Il Livello di Trasporto nel Modello OSI

1 Introduzione al Livello di Trasporto

Il **livello di trasporto** è il quarto livello del modello OSI e una parte essenziale della suite di protocolli TCP/IP. Fornisce servizi fondamentali per la comunicazione *end-to-end* tra processi su dispositivi remoti. La sua funzione principale è garantire che i dati siano trasferiti in modo affidabile, efficiente e strutturato tra applicazioni in esecuzione su dispositivi diversi.

Il livello di trasporto agisce come un intermediario tra il livello di rete e il livello applicativo, creando una connessione logica tra i processi, gestendo la trasmissione dei dati e assicurandosi che l'integrità dei dati venga mantenuta durante il trasferimento.

2 Funzioni Principali del Livello di Trasporto

2.1 Comunicazione End-to-End

Il livello di trasporto stabilisce una connessione logica tra processi applicativi su dispositivi remoti. Utilizza indirizzi *IP* (per identificare gli host) e numeri di *porta* (per identificare i processi) per gestire la comunicazione diretta tra due applicazioni.

2.2 Multiplexing e Demultiplexing

- **Multiplexing**: Consente a più applicazioni di condividere la stessa connessione di rete, distinguendole tramite numeri di porta.
- **Demultiplexing**: Smista i dati ricevuti dal livello di trasporto e li indirizza verso l'applicazione appropriata sulla base del numero di porta.

2.3 Incapsulamento e Decapsulamento

- **Incapsulamento**: I dati ricevuti dal livello applicativo vengono trasformati in segmenti (per il protocollo TCP) o datagrammi (per il protocollo UDP) e inviati al livello di rete.
- **Decapsulamento**: I segmenti o datagrammi ricevuti dal livello di rete vengono elaborati e consegnati al livello applicativo.

2.4 Controllo di Flusso

Il controllo di flusso previene il sovraccarico del ricevitore regolando la velocità di trasmissione dei dati. Una delle tecniche più comuni per implementare il controllo di flusso è la *Sliding Window*, che consente di gestire la quantità di dati che possono essere inviati prima di ricevere un riconoscimento dal ricevitore.

2.5 Controllo degli Errori

Il livello di trasporto è responsabile per l'identificazione e la correzione degli errori nei dati trasmessi. Utilizza tecniche di controllo degli errori come il *checksum*, che verifica l'integrità dei dati, e garantisce che i dati vengano ricevuti nell'ordine corretto, senza duplicazioni.

2.6 Controllo della Congestione

Il controllo della congestione regola la velocità di trasmissione dei dati per evitare il sovraccarico della rete, migliorando così le prestazioni complessive della comunicazione. Quando la rete è congestionata, il livello di trasporto può ridurre la velocità di invio dei dati per prevenire la perdita di pacchetti.

3 Protocolli Principali del Livello di Trasporto

3.1 User Datagram Protocol (UDP)

3.1.1 Caratteristiche

- **UDP** è un protocollo *leggero* e *non orientato alla connessione*.
- Non garantisce affidabilità, ordine o correzione degli errori, il che lo rende adatto a applicazioni che richiedono velocità e bassa latenza, come *streaming*, *VoIP* e *DNS*.

3.1.2 Struttura del Datagramma UDP

Il datagramma UDP è composto da una serie di campi di dati. La struttura di base di un datagramma UDP è la seguente:

Numero Porta Mittente (16 bit)	Numero Porta Destinatario (16 bit)	Lunghezza (16 bit)	Checksum (16

3.1.3 Checksum UDP

Il *checksum* UDP serve a verificare l'integrità del datagramma, comprese la dimensione totale del datagramma e la sua intestazione. Il *checksum* è calcolato su un *pseudo-header* che include:

- Gli indirizzi IP del mittente e del destinatario.
- Il protocollo di livello superiore (UDP).
- La dimensione totale del datagramma.

Se il *checksum* non è valido, il datagramma UDP verrà scartato dal ricevitore.

3.2 Transmission Control Protocol (TCP)

3.2.1 Caratteristiche

- **Protocollo orientato alla connessione**: TCP stabilisce una connessione logica tra il mittente e il destinatario prima di iniziare a trasmettere i dati.
- **Affidabilità**: TCP garantisce che i dati arrivino correttamente e nell'ordine giusto, senza duplicazioni.
- **Comunicazione full-duplex**: Permette al mittente e al destinatario di inviare e ricevere dati simultaneamente.

3.2.2 Struttura del Segmento TCP

Il segmento TCP è costituito da vari campi, ciascuno con un ruolo specifico nella gestione della trasmissione dei dati. La struttura di un segmento TCP è la seguente:

Campo	Lunghezza (bit)	Descrizione
Numero Porta Sorgente	16	Identifica l'applicazione mittente.
Numero Porta Destinatario	16	Identifica l'applicazione destinataria.
Numero di Sequenza	32	Indica il numero del primo byte trasmesso.
Numero di Riscontro (ACK)	32	Indica il numero del prossimo byte atteso.
Lunghezza Intestazione	4	Dimensione dell'intestazione in multipli di 4 byte.
Flag di Controllo	6	Specifica lo stato della connessione (SYN, ACK).
Dimensione Finestra	16	Quantità di dati che il mittente può inviare.
Checksum	16	Verifica l'integrità del segmento.
Puntatore Urgente	16	Indica dati urgenti, se presenti.

3.2.3 Funzionalità di TCP

- **Numerazione dei byte**: Ogni byte inviato da TCP viene numerato in modo che il destinatario possa riorganizzare i dati nel giusto ordine.
- **Ritrasmissione per pacchetti persi**: Se un pacchetto viene perso durante la trasmissione, TCP ritrasmette il pacchetto fino a quando non riceve un riscontro.
- **Timer per gestire ritardi e congestioni**: TCP utilizza vari timer per gestire i ritardi e per reagire alle congestioni di rete.

4 Meccanismi del Livello di Trasporto

4.1 Apertura e Chiusura della Connessione (TCP)

4.1.1 Apertura (Three-Way Handshake)

Il processo di apertura della connessione in TCP avviene attraverso un *three-way handshake*:

- **SYN**: Il client invia un segmento con il flag SYN per iniziare la connessione.
- **SYN-ACK**: Il server risponde con un segmento SYN-ACK per confermare la richiesta di connessione.
- **ACK**: Il client invia una conferma (ACK) per completare l'instaurazione della connessione.

4.1.2 Chiusura (Four-Way Handshake)

La chiusura della connessione TCP avviene in quattro fasi:

- **FIN**: Il mittente invia un segmento con il flag FIN per indicare la chiusura della connessione.
- **ACK**: Il ricevitore risponde con un segmento di conferma (ACK).
- **FIN**: Il ricevitore invia un segmento FIN per chiudere anche la propria connessione.
- **ACK**: Il mittente invia una conferma finale (ACK) per completare la chiusura.

4.2 Controllo di Flusso

4.2.1 Sliding Window

Il controllo di flusso in TCP è implementato attraverso la tecnica della *sliding window*, che determina quanti dati il mittente può inviare senza attendere un riscontro (ACK). La finestra si adatta dinamicamente in base alle capacità del ricevitore, migliorando l'efficienza e riducendo il rischio di sovraccarico.

4.3 Controllo della Congestione

TCP utilizza diversi algoritmi per gestire la congestione nella rete:

- **Slow Start**: Durante l'inizializzazione della connessione, la finestra di congestione cresce esponenzialmente fino a raggiungere una soglia predefinita.

- **Congestion Avoidance**: Dopo che la finestra di congestione supera una certa soglia, l'incremento della finestra diventa lineare per evitare la congestione.
- **Fast Retransmit**: TCP ritrasmette i segmenti persi quando riceve tre *ACK* duplicati.
- **Fast Recovery**: Durante una congestione moderata, TCP non riduce completamente la finestra di congestione ma la recupera velocemente per evitare di ripartire da zero.

4.4 Timer in TCP

TCP utilizza vari timer per gestire la connessione:

- **Timer di Ritrasmissione (RTO)**: Il timer di ritrasmissione viene calcolato in base al tempo di round-trip (RTT) e determina quanto tempo TCP aspetta prima di ritrasmettere un pacchetto non confermato.
- **Timer di Persistenza**: Mantiene attiva la connessione TCP quando la finestra di ricezione è nulla, evitando il blocco della connessione.
- **Timer Keepalive**: Previene la chiusura inattesa della connessione TCP, inviando pacchetti di controllo per verificare che la connessione sia ancora attiva.
- **Timer 2MSL**: Assicura che tutti i segmenti siano ricevuti prima della chiusura definitiva della connessione, rispettando un tempo doppio rispetto al massimo round-trip time (2MSL, Maximum Segment Lifetime).

5 Confronto tra TCP e UDP

Caratteristica	**TCP**	**UDP**
Affidabilità	Sì (ritrasmissioni, ACK)	No
Ordinamento	Sì	No
Connessione	Orientato alla connessione	Non orientato alla connessione
Overhead	Alto	Basso
Applicazioni Tipiche	File transfer, email	Streaming, VoIP, DNS

6 TCP Tahoe e TCP Reno

6.1 TCP Tahoe

- **Riparte da Slow Start**: In caso di congestione, TCP Tahoe riparte dalla fase di *Slow Start* dopo aver ricevuto 3 *ACK* duplicati.

- **Riduzione drastica della finestra di congestione**: TCP Tahoe riduce in modo significativo la finestra di congestione quando rileva la perdita di pacchetti.

6.2 TCP Reno

- **Introduzione di Fast Recovery**: TCP Reno migliora l'efficienza rispetto a TCP Tahoe introducendo la tecnica di *Fast Recovery*, evitando di tornare completamente alla fase di *Slow Start* dopo una congestione.
- **Maggiore efficienza durante congestioni moderate**: TCP Reno è più efficiente di Tahoe durante congestioni moderate, grazie alla gestione più aggressiva della finestra di congestione.

7 Considerazioni Finali

Il livello di trasporto è cruciale per garantire comunicazioni affidabili e ottimizzate nelle reti moderne. TCP, con il suo rigoroso controllo di flusso, correzione degli errori e gestione della congestione, è ideale per applicazioni critiche che richiedono elevata affidabilità, come il trasferimento di file e la posta elettronica. D'altro canto, UDP offre prestazioni migliori in termini di velocità e semplicità, ed è quindi preferito per applicazioni che non richiedono affidabilità assoluta, come lo streaming video, VoIP e DNS. La comprensione approfondita del livello di trasporto è essenziale per progettare sistemi di comunicazione robusti, efficienti e in grado di adattarsi alle diverse esigenze delle applicazioni moderne.