**Capitolo Completo: Il Livello di Trasporto**

**1. Introduzione al Livello di Trasporto**

Il livello di trasporto è il quarto livello del modello OSI ed è parte integrante della suite di protocolli TCP/IP. Fornisce servizi essenziali per la comunicazione tra processi su host diversi. La sua funzione principale è garantire un trasferimento di dati efficiente, affidabile e strutturato tra applicazioni eseguite su dispositivi remoti.

**2. Funzioni Principali**

1. **Comunicazione End-to-End**:
   * Stabilisce una connessione logica tra processi applicativi su due dispositivi remoti.
   * Utilizza indirizzi IP e numeri di porta per identificare gli host e i processi.
2. **Multiplexing e Demultiplexing**:
   * **Multiplexing**:
     + Permette a più applicazioni di utilizzare simultaneamente una connessione di rete.
     + Utilizza numeri di porta per distinguere i processi.
   * **Demultiplexing**:
     + Smista i dati ricevuti verso l'applicazione appropriata.
3. **Incapsulamento e Decapsulamento**:
   * I dati provenienti dal livello applicativo sono incapsulati in segmenti o datagrammi, a seconda del protocollo utilizzato (TCP o UDP).
   * Una volta ricevuti, i dati vengono decapsulati e consegnati al livello applicativo.
4. **Controllo di Flusso**:
   * Previene il sovraccarico del ricevitore regolando la velocità di trasmissione.
   * Utilizza tecniche come la finestra scorrevole (Sliding Window).
5. **Controllo degli Errori**:
   * Identifica e corregge errori nei dati trasmessi.
   * Garantisce che i dati vengano ricevuti nell'ordine corretto, senza duplicati.
6. **Controllo della Congestione**:
   * Regola la trasmissione per evitare il sovraccarico della rete.

**3. Protocolli Principali del Livello di Trasporto**

**3.1 User Datagram Protocol (UDP)**

* **Caratteristiche**:
  + Protocollo non orientato alla connessione.
  + Nessuna garanzia di affidabilità o ordine nella consegna.
  + Utilizzato per applicazioni che richiedono velocità, come DNS, VoIP e streaming.
* **Struttura del Datagramma UDP**:

| **Campo** | **Lunghezza (bit)** | **Descrizione** |
| --- | --- | --- |
| Numero Porta Mittente | 16 | Identifica il processo mittente. |
| Numero Porta Destinatario | 16 | Identifica il processo destinatario. |
| Lunghezza | 16 | Dimensione totale del datagramma (header + dati). |
| Checksum | 16 | Verifica l'integrità dei dati e dell'intestazione. |

* **Checksum UDP**:
  + Comprende:
    - Pseudo-header (indirizzi IP e protocollo).
    - Header UDP.
    - Dati del datagramma.
  + Se il checksum non è valido, il datagramma viene scartato.

**3.2 Transmission Control Protocol (TCP)**

* **Caratteristiche**:
  + Protocollo orientato alla connessione.
  + Garantisce affidabilità, ordine e assenza di duplicati.
  + Supporta la comunicazione full-duplex.
* **Struttura del Segmento TCP**:

| **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 nel segmento. |
| Numero di Riscontro (ACK) | 32 | Indica il numero del prossimo byte atteso dal mittente. |
| Lunghezza Intestazione | 4 | Dimensione dell'intestazione in multipli di 4 byte. |
| Flag di Controllo | 6 | Specifica lo stato della connessione (SYN, ACK, FIN, ecc.). |
| Dimensione Finestra | 16 | Quantità massima di dati che il mittente può inviare. |
| Checksum | 16 | Verifica l'integrità del segmento. |
| Puntatore Urgente | 16 | Indica dati urgenti, se presenti. |

* **Funzionalità di TCP**:
  + Numerazione dei byte per garantire la consegna ordinata.
  + Meccanismi di ritrasmissione per pacchetti persi.
  + Utilizzo di timer per gestire la ritrasmissione e il controllo di congestione.

**4. Meccanismi del Livello di Trasporto**

**4.1 Apertura e Chiusura della Connessione (TCP)**

1. **Apertura (Three-Way Handshake)**:
   * Il client invia un segmento con flag SYN al server.
   * Il server risponde con un segmento SYN-ACK.
   * Il client conferma con un segmento ACK.
2. **Chiusura (Four-Way Handshake)**:
   * Il mittente invia un segmento con flag FIN.
   * Il ricevitore risponde con un segmento ACK.
   * Il ricevitore invia un segmento FIN.
   * Il mittente risponde con un segmento ACK finale.

**4.2 Controllo di Flusso**

* **Sliding Window**:
  + Gestisce la quantità di dati inviati senza attendere un riscontro (ACK).
  + Utilizza una finestra di invio e una finestra di ricezione.
* **Ritrasmissione**:
  + Ritrasmette i segmenti non riscontrati entro un determinato timeout.

**4.3 Controllo della Congestione**

* **Slow Start**:
  + Incrementa esponenzialmente la finestra di congestione all'inizio.
* **Congestion Avoidance**:
  + Incremento lineare della finestra per evitare congestione.
* **Fast Retransmit e Fast Recovery**:
  + Ritrasmette segmenti persi dopo 3 ACK duplicati.
  + Riduce il valore di ssthresh e ripristina la trasmissione.

**4.4 Timer in TCP**

1. **Timer di Ritrasmissione (RTO)**:
   * Calcolato in base all'RTT e alla sua deviazione.
   * Incrementa progressivamente in caso di timeout.
2. **Timer di Persistenza**:
   * Evita il blocco della connessione in caso di finestra di ricezione nulla.
3. **Timer Keepalive**:
   * Mantiene attiva la connessione.
4. **Timer 2MSL**:
   * Assicura che tutti i segmenti siano stati ricevuti prima di chiudere definitivamente la connessione.

**5. Confronto tra TCP e UDP**

| **Caratteristica** | **TCP** | **UDP** |
| --- | --- | --- |
| Affidabilità | Sì (conferme, ritrasmissioni) | No |
| Ordinamento | Sì | No |
| Connessione | Orientato alla connessione | Non orientato alla connessione |
| Overhead | Alto | Basso |
| Utilizzo | File transfer, e-mail, web | Streaming, VoIP, DNS |

La differenza principale tra **TCP Tahoe** e **TCP Reno** riguarda la gestione dei **3 ACK duplicati** e il comportamento successivo in caso di congestione. Di seguito una spiegazione dettagliata:

**TCP Tahoe**

1. **Gestione dei 3 ACK Duplicati**:
   * Quando il mittente riceve 3 ACK duplicati, Tahoe interpreta questo come **perdita di pacchetti**.
   * In risposta:
     + **Riduce la soglia di congestione** (ssthreshssthresh) a metà della dimensione della finestra di congestione attuale (cwndcwnd).
     + **Ripristina il valore di cwndcwnd a 1 MSS (Maximum Segment Size)**.
     + Riavvia il processo di **Slow Start**.
2. **Effetto**:
   * Tahoe riparte con un incremento **esponenziale** della finestra di congestione, riducendo drasticamente il throughput fino al raggiungimento di ssthreshssthresh.
3. **Strategia di Congestione**:
   * **Slow Start**: Aumenta cwndcwnd esponenzialmente finché cwnd<ssthreshcwnd < ssthresh.
   * **Congestion Avoidance**: Dopo aver raggiunto ssthreshssthresh, incrementa cwndcwnd linearmente.

**TCP Reno**

1. **Gestione dei 3 ACK Duplicati (Fast Recovery)**:
   * Quando il mittente riceve 3 ACK duplicati, Reno interpreta questo come **perdita moderata di pacchetti** (non un completo timeout).
   * In risposta:
     + Riduce la soglia di congestione (ssthreshssthresh) a metà della dimensione di cwndcwnd.
     + **Non ripristina cwndcwnd a 1 MSS**, ma lo riduce a ssthresh+3MSSssthresh + 3 MSS, permettendo di mantenere il flusso di dati.
     + Entra nello stato di **Fast Recovery**, trasmettendo nuovi pacchetti senza tornare a Slow Start.
   * Solo dopo aver ricevuto un ACK cumulativo che copre il pacchetto perso, Reno esce da Fast Recovery e passa a **Congestion Avoidance**.
2. **Effetto**:
   * Reno evita di ripartire da 1 MSS, mantenendo una maggiore efficienza durante la congestione rispetto a Tahoe.
3. **Strategia di Congestione**:
   * **Fast Recovery**: Recupera rapidamente il flusso di trasmissione dopo la congestione.
   * Dopo Fast Recovery, Reno entra in **Congestion Avoidance** per incrementare cwndcwnd linearmente.

**Confronto tra TCP Tahoe e TCP Reno**

| **Caratteristica** | **TCP Tahoe** | **TCP Reno** |
| --- | --- | --- |
| **Gestione dei 3 ACK duplicati** | Riparte da Slow Start (cwnd = 1 MSS). | Entra in Fast Recovery. |
| **Riduzione di cwndcwnd** | cwndcwnd ridotto a 1 MSS. | cwndcwnd ridotto a ssthresh+3MSSssthresh + 3 MSS. |
| **Efficienza** | Minore (diminuzione drastica del throughput). | Maggiore (evita Slow Start completa). |
| **Congestione moderata** | Non distinta da congestione grave. | Gestita con Fast Recovery. |
| **Congestione grave (timeout)** | Comportamento identico: riparte da Slow Start. | Comportamento identico: riparte da Slow Start. |

**Conclusione**

* **TCP Tahoe** è più semplice, ma meno efficiente in presenza di congestione moderata, poiché riparte sempre da Slow Start.
* **TCP Reno** migliora le prestazioni introducendo **Fast Recovery**, che evita di ripartire da zero in caso di perdita moderata di pacchetti. Tuttavia, in caso di **timeout** (congestione grave), entrambi i protocolli si comportano allo stesso modo, ripristinando cwndcwnd a 1 MSS.

**6. Considerazioni Finali**

Il livello di trasporto è fondamentale per garantire la comunicazione end-to-end tra dispositivi remoti. TCP offre affidabilità, ordine e controllo di flusso, mentre UDP privilegia la velocità e la semplicità. La scelta del protocollo dipende dalle esigenze specifiche dell'applicazione, rendendo il livello di trasporto un elemento cruciale nelle reti moderne.