**L’Handshaking TCP.**

**Comunicazione TCP connection-oriented:**

La comunicazione TCP tra host mittente (client) e host destinatario (server) è di tipo connection-oriented e si articola in tre fasi principali:

1. **Instaurazione della sessione (Three-Way Handshake)** Scambio iniziale di parametri tra client e server per avviare la connessione.
2. **Trasmissione dati** Invio affidabile delle informazioni, con controllo di errori, flusso e congestione.
3. **Abbattimento della sessione (Doublé Two-Way Handshake)** Chiusura della connessione, che può essere simultanea (Three-Way) o asincrona (Doublé Two-Way).

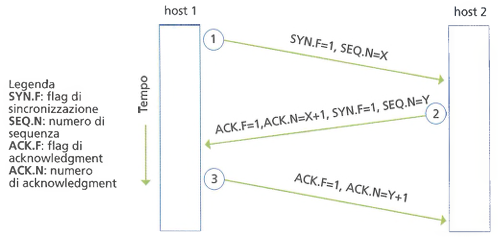
**Primitive TCP e TPDU:**

La comunicazione avviene tramite **Socket** e primitive, che generano messaggi chiamati **TPDU (Transport Protocol Data Unit)** o segmenti TCP:

| **Primitiva** | **TPDU inviata** | **Descrizione** |
| --- | --- | --- |
| Accept () | Nessuna | Attende richieste di connessione |
| Connect () | Connection Request | Prova a stabilire una connessione |
| Send () | Data | Invio di informazioni |
| Receive () | Nessuna | Attende ricezione di dati |
| Disconnect () | Disconnection Request | Rilascio della connessione |

**Three-Way Handshake: Fasi dettagliate:**

1. **Client → Server** Invio segmento TCP con flag SYN = 1 e numero di sequenza casuale X. → Primitiva: connect ()
2. **Server → Client** Risposta con ACK = 1, acknowledgment number X + 1, e proprio SYN = 1 con nuovo sequence number Y. → Primitiva: send ()
3. **Client → Server** Conferma finale con ACK = 1 e acknowledgment number Y + 1. → Primitiva: send ()



Se il server non ha un processo in ascolto sulla porta indicata, risponde con RST = 1 per rifiutare la connessione.

**Parametri di configurazione scambiati:**

* **Sequence Number iniziale**: ogni host comunica il proprio numero di sequenza.
* **MSS (Maximum Segment Size)**: Dimensione massima del segmento TCP, calcolata come:

MSS = min (MTU, MRU) − 20 byte

* Default: 536 byte (576 - 20 IP header - 20 TCP header)
* **Window Size**: Indica la dimensione del buffer TCP per la ricezione.
* **MTU (Maximum Transfer Unit)**: Dimensione massima del campo dati nel frame Data Link (es. Ethernet = 1500 byte).
* **MRU (Maximum Receive Unit)**: MTU del destinatario.

TCP adatta la dimensione dei segmenti per evitare frammentazioni IP e migliorare le prestazioni.

**Trasmissione dei dati: Protocollo Sliding Window**

TCP utilizza il meccanismo **sliding window** per:

* Garantire la consegna affidabile dei dati
* Mantenere la corretta sequenza
* Controllare il flusso tra mittente e destinatario

Differenze rispetto al livello Data Link:

* TCP gestisce la finestra a livello di **byte**, non di frame
* La **dimensione della finestra** in TCP è **variabile**, non fissa

**Gestione del buffer e controllo di flusso:**

* Il destinatario comunica al mittente quanti byte può ricevere tramite il campo **Window Size** nell’ACK.
* Il mittente **non deve superare** il valore indicato.
* Il destinatario **non può rifiutare** byte che ha dichiarato di poter accettare.

Esempio:

* Buffer ricezione = 2000 byte
* Occupato = 1000 byte → ACK con Window Size = 1000
* Mittente invia 700 byte
* Ricevente conferma 500 byte → nuova finestra ≥ 200 byte (per evitare perdita dei 200 byte ancora in transito)

Se il buffer è pieno → ACK con Window Size = 0 → il mittente **sospende la trasmissione**

**Gestione degli errori: Go-Back-N con timeout:**

TCP **non usa NACK** (Not Acknowledged), ma:

* Imposta un **timeout** per ogni segmento inviato
* Se il timer scade → ritrasmissione del segmento **non confermato** e di **tutti i successivi**, anche se alcuni sono già arrivati

Il mittente mantiene in memoria i segmenti non ancora riscontrati.

**Segmenti persi, duplicati e fuori sequenza:**

* Segmenti **danneggiati** → scartati grazie al campo **checksum**
* Segmenti **fuori ordine** → il destinatario invia ACK per l’ultimo segmento valido ricevuto
* Segmenti **duplicati** → scartati se il numero di sequenza è già stato ricevuto

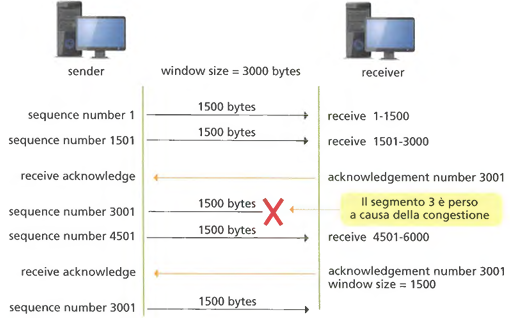
Se un ACK non arriva → TCP non se ne accorge, perché il **riscontro è cumulativo**:

* Es. ACK = 1301 → tutti i byte fino al 1300 sono stati ricevuti
* La perdita di un ACK intermedio (es. 1151) è **irrilevante**

**Relazione tra finestra, congestione e perdita:**

La **dimensione della finestra** influisce direttamente sulla gestione della congestione e sulla probabilità di perdita dei segmenti. L’esempio in figura 17 mostra come:

* La perdita di un segmento (es. il terzo) causa la ritrasmissione di tutti i successivi
* Il mittente riceve ACK duplicati e interpreta la situazione come errore



**Abbattimento della sessione TCP: Doublé Two-Way Handshake:**

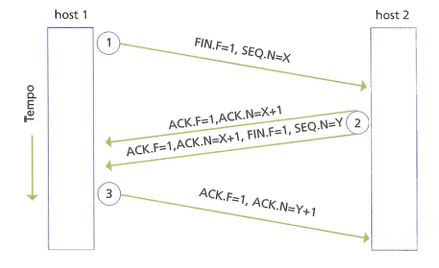
**Connessione bidirezionale:**

Una connessione TCP è composta da due flussi indipendenti (uno per direzione). La chiusura avviene separatamente per ciascun flusso.

**Procedura di chiusura:**

1. Il programma applicativo comunica al TCP di chiudere la connessione.
2. TCP trasmette i dati residui e riceve conferma (ACK).
3. Invia un segmento con **flag FIN = 1** per segnalare la chiusura.
4. Il destinatario conferma con **ACK = 1** e **ACK.N = X + 1**, ma può continuare a inviare dati se ne ha ancora.
5. Quando anche il secondo flusso termina, si invia un secondo **FIN** e relativo **ACK**.

Questo processo prende il nome di **Doublé Two-Way Handshake**.



**Particolarità rispetto al Three-Way Handshake:**

* Si usa il **flag FIN** invece di SYN.
* Il secondo passo è diviso in due tempi: ACK immediato e FIN successivo, dopo che il software applicativo è stato informato.

**Chiusura forzata: Reset.**

In caso di errore o anomalia, TCP può inviare un segmento con **flag RST = 1**, che chiude la connessione **immediatamente**, senza ulteriori scambi.

**Vulnerabilità delle connessioni TCP**

**SYN-FLOODING:**

* Attacco DoS che sfrutta il **Three-Way Handshake**.
* L’attaccante invia molti segmenti **SYN** senza completare la connessione.
* Il server resta bloccato in attesa degli ACK, saturando la coda di connessioni.

Contromisure:

* Aumentare la **SYN ACK queue**
* Ridurre il **timeout** per completare la connessione

**SEQUENCE GUESSING:**

* L’attaccante **indovina il numero di sequenza** e invia segmenti TCP falsificati ma formalmente corretti.
* Deve impedire che il vero mittente riceva gli ACK.

Contromisure:

* **Ingress filtering**: bloccare traffico interno falsificato a livello di router

**SESSION HIJACKING:**

* L’attaccante **intercetta** la sessione tra due host e si sostituisce a uno di essi.
  + Registra i numeri di sequenza
  + Blocca uno degli host (es. con SYN-Flooding)
  + Invia segmenti con source falsificato e sequenza corretta

Contromisure:

* **Ingress filtering** per impedire traffico interno contraffatto