# **APPUNTI TCP - STRUTTURA SEGMENTO E HANDSHAKE**

================================================================================

## **COS'È TCP?**

TCP (Transmission Control Protocol) è un protocollo connection-oriented.

Caratteristiche:

* Affidabile: garantisce che i dati arrivino correttamente
* Ordinato: i dati arrivano nella sequenza giusta
* Full-duplex: comunicazione bidirezionale contemporanea

================================================================================

## **STRUTTURA DEL SEGMENTO TCP**

Il segmento (o TPDU) è il messaggio scambiato a livello TCP.

Campi principali:

* Source/Destination Port: identificano le applicazioni
* Sequence Number: identifica i byte inviati
* Acknowledgment Number: conferma i byte ricevuti
* Flags (SYN, ACK, FIN, RST): controllano la connessione
* Window Size: controllo di flusso
* Checksum: verifica integrità

================================================================================

## **I FLAG TCP**

SYN (Synchronize)

* Valore: SYN = 1
* Uso: apertura connessione
* Significato: "Voglio connettermi e sincronizzare"

ACK (Acknowledgment)

* Valore: ACK = 1
* Uso: conferme
* Significato: "Ho ricevuto correttamente"

FIN (Finish)

* Valore: FIN = 1
* Uso: chiusura connessione
* Significato: "Ho finito, voglio chiudere"

RST (Reset)

* Valore: RST = 1
* Uso: chiusura immediata/rifiuto
* Significato: "Chiudo immediatamente"

================================================================================

## **THREE-WAY HANDSHAKE**

È la procedura a 3 passi per stabilire la connessione.

Obiettivi:

* Sincronizzare i numeri di sequenza iniziali
* Concordare parametri (MSS, Window Size)
* Stabilire connessione full-duplex

PASSO 1 - Client → Server

* SYN = 1, SEQ = X
* "Voglio connettermi, parto dal byte X"
* Primitiva: connect()

PASSO 2 - Server → Client

* SYN = 1, ACK = 1, SEQ = Y, ACK.N = X+1
* "Accetto! Parto da Y, ho ricevuto X"
* Primitiva: send()
* Stabilisce anche la direzione inversa (full-duplex)

PASSO 3 - Client → Server

* ACK = 1, ACK.N = Y+1
* "Ok, ho ricevuto Y"
* Primitiva: send()
* ✅ CONNESSIONE ATTIVA

Schema visivo:

CLIENT SERVER |──── SYN=1, SEQ=X ───────────>| |<─── SYN=1, ACK=1, SEQ=Y ─────| | ACK.N=X+1 | |──── ACK=1, ACK.N=Y+1 ────────>| └──── CONNESSIONE ATTIVA ───────┘

Rifiuto connessione: Se non c'è processo in ascolto → server invia RST = 1

================================================================================

## **PARAMETRI SCAMBIATI**

1. MSS (Maximum Segment Size)

* Dimensione massima segmenti TCP
* Formula: MSS = min(MTU, MRU) - 20 byte
* MTU = dimensione max frame Data Link (es. Ethernet = 1500 byte)
* MRU = MTU del destinatario
* Default: 536 byte (576 - 20 - 20)
* Si sceglie il valore minore tra i due host

1. Window Size

* Dimensione buffer di ricezione
* Indica quanti byte può accettare il destinatario
* Serve per controllo di flusso

1. Sequence Number iniziale

* Numero casuale da cui partire
* Motivo: sicurezza (protegge da Sequence Guessing)
* Serve per tracciare, riordinare e rilevare duplicati

================================================================================

## **PRIMITIVE TCP**

connect() → Connection Request (SYN) → client richiede connessione accept() → Nessuna → server accetta richieste send() → Data o SYN-ACK → invia dati/risposte

TPDU = Transport Protocol Data Unit = segmento TCP

================================================================================

## **ESEMPIO PRATICO**

Valori: Client SEQ=1000, Server SEQ=5000, MSS=1460, WS client=65535, WS server=8192

1. CLIENT → SERVER: SYN=1, SEQ=1000, MSS=1460, WS=65535
2. SERVER → CLIENT: SYN=1, ACK=1, SEQ=5000, ACK.N=1001, MSS=1460, WS=8192
3. CLIENT → SERVER: ACK=1, ACK.N=5001

Risultato: MSS=1460 (minore), client invia max 8192 byte, server max 65535 byte

================================================================================

## **CONFRONTO TRA I PASSI**

| PASSO 1 | PASSO 2 | PASSO 3

-----------|------------|---------------|------------- Chi | Client | Server | Client SYN | 1 | 1 | 0 ACK | 0 | 1 | 1 SEQ | X | Y | - ACK.N | - | X+1 | Y+1 Scopo | Richiesta | Accettazione | Conferma

================================================================================

## **PUNTI CHIAVE**

Concetti base:

* TCP è connection-oriented con handshake
* Segmento TCP = TPDU
* Flag controllano la connessione
* Connessione full-duplex

Three-Way Handshake:

1. Client: SYN=1, SEQ=X
2. Server: SYN=1, ACK=1, SEQ=Y, ACK.N=X+1
3. Client: ACK=1, ACK.N=Y+1

Parametri:

* Sequence Number: casuale per sicurezza
* MSS: min(MTU, MRU) - 20 byte, default 536 byte
* Window Size: dimensione buffer

Definizioni:

* MSS: dimensione max segmento TCP
* MTU: dimensione max frame Data Link (1500 byte Ethernet)
* MRU: MTU destinatario
* SYN: sincronizzazione/apertura
* ACK: conferma ricezione
* FIN: chiusura
* RST: reset/rifiuto

Primitive:

* connect(): apre (client)
* accept(): accetta (server)
* send(): invia

================================================================================

## **SCHEMA RIASSUNTIVO**

STRUTTURA SEGMENTO • Source/Destination Port • Sequence Number • Acknowledgment Number • FLAGS: SYN, ACK, FIN, RST • Window Size • Checksum

THREE-WAY HANDSHAKE

1. Client → Server: SYN=1, SEQ=X
2. Server → Client: SYN=1, ACK=1, SEQ=Y, ACK.N=X+1
3. Client → Server: ACK=1, ACK.N=Y+1 ✅ CONNESSIONE ATTIVA

PARAMETRI SCAMBIATI • Sequence Number iniziale (casuale) • MSS = min(MTU,MRU) - 20 byte • Window Size (buffer)

================================================================================

| **FLAG** | **NOME** | **QUANDO** | **SIGNIFICATO** |
| --- | --- | --- | --- |
| **SYN** | Synchronize | Apertura connessione | "Voglio connettermi" |
| **ACK** | Acknowledgment | Conferme | "Ho ricevuto correttamente" |
| **FIN** | Finish | Chiusura connessione | "Ho finito, chiudo" |
| **RST** | Reset | Chiusura anomala | "Chiudo immediatamente" |