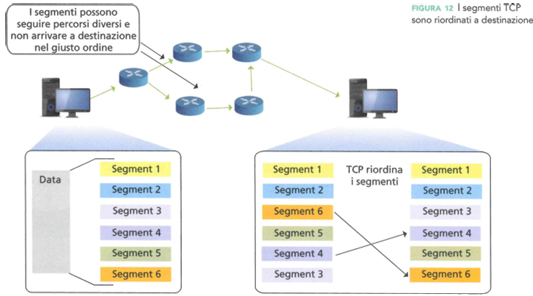
**TCP – Affidabilità, ack e ritrasmissioni**

Il **TCP (Transmission Control Protocol)** è un protocollo di trasporto molto utilizzato, più dell’UDP, perché:

* È **connection-oriented**: stabilisce una connessione tra mittente e destinatario.
* È **affidabile**: garantisce la consegna ordinata dei dati come un flusso continuo di byte (data stream).

**Funzionamento:**

* Utilizza **numeri di sequenza** per ricostruire il messaggio originale, ordinando correttamente i segmenti ricevuti.
* Dà l’impressione all’applicazione di avere una **linea dedicata** tra mittente e destinatario.



**Caratteristiche della connessione TCP:**

* **Full-duplex**: trasmissione e ricezione simultanea.
* **Point-to-point**: connessione tra un solo mittente e un solo destinatario.
* Richiede **inizializzazione di variabili di stato** da entrambe le parti prima del trasferimento dati.

**Risorse del computer:**

Rispetto all’UDP, TCP:

* Usa più **CPU** per l’elaborazione.
* Richiede più **memoria** per gestire lo stato della connessione.
* Ha bisogno di più **banda** per via della ritrasmissione e dell’**header** più grande (20 byte contro gli 8 di UDP).

### TCP e le applicazioni affidabili:

Il **TCP** è utilizzato da applicazioni che richiedono una trasmissione **affidabile** dei dati, come:

* **FTP**: trasferimento di file
* **SMTP**: invio di posta elettronica
* **HTTP**: trasferimento di pagine web

Questi protocolli si basano su TCP per garantire che i dati arrivino correttamente e in ordine.

### Standard TCP principali:

Nel tempo sono stati definiti diversi **RFC (Request for Comments)** per migliorare e regolare TCP:

* **RFC 793**: prima specifica stabile del protocollo TCP
* **RFC 7323**: introduce estensioni per migliorare le **prestazioni**
* **RFC 5681**: descrive gli algoritmi per il **controllo della congestione** nella rete

### Comunicazione TCP e processo applicativo:

#### Accesso sincrono alle porte:

* I sistemi operativi gestiscono le porte in modo **sincrono**, quindi il processo si **blocca temporaneamente** durante l’accesso.

#### Possibili scarti dei dati ricevuti:

I dati possono essere scartati se:

* Il processo destinatario **non è pronto** o la **porta non esiste**.
* Il **buffer di ricezione** è pieno e non può contenere altri dati.

### Invio e ricezione dei dati:

* Il processo **scrive i dati** in un buffer, aggiunge l’**header** e passa il controllo a TCP.
* Per ricevere, il processo **definisce un buffer** e affida la gestione a TCP.

### 

### Informazioni di controllo passate a TCP:

Il processo comunica a TCP:

* **Source address**: indirizzo completo del mittente.
* **Destination address**: indirizzo completo del destinatario.
* **Next packet sequence number**: numero di sequenza del prossimo pacchetto.
* **Current buffer size**: dimensione del buffer.
* **Next write position**: dove scrivere i nuovi dati.
* **Next read position**: da dove leggere i dati da inviare.
* **Timeout/flag**: tempo di ritrasmissione e sincronizzazione (es. semafori).

### Bufferizzazione dei dati:

* TCP può **salvare i dati in un buffer** per inviarli più efficientemente quando si accumula una certa quantità.

#### Eccezioni alla bufferizzazione:

1. **Flag PUSH = 1**: invio immediato dei dati, senza bufferizzazione.
2. **Flag URG = 1**: trasmissione urgente; l’applicazione interrompe l’attività per gestire subito i dati ricevuti.