IL TRASNPORT LAYER DEL TCP/IP

LE PORTE

Su un host ci possono essere diverse applicazioni aperte contemporaneamente che generano 1 o più processi di richiesta, per identificare quale processo è il corretto destinatario vengono utilizzati dei **punti di accesso** chiamati **porte**

Ogni porta → 16 bit (4 cifre decimali)

- 0 - 1.023 → **porte note**
- 1.024 - 49.151 → **porte registrate**
- 49.152 - 65.535 → **porte dinamiche e/o private**

L'host client e l'host server oltre al reciproco indirizzo IP devono conoscere i rispettivi numeri di porta

- **Source port** → numero della porta sull'host client (in attesa di risposta dal server)
- **Destination port** → numero della porta su cui il processo server è in ascolto

Ogni pacchetto che viaggia in rete deve contenere 4 informazioni:

Client IP, Server IP, Client Port, Server Port

Andando ad aggiungere anche il protocollo si forma una **quintupla** detta **Association**:

TCP, 200.18.6.12, 3000, 58.163.138.10, 4000

Ciascuna di queste parti è una **socket** su un host diverso

Il livello Transport si occupa della **comunicazione end-to-end**

TCP	UDP
Tipo di servizio	
Connection oriented	Connectionless
Servizio orientato alla connessione	Servizio non orientato alla connessione
Controllo dei dati	
Assciura la correttezza dei dati	Non asscira la correttezza dei dati
Riordina i pacchetti di dati arrivati	Possibili dati in disordine
Utilizzo	
HTTP, FTP, SMTP	Chiamate VOIP, streaming

- Protocollo TCP → PDU chiamata **segment**
- Protocollo UDP → PDU chiamata **datagram**

Multiplexing (trasmissione)

Il livello Transport riceve i dati dalle socket e aggiunge le proprie informazioni di controllo (header)

Demultiplexing (ricezione)

Il livelo Transport legge l'header e determina a quale socket consegnare i dati

PROTOCOLLO UDP

Protocollo che non prevede l'utilizzo di una connessione tra l'host mittente e l'host destinatario → ciascun datagram (PDU) è trattato in modo indipendente

source port number (16 bit)	destination port number (16 bit)
lenght (16 bit)	chechsum (opzionale) (16 bit)
data	

- **Source port number** → porta sull'host mittente
- **Destination port number** → porta sull'host del destinatario
- **Lenght** → lunghezza totale in byte del datagram PDU (header + dati)
- **Chechsum** → codice di controllo del datagram PDU, se danneggiato viene scartato
- **Data** → contiene le informazioni trasmesse/ricevute

Multiplexing

1. datagram UDP formato da Header + messaggio ricevuto dal liv. Application
2. source port preso dalla socket attraverso cui è stato inviato il messaggio
3. destination port è il parametro della primitiva data_request (liv. Application)
4. lenght → calcolato
5. chechsum → calcolato
6. il datagram viene inviato al livello Network

Demultiplexing

1. datagram UDP → ricevuto dal livello Network
2. si verifica la sua integrità e lo ci confronta con quello nel campo checksum
3. si cerca la socket UDP associata al numero di porta nel campo destination port
4. la parte "data" del datagram UDP viene inviata alla socket individuata

Vantaggi di questo protocollo:

- non richiede di stabilire una connessione
- non mantiene lo stato della connessione (server supporta molti più client)
- sovraccarico dovuto all'header del pacchetto è minimo (8 byte)
- controllo del livello Application molto più efficace, il mittente non viene bloccato

UDP LITE

Il campo **checksum coverage lenght** → specifica quanti byte del datagram UDP saranno controllati:

- solo 8 byte → controllo solo dell'header
- tutti → controllo totale

PROTOCOLLO TCP

Protocollo di trasporto che offre un servizio affidabile (**connection oriented**), garantendo la consegna dei dati in modo ordinato

La connessione ha le seguenti caratteristiche:

- **Full duplex** → si può trasmettere/ricevere in contemporanea
- **Point-to-point** → un solo mittente e un solo destinatario
- **Inizializzazione delle variabili di stato** → accordi prima di trasferire i dati

Comunicazione tra TCP e processo applicativo

- **Source address** → indirizzo completo del mittente (network + host + port)
- **Destination address** → indirizzo complete del destinatario (network + host + port)
- **Next packet sequence number** → numero di sequenza del pacchetto
- **Current buffer size** → dimensione del buffer del mittente
- **Next write position** → inizio dell'area del buffer per i nuovi dati da trasmettere
- **Next read position** → indirizzo dell'area del buffer dove leggere i dati da inviare
- **Timeout / flag** → tempo dopo il quale i dati non riscontarti vengono ritrasmessi

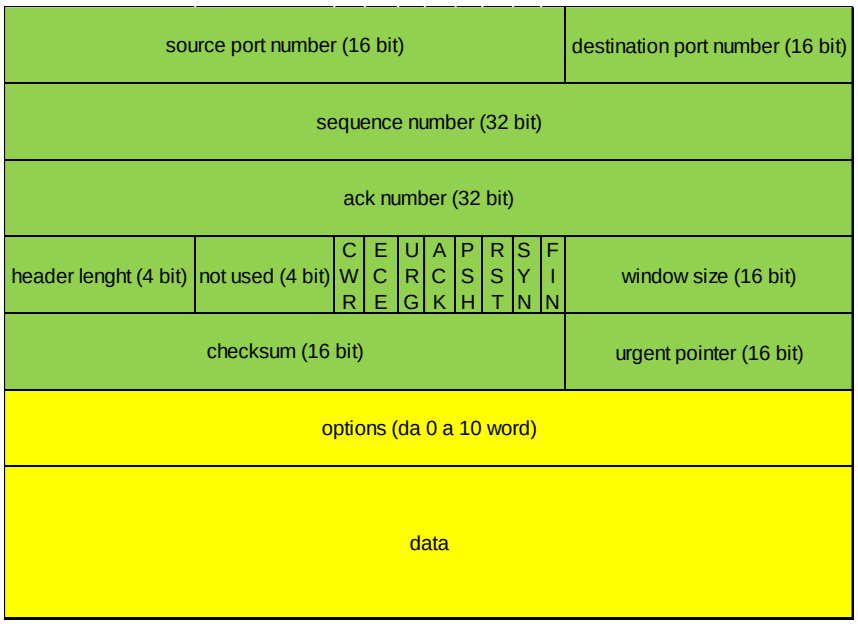
Quando un'applicazione passa dei dati a TCP si può:

- **Salvarli in un buffer** → migliora l'efficienza (dati spediti ad una certa quantità)
- **Spedirlo immediatamente**

Esistono 2 eccezioni alla bufferizzazione:

- Flag PUSH = 1 → dati spediti subito
- Flag URG = 1 → dati non accumulati e subito trasmessi o esaminati

1 word = 32 bit



- **Source port number** → porta sull'host mittente
- **Destination port number** → porta sull'host del destinatario
- **Sequence number** → numero di sequenza progressivo

- **Ack number** → numero di riscontro
- **Header lenght** → indica la lunghezza dell'header nel segmento TCP
- **Not used** → non utilizzato
- **8 bit dedicati a flag** →
  - **ACK** → se = 1 → segmento TCP usato in risposta ad uno ricevuto
  - **RST** → se = 1 → connessione NON valida
  - **SYN** → se = 1 → l'host mittente vuole aprire una connessione
  - **FIN** → se = 1 → l'host mittente non ha più dati da inviare
- **Windows size** → usato per dire al mittente quanti dati può ricevere
- **Checksum** → verifica della validità del segmento
- **Urgent pointer** → ha significato se URG = 1 → contiene il numero che deve essere sommato (offset) al sequence number per ottenere il numero dell'ultimo byte urgente nel campo data
- **Options** → campo facoltativo, si può specificare la dimensione massima della PDU
- **Data** → contiene i dati da trasmettere/ricevuti

GESTIONE DELLA CONGESTIONE

La **finestra di congestione** è il massimo numero di byte che la rete è in grado di trasmettere senza che si verifichino dei timeout (perdita di dati)

La **finestra di ricezione** indica quanti byte il destinatario è in grado di ricevere

maxWindow = **min**(Finestra di congestione, Finestra di ricezione)

SLOW START E CONGESTION AVOIDANCE

1. Il mittente imposta la Finestra di congestione alla massima quantità di bte che la rete può spedire
2. Ogni volta che viene inviato un segmento TCP e viene ricevuto il relativo ACK il valore della finestra viene raddoppiato (**crescita esponenziale**)
3. Raggiunto il **threshold** (di solito impostato a 64KB) la finestra viene regolata dal **congestion avoidance** (**incremento lineare**)
4. Quando si genera un timeout:
   a. La soglia viene impostata alla metà della finestra di congestione
   b. La finestra di congestione viene riportata al suo livello iniziale

Vedi schema pag. 284

HANDSHAKING

1. **Instaurazione della connessione TCP** (Three-Way Handshake)
   Colloquio iniziale tra client e server
2. **Trasmissione dei dati**
   Si attuano meccanismi di controllo degli errori, controllo di flusso e di congestione
3. **Abbattimento della connessione TCP** (Double Two-Way Handshake)
   Three-Way se la chiusura è contemporanea, altrimenti Double Two-Way

Vedi schema pag. 286-289