.0.0.0 - 239.255.255.255
- Non utilizzata per indirizzare host
- Usata per trasmissioni multicast

4.2.5 Classe E (Riservata/Sperimentale)

- Primi bit: 1111
- Range: 240.0.0.0 - 255.255.255.255
- Riservata per utilizzo futuro

4.3 Indirizzi speciali

- **Loopback**: 127.0.0.0/8 (in particolare 127.0.0.1)
- **Broadcast**: xxx.xxx.xxx.255 (broadcast locale)
- **Broadcast di rete**: tutti i bit host a 1
- **Indirizzo di rete**: tutti i bit host a 0

- **Indirizzi privati**:
  - 10.0.0.0/8 (Classe A)
  - 172.16.0.0/12 (Classe B)
  - 192.168.0.0/16 (Classe C)
- **APIPA**: 169.254.0.0/16 (assegnazione automatica quando DHCP fallisce)
- **Multicast**: 224.0.0.0/4
  - 224.0.0.1: tutti gli host del segmento
  - 224.0.0.2: tutti i router del segmento

4.4 Subnetting e CIDR

4.4.1 Subnet Mask

- Maschera binaria che identifica la parte di rete e di host di un indirizzo
- Esempi:
  - 255.0.0.0 = /8
  - 255.255.0.0 = /16
  - 255.255.255.0 = /24
  - 255.255.255.240 = /28

4.4.2 CIDR (Classless Inter-Domain Routing)

- Supera il concetto di classi
- Notazione: indirizzo/prefisso
- Esempio: 192.168.1.0/24
- Vantaggi: utilizzo efficiente dello spazio di indirizzi
- Consente aggregazione di rotte (supernetting)

4.4.3 Calcolo del subnetting

1. Determinare quante subnet sono necessarie
2. Determinare quanti host per subnet
3. Calcolare il numero di bit da prendere in prestito dagli host
4. Calcolare la nuova subnet mask
5. Calcolare gli indirizzi di rete, broadcast e il range di host per ogni subnet

Esempio:

Suddividere 192.168.1.0/24 in 4 subnet

1. Per 4 subnet servono 2 bit (2^2 = 4)
2. La nuova subnet mask è /26 (24+2)
3. Le subnet risultanti sono:

- 192.168.1.0/26 (host: 192.168.1.1 - 192.168.1.62, broadcast: 192.168.1.63)
- 192.168.1.64/26 (host: 192.168.1.65 - 192.168.1.126, broadcast: 192.168.1.127)
- 192.168.1.128/26 (host: 192.168.1.129 - 192.168.1.190, broadcast: 192.168.1.191)
- 192.168.1.192/26 (host: 192.168.1.193 - 192.168.1.254, broadcast: 192.168.1.255)

4.4.4 VLSM (Variable Length Subnet Mask)

- Permette di utilizzare subnet mask di lunghezza variabile all'interno della stessa rete
- Vantaggi: utilizza lo spazio di indirizzi in modo più efficiente
- Esempio: una rete con sottoreti di dimensioni diverse (punto-punto vs grandi LAN)

4.5 IPv6

IPv6 è la nuova versione del protocollo IP, progettata per sostituire IPv4.

4.5.1 Caratteristiche principali

- Indirizzi a 128 bit (2^128 indirizzi disponibili)
- Header semplificato
- Supporto integrato per sicurezza (IPsec)
- Nessuna frammentazione a livello di router
- Nessun checksum nell'header

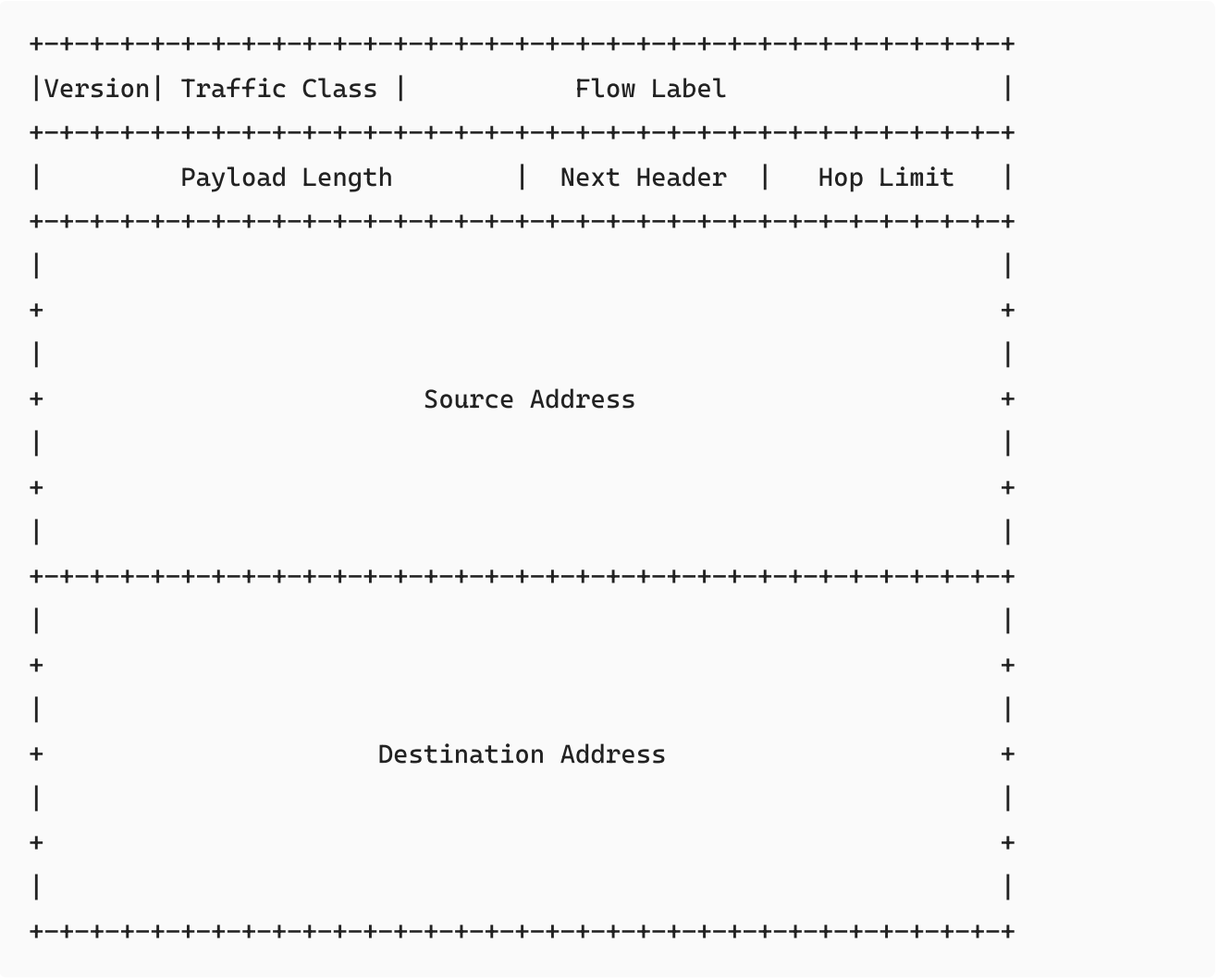
4.5.2 Formato degli indirizzi IPv6

- 8 gruppi di 4 cifre esadecimali separati da ":"
- Esempio: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
- Regole di semplificazione:
  - I gruppi di zeri possono essere omessi: 2001:0db8:85a3::8a2e:0370:7334
  - Gli zeri iniziali in ogni gruppo possono essere omessi: 2001:db8:85a3::8a2e:370:7334
  - Solo una sequenza di zeri può essere abbreviata con "::"

4.5.3 Tipi di indirizzi IPv6

- **Unicast**: identifica una singola interfaccia
  - **Global Unicast**: equivalenti agli indirizzi pubblici IPv4, iniziano con 2000::/3
  - **Link-Local**: validi solo nel link locale, iniziano con fe80::/10
  - **Unique Local**: equivalenti agli indirizzi privati IPv4, iniziano con fc00::/7
  - **Loopback**: ::1/128 (equivalente a 127.0.0.1 in IPv4)
- **Multicast**: identifica un gruppo di interfacce, iniziano con ff00::/8
- **Anycast**: identifica un gruppo di interfacce, ma il pacchetto viene inviato solo alla più vicina

4.5.4 Header IPv6



- **Version**: Sempre 6 per IPv6
- **Traffic Class**: Priorità del pacchetto (simile al Type of Service in IPv4)
- **Flow Label**: Etichetta per identificare pacchetti dello stesso flusso
- **Payload Length**: Lunghezza del payload (escluso header principale)
- **Next Header**: Tipo di header seguente (estensione o protocollo di livello superiore)
- **Hop Limit**: Equivalente al TTL in IPv4
- **Source Address**: Indirizzo IPv6 sorgente (128 bit)
- **Destination Address**: Indirizzo IPv6 destinazione (128 bit)

4.5.5 Extension Header

In IPv6, funzionalità aggiuntive sono implementate tramite header di estensione:

- **Hop-by-Hop Options**: opzioni per ogni nodo nel percorso
- **Routing**: routing source-routed
- **Fragment**: informazioni di frammentazione
- **Authentication (AH)**: integrità e autenticazione (IPsec)
- **Encapsulating Security Payload (ESP)**: cifratura (IPsec)

- **Destination Options**: opzioni solo per il nodo destinazione

4.5.6 Transizione da IPv4 a IPv6

Tecniche per la coesistenza e migrazione:

- **Dual Stack**: supporto simultaneo di IPv4 e IPv6
- **Tunneling**: incapsulamento di pacchetti IPv6 in pacchetti IPv4
  - **6to4**: automatico, usa prefisso 2002::/16
  - **6in4**: tunnel configurato manualmente
  - **Teredo**: attraversa NAT, usa prefisso 2001::/32
- **Translation**: traduzione diretta tra pacchetti IPv4 e IPv6
  - **NAT64/DNS64**: permette a client IPv6 di comunicare con server IPv4

4.6 Protocolli ausiliari di livello 3

4.6.1 ARP (Address Resolution Protocol)

- Risolve un indirizzo IP in un indirizzo MAC
- Funzionamento:
  1. Il mittente invia un broadcast ARP ("Chi ha questo IP?")
  2. Il destinatario risponde ("Io ho questo IP, questo è il mio MAC")
  3. Il mittente memorizza l'associazione IP-MAC nella sua cache ARP
- Struttura pacchetto ARP:
  - Hardware Type (Ethernet = 1)
  - Protocol Type (IPv4 = 0x0800)
  - Hardware Address Length (6 per MAC)
  - Protocol Address Length (4 per IPv4)
  - Operation (1 = request, 2 = reply)
  - Sender Hardware Address (MAC mittente)
  - Sender Protocol Address (IP mittente)
  - Target Hardware Address (MAC destinatario)
  - Target Protocol Address (IP destinatario)

4.6.2 RARP (Reverse ARP)

- Funzione opposta ad ARP
- Risolve un indirizzo MAC in un indirizzo IP
- Usato principalmente per boot diskless
- Sostituito da protocolli più moderni (BOOTP, DHCP)

4.6.3 DHCP (Dynamic Host Configuration Protocol)

- Assegna automaticamente indirizzi IP e altre configurazioni
- Processo in 4 fasi:
  1. **DISCOVER**: client broadcast per trovare server DHCP
  2. **OFFER**: server offre un indirizzo IP
  3. **REQUEST**: client richiede l'indirizzo offerto
  4. **ACK**: server conferma l'assegnazione
- Oltre all'indirizzo IP fornisce:
  - Subnet mask
  - Gateway predefinito
  - Server DNS
  - Lease time (tempo di validità dell'assegnazione)

4.6.4 ICMP (Internet Control Message Protocol)

- Usato per diagnostica e reporting di errori
- Applicazioni:
  - **Ping**: verifica raggiungibilità (Echo Request/Reply)
  - **Traceroute**: traccia percorso (Time Exceeded)
  - **Path MTU Discovery**: determina MTU del percorso
- Tipi di messaggi comuni:
  - Echo Request/Reply (ping)
  - Destination Unreachable
  - Time Exceeded
  - Redirect

4.6.5 NAT (Network Address Translation)

- Permette a una rete privata di condividere un singolo indirizzo IP pubblico
- Funzioni:
  - Conservazione indirizzi IP pubblici
  - Sicurezza (nasconde struttura interna)
  - Facilita cambio di ISP (solo indirizzi pubblici cambiano)
- Tipi:
  - **Static NAT**: mappatura 1:1 tra IP privati e pubblici
  - **Dynamic NAT**: pool di indirizzi pubblici assegnati dinamicamente
  - **PAT/NAPT**: mappatura molti:1 usando diverse porte

PAT (Port Address Translation)

- Variante del NAT
- Usa porte TCP/UDP per mappare più host interni su un singolo IP pubblico

- Anche detto NAT overload o NAPT
- È il tipo più comune nelle reti domestiche e piccole aziende

---

5. LIVELLO DI TRASPORTO

5.1 Funzioni del livello di trasporto

- Trasporto end-to-end tra processi applicativi
- Multiplazione/demultiplazione tra applicazioni
- Controllo di flusso
- Controllo della congestione
- Recupero errori (facoltativo)
- Segmentazione e riassemblaggio

5.1.1 Posizionamento nel modello ISO/OSI

- Si trova tra il livello di rete (3) e il livello di sessione (5)
- Primo livello end-to-end (i livelli 1-3 operano hop-by-hop)
- Nasconde dettagli della rete sottostante alle applicazioni

5.1.2 Servizi offerti

- **Connection-oriented**: stabilisce una connessione prima del trasferimento dati
- **Connectionless**: invia dati senza stabilire una connessione
- **Reliable**: garantisce consegna dei dati
- **Unreliable**: nessuna garanzia di consegna
- **Stream-oriented**: trattamento dei dati come flusso continuo di byte
- **Message-oriented**: trattamento dei dati come messaggi discreti

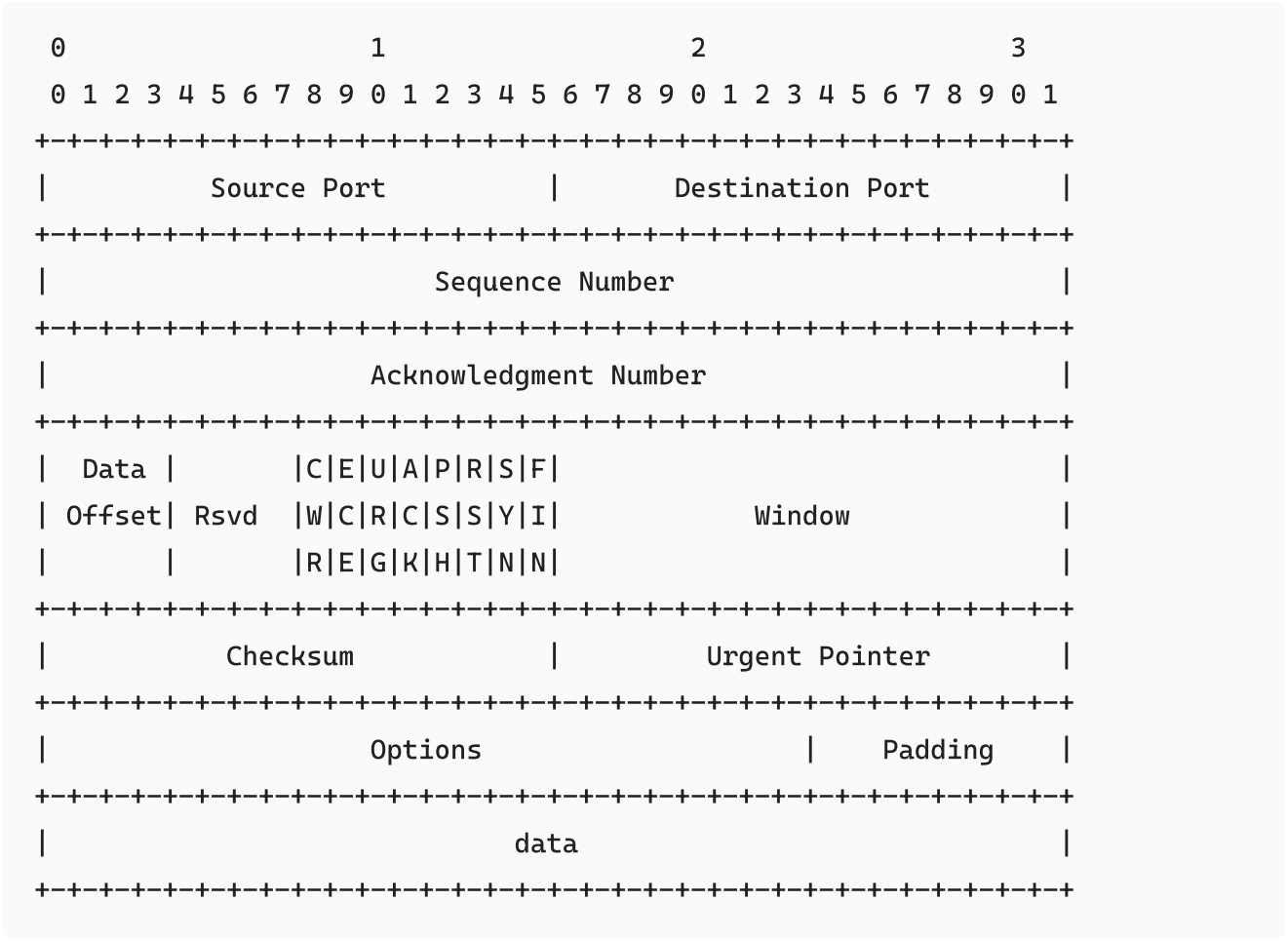
5.2 TCP (Transmission Control Protocol)

5.2.1 Caratteristiche

- Connessione orientata
- Affidabile
- Ordinamento garantito
- Controllo di flusso
- Controllo della congestione
- Bidirectional byte stream
- Full duplex
- Utilizzo:

- Web (HTTP/HTTPS)
- Email (SMTP, POP3, IMAP)
- File transfer (FTP)
- Remote login (SSH)

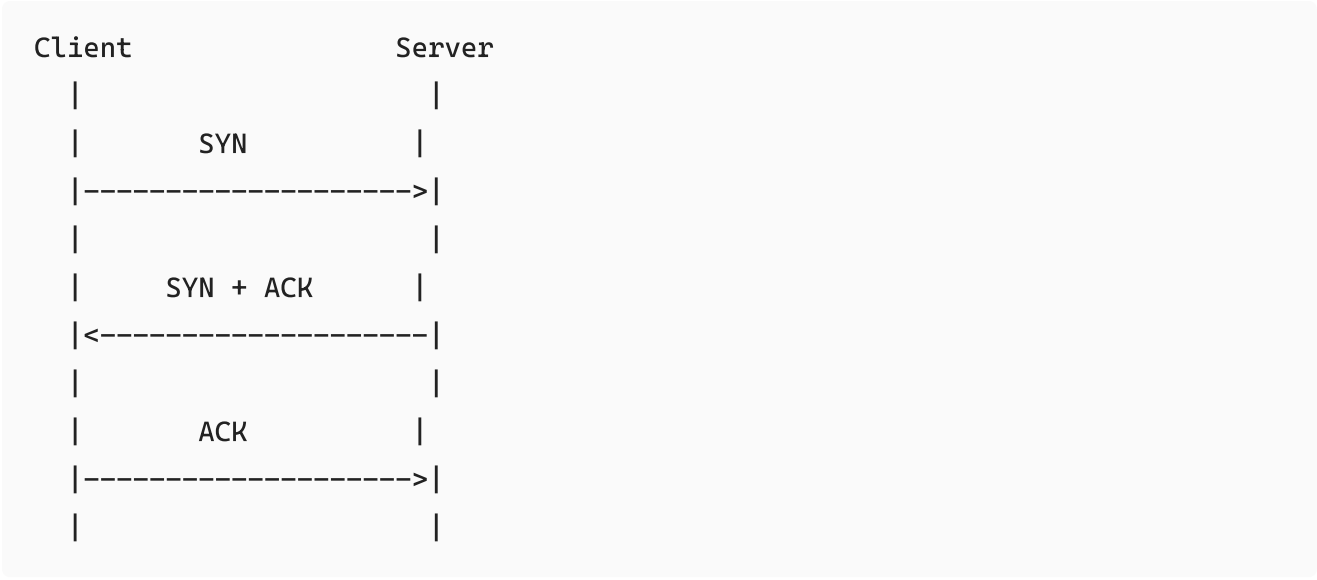
5.2.2 Struttura del segmento TCP



- **Source Port, Destination Port**: identificano le applicazioni
- **Sequence Number**: numero di sequenza del primo byte di dati
- **Acknowledgment Number**: prossimo byte atteso
- **Data Offset**: lunghezza header in parole di 32 bit
- **Flags**: bit di controllo
  - URG: campo Urgent Pointer valido
  - ACK: campo Acknowledgment valido
  - PSH: Push function (consegna immediata)
  - RST: Reset della connessione
  - SYN: Sincronizzazione dei numeri di sequenza
  - FIN: Fine dei dati
- **Window**: dimensione della finestra di ricezione
- **Checksum**: integrità del segmento
- **Urgent Pointer**: offset dei dati urgenti

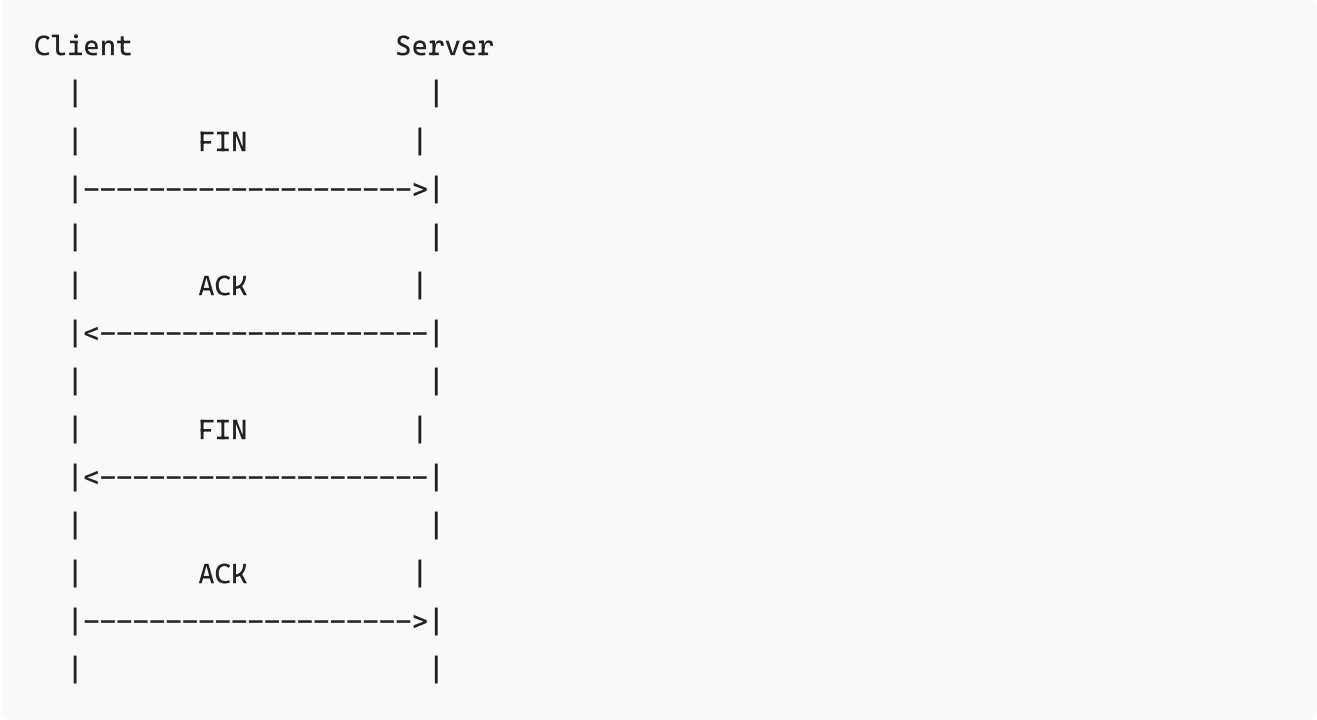
- **Options**: opzioni (MSS, window scaling, timestamp)

5.2.3 Three-way handshake (apertura connessione)



1. **Client** → **Server**: SYN, seq=x
2. **Server** → **Client**: SYN+ACK, seq=y, ack=x+1
3. **Client** → **Server**: ACK, seq=x+1, ack=y+1

5.2.4 Chiusura connessione (Four-way handshake)



1. **Client** → **Server**: FIN, seq=x
2. **Server** → **Client**: ACK, ack=x+1
3. **Server** → **Client**: FIN, seq=y
4. **Client** → **Server**: ACK, ack=y+1

5.2.5 Stati di una connessione TCP

- **CLOSED**: nessuna connessione
- **LISTEN**: in attesa di connessione
- **SYN-SENT**: inviato SYN, attesa risposta
- **SYN-RECEIVED**: ricevuto SYN, inviato SYN+ACK
- **ESTABLISHED**: connessione stabilita
- **FIN-WAIT-1**: inviato FIN
- **FIN-WAIT-2**: ricevuto ACK per FIN
- **CLOSE-WAIT**: ricevuto FIN, invio ACK
- **LAST-ACK**: inviato FIN, attesa ultimo ACK
- **TIME-WAIT**: attesa tempo 2MSL dopo chiusura
- **CLOSING**: inviato e ricevuto FIN contemporaneamente

5.2.6 Controllo di flusso

- Evita di sovraccaricare il ricevitore
- Window Size nel header TCP indica quanti byte il ricevitore può accettare
- Il mittente non può inviare più dati di quanto indicato nella finestra
- Window scaling (opzione TCP) permette finestre fino a 1GB

5.2.7 Controllo della congestione

Algoritmi per prevenire sovraccarico della rete:

Slow Start

- Inizia con una piccola finestra di congestione (1 MSS)
- Raddoppia ad ogni RTT
- Cresce esponenzialmente fino a raggiungere la soglia (ssthresh)
- Formula: cwnd = cwnd + MSS per ogni ACK ricevuto

Congestion Avoidance

- Inizia dopo lo Slow Start
- Incremento lineare della finestra (1 MSS per RTT)
- Formula: cwnd = cwnd + MSS * (MSS/cwnd) per ogni ACK ricevuto
- Entro in questa fase dopo aver raggiunto la soglia

Fast Retransmit

- Ritrasmissione rapida in caso di 3 ACK duplicati
- Non attende timeout, migliorando l'efficienza

- Indica perdita di singoli segmenti, non congestione grave

Fast Recovery

- Dopo Fast Retransmit, evita di ricominciare da Slow Start
- Dimezza la finestra di congestione e passa a Congestion Avoidance
- Permette di mantenere un throughput migliore durante perdite occasionali

5.2.8 Parametri di connessione TCP

- **RTT (Round Trip Time)**: tempo di andata e ritorno
- **RTO (Retransmission Timeout)**: timeout per ritrasmissione
- **MSS (Maximum Segment Size)**: dimensione massima dati in un segmento
- **MTU (Maximum Transmission Unit)**: dimensione massima frame a livello 2
- **Bandwidth-delay product**: quantità di dati "in volo" = bandwidth * RTT

5.2.9 Problemi tipici

- **Slow Start**: avvio lento della connessione
- **Head of Line Blocking**: pacchetti bloccati in attesa di quelli persi
- **Fairness**: equità nell'allocazione della banda
- **Bufferbloat**: buffer eccessivi che aumentano latenza
- **RTT fairness**: connessioni con RTT diversi ricevono bandwidth diverse

5.3 UDP (User Datagram Protocol)

5.3.1 Caratteristiche

- Connectionless (senza connessione)
- Non affidabile (nessuna garanzia di consegna)
- Nessun ordinamento garantito
- Nessun controllo di flusso o congestione
- Overhead minimo
- Latenza potenzialmente inferiore
- Utilizzo:
  - DNS
  - Streaming video
  - VoIP
  - Online gaming
  - IoT e sensori

5.3.2 Struttura del datagramma UDP

```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|                                   |
|              Data                 |
+-----------------------------------+
```

- **Source Port**: porta di origine (facoltativa, 0 se non usata)
- **Destination Port**: porta di destinazione
- **Length**: lunghezza totale (header + dati)
- **Checksum**: verifica integrità (facoltativa in IPv4, obbligatoria in IPv6)

5.3.3 UDP Lite

- Variante di UDP (RFC 3828)
- Checksum parziale che protegge solo header e parte iniziale dei dati
- Utile per applicazioni multimedia che possono tollerare errori nei dati

5.3.4 Confronto UDP vs TCP

Caratteristica	TCP	UDP
Connessione	Connection-oriented	Connectionless
Affidabilità	Garantita	Non garantita
Ordinamento	Garantito	Non garantito
Controllo flusso	Sì	No
Controllo congestione	Sì	No
Overhead	Alto	Basso
Velocità	Potenzialmente più lenta	Potenzialmente più veloce
Uso bandwidth	Più efficiente	Meno efficiente
Applicazioni tipiche	Web, email, file transfer	Streaming, gaming, DNS

5.4 Porte e Socket

5.4.1 Concetto di porta

- Identificatore numerico (0-65535) per processi applicativi
- Permette multiplazione/demultiplazione
- Categorie:
  - **Well-known ports** (0-1023): servizi standard (HTTP=80, HTTPS=443)
  - **Registered ports** (1024-49151): applicazioni registrate IANA
  - **Dynamic/private ports** (49152-65535): allocate dinamicamente

5.4.2 Socket

- Endpoint di comunicazione
- Identificato da indirizzo IP + porta
- API socket: interfaccia di programmazione per comunicazione di rete
- Tipi principali:
  - **Stream socket**: basati su TCP
  - **Datagram socket**: basati su UDP
  - **Raw socket**: accesso diretto al livello IP

Funzioni tipiche API socket

- `socket()`: crea un nuovo socket
- `bind()`: associa un socket a un indirizzo
- `listen()`: predispone un socket per accettare connessioni
- `accept()`: accetta una connessione
- `connect()`: inizia una connessione
- `send()/recv()`: invia/riceve dati
- `close()`: chiude un socket
