Reti di Calcolatori

IL LIVELLO TRASPORTO
Protocolli TCP e UDP

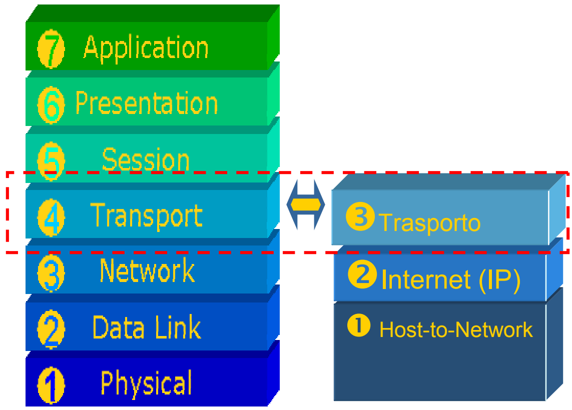
Il Livello Trasporto

- I servizi del livello Trasporto
- Le primitive di Trasporto
- Indirizzamento
- Protocolli di Trasporto
- Livello Trasporto in Internet
  - **UDP**
  - **TCP**

Livello TRASPORTO

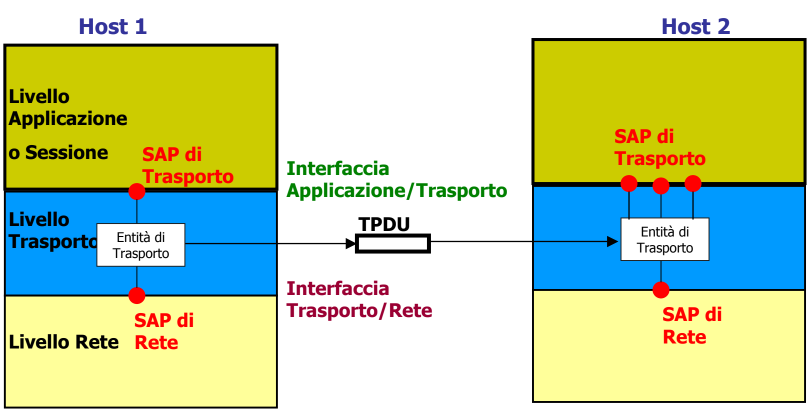
Funzionalità del livello **Trasporto**:

1. Controllo di flusso
2. Controllo delle connessioni
3. Controllo di errori
4. Sequenzializzazione
5. Multiplexing sulle applicazioni
6. Controllo della congestione.



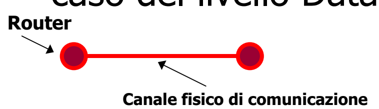
Servizi di Trasporto

- Servizio efficiente e affidabile per le applicazioni di rete.
- Il software o l'hardware che fornisce i servizi di trasporto è detta **UNITA' DI TRASPORTO**.

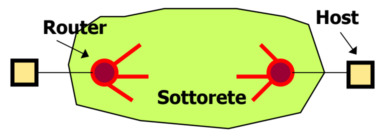


Protocolli di Trasporto

- Gestiscono l'**indirizzamento**, il **controllo di flusso**, il **multiplexing**, i **numeri di sequenza** e il **controllo degli errori** per un collegamento attraverso una rete.
- La situazione da gestire è più complessa del caso del livello Data Link.



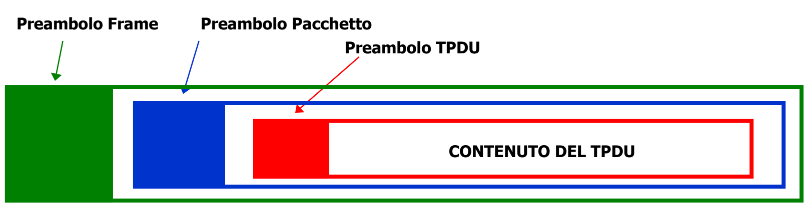
Livello Data Link



Livello Trasporto

Primitive di Trasporto

- **TPDU (Transport Protocol Data Unit)** è l'unità dei dati scambiati dal protocollo di trasporto.



Primitive di Trasporto

Esempi di primitive di un servizio di Trasporto

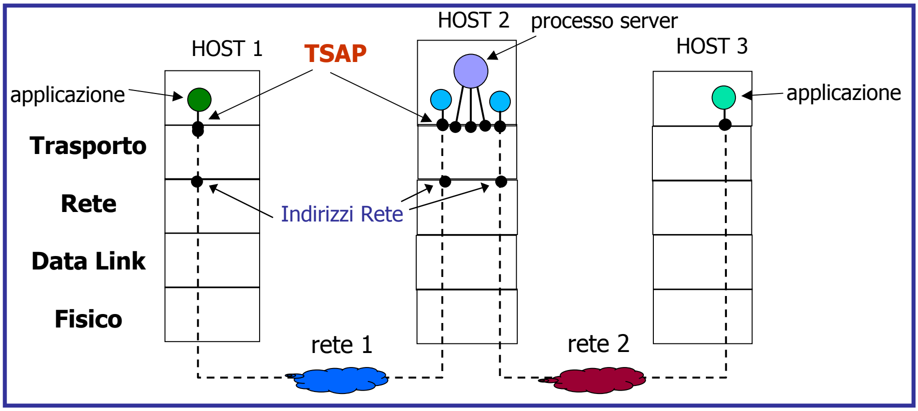
Primitive	**Significato**
CREATE_CE	Crea un elemento di connessione
CONNECT	Richiede una connessione
SEND	Spedisce dati su una connessione
RECEIVE	Riceve dati su una connessione
DISCONNECT	chiude una connessione

Protocolli di Trasporto

- Occorre definire la modalità di indirizzamento a livello Trasporto.
- Su uno stesso host possono essere disponibili più connessioni quindi il livello di Trasporto su un host gestisce numerose connessioni.
- Si deve risolvere il problema della capacità di memorizzazione della rete. Un pacchetto può essere memorizzato in un router e consegnato dopo un certo ritardo.

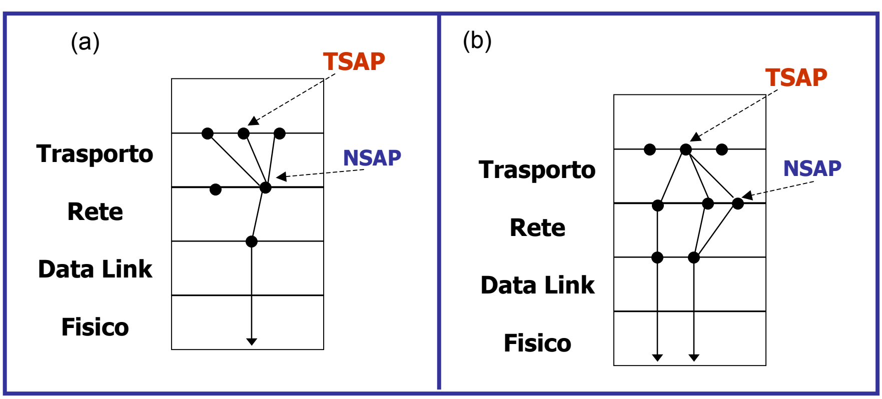
Indirizzamento e Connessioni

- Un indirizzo di trasporto identifica l'host e la specifica connessione sull'host ***Transport Service Access Point*** (**TSAP**).



Multiplexing

- Il livello Trasporto gestisce anche connessioni multiple.
  - Upward multiplexing (a)
  - downward multiplexing (b)

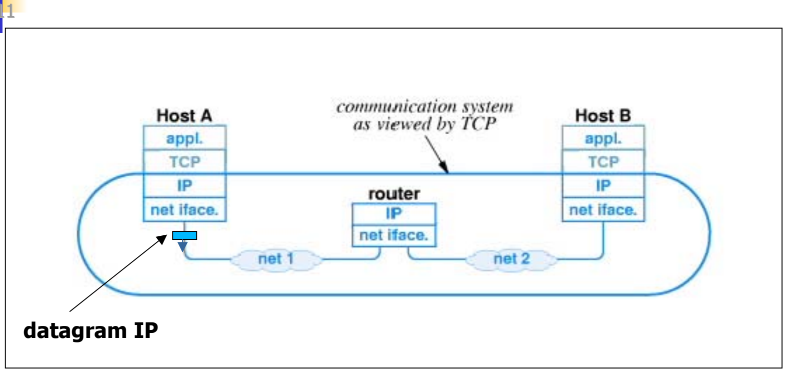


TCP e UDP

Protocolli di trasporto definiti su rete Internet (su IP)

- **Trasmission Control Protocol** (**TCP**) definisce un protocollo di trasporto orientato alla connessione
  - progettato per fornire un flusso affidabile end-to-end su una internet inaffidabile.
- **User Data Protocol** (**UDP**) definisce un protocollo senza connessione
  - permette di inviare datagram IP senza stabilire una connessione
  - si usa per comunicazioni che prevedono una richiesta e una risposta.

TCP/IP



- Il **TCP** su un computer usa **IP** per comunicare con il **TCP** di un altro computer.

Funzionalità del TCP

Trasmissione

- Riceve un flusso di dati dall'applicazione,
- Li organizza in unità lunghe al massimo 64Kb,
- Spedisce le unità di dati come datagram IP.

Ricezione

- Riceve i datagram IP,
- Ricostruisce il flusso di byte originale nella sequenza corretta.

E' necessaria la ritrasmissione dei datagram non ricevuti e il riordinamento dei datagram arrivati in ordine errato.

Socket

- Il concetto di **socket** è stato introdotto su UNIX BSD.
- Ogni socket è caratterizzato da un indirizzo consistente nell'**indirizzo IP** dell'host e di un numero locale a 16 bit (**porta**)
- Per ottenere un servizio TCP si deve creare esplicitamente una connessione fra un socket della macchina mittente e un socket della macchina ricevente.
- Una volta attivato un socket è utilizzato come un file.
- Le connessioni sono identificate con gli identificatori dei socket dei due lati ***(socket1,socket2)***.

Primitive di Trasporto : i Socket

Primitive	Significato
SOCKET	Crea un elemento di connessione (socket)
BIND	Assegna un indirizzo al socket
LISTEN	Accetta connessioni
ACCEPT	Attende una connesione
CONNECT	Richiede una connessione
SEND	Spedisce dati su una connessione
RECEIVE	Riceve dati su una connessione
CLOSE	chiude una connessione

Le Porte: I TSAP del TCP

- Le porte attive definiscono i servizi TCP disponibili.
- Per connettersi ad un servizio specifico su un server si deve conoscere il numero di porta su cui il processo server accetta le connessioni.
  - Le porte inferiori alla 256 sono dette **porte ben note** (well-known ports) e corrispondono a servizi standard.
  - In Unix la lista dei servizi e delle porte è nel file /etc/services.
  - Ad esempio
    - la **porta 21** di TCP corrisponde al servizio **FTP** (File Transfer Protocol).
    - la **porta 80** di TCP corrisponde al servizio **HTTP** (Hypertext Transfer Protocol) ovvero al server Web.
  - Un servizio "standard" può anche essere attivato su una porta diversa (es. HTTP su 8080).

Le Porte del Client

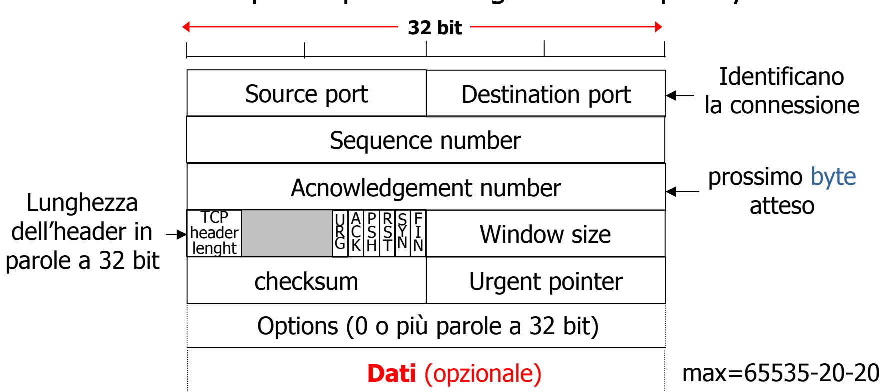
- Il client definisce la porta di ogni sua connessione utilizzando numeri in genere elevati e scelti in modo da essere unici sull'host.
- Ad esempio nella richiesta di connessione ad un server HTPP si ha:

```
client port  18426
server port  80
```

- Le connessioni TCP sono punto-a-punto e full duplex.

I Segmenti TCP

- Ogni **segmento** ha un header fisso di 20 byte più eventuali parti opzionali seguiti da 0 o più byte di dati.



I Flag TCP

- Nel segmento TCP sono presenti 6 bit di flag

URG
l'Urgent Pointer indica la posizione a partire dal numero di sequenza attuale di dati urgenti (es. pressione di CTRL-C per interrompere il programma remoto).

ACK
Indica se il campo Acknowledgement number è valido.

PSH
Indica dati di tipo PUSH ovvero si richiede di consegnare subito i dati senza bufferizzarli.

RST
Richiesta di re-inizializzazione di una connessione diventata instabile. Viene anche usato per rifiutare un segmento non valido o l'apertura di una connessione.

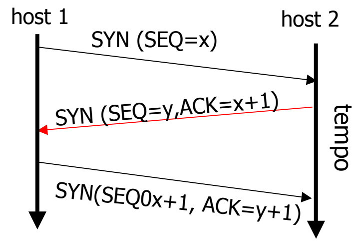
I Flag TCP

SYN
Viene utilizzato per creare connessioni. La richiesta di connessione è caratterizzata da SYN=1 e ACK=0. La risposta di connessione contiene un ack e quindi ha SYN=1 e ACK=1. Individua i segmenti CONNECTION REQUEST e CONNECTION ACCEPTED.

FIN
Viene utilizzato per chiudere una connessione (il mittente non ha altri dati da spedire).

Apertura della Connessione

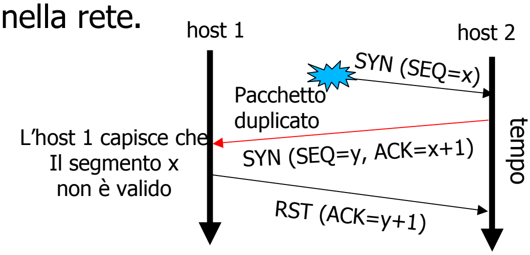
- Si utilizza un protocollo **3-way handshake**



- Se il TCP ricevente non verifica la presenza di nessun processo in attesa sulla porta destinazione manda un segmento di rifiuto della connessione (RST).

Gestione di Pacchetti Duplicati

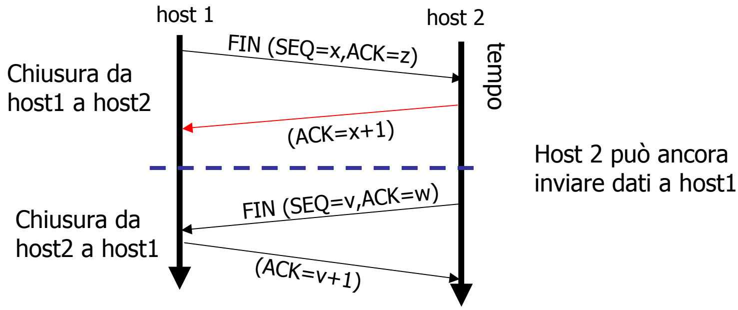
- I pacchetti possono essere memorizzati e ricomparire nella rete.



- La numerazione iniziale è fatta con un orologio locale (tick=4 μs)
- L'intervallo dei numeri di sequenza (32 bit) garantisce che non venga riutilizzato lo stesso numero prima di qualche ora
- A causa del **time to live** dei pacchetti IP, segmenti con lo stesso numero non possono coesistere sulla rete

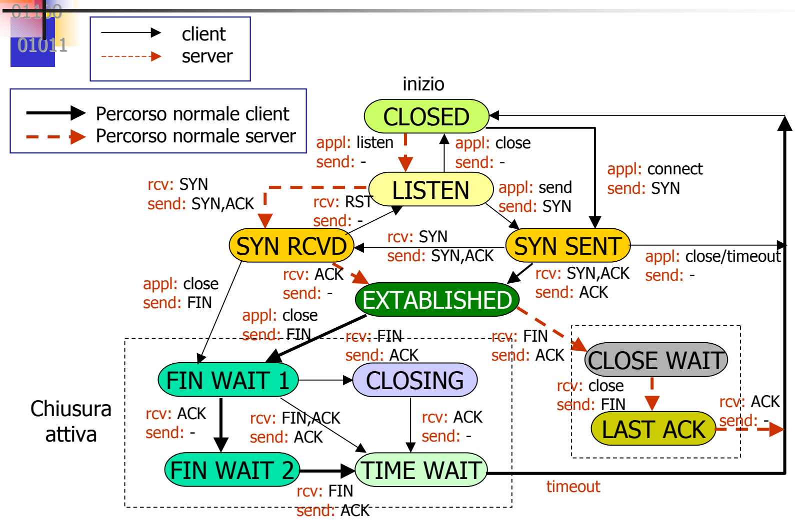
Chiusura della Connessione

- La connessione è full-duplex e le due direzioni devono essere chiuse indipendentemente.



- Se l'ack di un messaggio FIN si perde l'host mittente chiude comunque la connessione dopo un timeout.

Il diagramma degli stati TCP



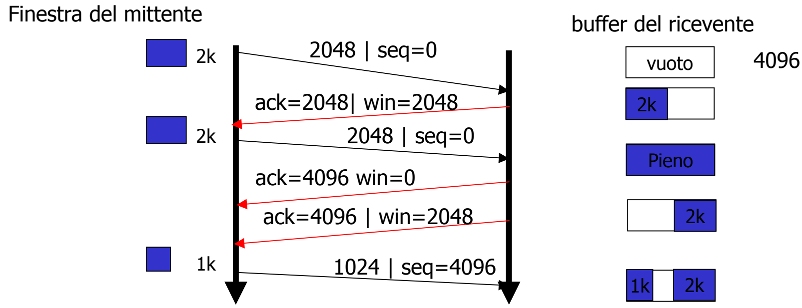
Il timeout MSL

- Il **Maximum Segment Lifetime** (MSL=2 min) indica il massimo tempo per il quale un segmento TCP può sopravvivere nella rete prima di essere scartato.
- Attendere 2MSL nello stato TIME WAIT garantisce che tutti i segmenti relativi alla connessione siano spariti dalla rete.
- Nello stato TIME WAIT si impedisce che nel client possa aprirsi una connessione con lo stesso indirizzo di quella appena chiusa (porte+IP).

Il timeout MSL

- Un vincolo più rigido usato in molte implementazioni è che non venga riusato il numero di porta locale.
- Per il server questo non avviene (la porta essendo pubblicata deve rimanere attiva).

Gestione dei Flussi di Dati



- Se il ricevente indica una finestra 0 il mittente non può trasmettere dati.
- Il mittente può inviare un segmento di un byte per forzare il destinatario a indicare il prossimo byte atteso e l'ampiezza della finestra per non rimanere in attesa infinita se si perdono pacchetti.

Flusso di dati interattivi

- Si consideri il caso di una connessione interattiva (es. Telnet).
- Non si possono accumulare i dati ma occorre inviare segmenti piccoli.
- Il 90% dei segmenti telnet porta circa 10 byte.
- Nel caso limite si ha un segmento per ogni carattere battuto.
- Il ricevente server in genere fa un echo del carattere battuto.

Flusso di Dati Interattivi

- Per gestire un singolo carattere in una connessione interattiva Telnet.

  - Segmento dal client col carattere battuto
  - (20 IP + 20 TCP + 1byte = 41byte)
  - Segmento di ack dal server al client (40 byte)
  - Segmento di echo dal server (41 byte)
  - Segmento di ack dal client (40 byte).

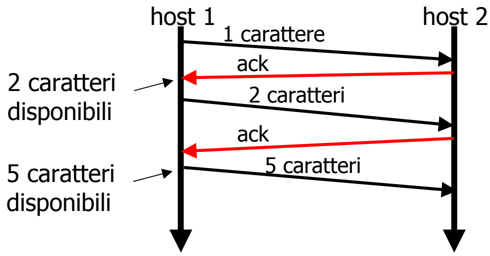
  In totale si userebbero 162 byte in 4 segmenti TCP per 1 solo carattere !!

Ack Ritardati

- Normalmente il TCP non invia un ack istantaneamente ma ritarda l'invio sperando di avere dati da spedire con l'ack.
- Questa tecnica è detta **Ack piggybacking**.
- Molte implementazioni usano un ritardo di 200ms.

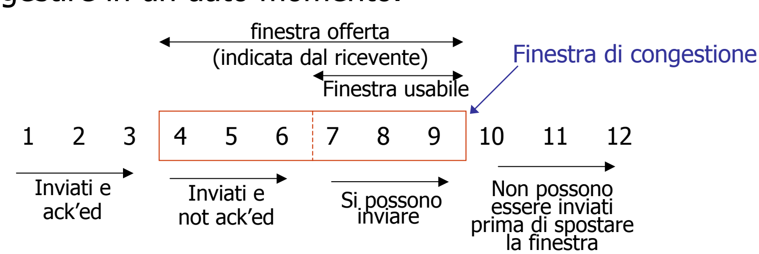
L'Algoritmo di Nagle

- Ha effetto per connessioni lente (es. WAN).
- Si accumulano i dati fino a che non si riceve l'ack per il segmento inviato in precedenza.
- In alcuni casi (alta interattività) deve essere disabilitato (es. mouse in Xwindows).



Flussi di Dati e Finestre di Congestione

- Viene utilizzato un protocollo a finestra scorrevole (*sliding window*).
- Il ricevente indica la dimensione della finestra che può gestire in un dato momento.



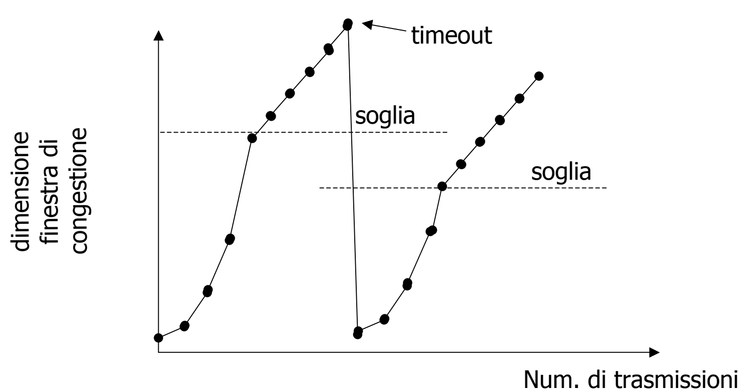
- La finestra di dati trasmissibili ancora senza aspettare l'ack è ottenuta dall'ampiezza della finestra e dal numero dell'ultimo byte ricevuto.

Controllo di congestione

- Il TCP adatta la velocità di trasmissione alla capacità della rete.
- Si utilizza la finestra di congestione che ha la stessa funzionalità della finestra di trasmissione usata per il ricevente.
- La dimensione della finestra di congestione è ridotta se scade il timeout di ritrasmissione, mentre è aumentata se il pacchetto viene consegnato prima del timeout.

Controllo di congestione di Internet

- Un esempio di funzionamento dell’algoritmo di controllo della congestione in Internet (con uso di una **soglia**).



TCP Timeout e Ritrasmissione

- TCP utilizza un timeout di attesa dell'ack dopo di che provvede alla ritrasmissione dei dati.
- Il problema è determinare il valore del timeout migliore (i ritardi possono essere molto variabili nel tempo sulla rete)
  - Se il timeout è troppo piccolo si fanno ritrasmissioni inutili
  - Se il timeout è troppo elevato si avranno ritardi di trasmissione

- Si utilizza un algoritmo di stima del migliore timeout basato sulla misura del **Round-Trip Time** (**RTT**).

Stima del Timeout

- Per ogni connessione si tiene una stima di RTT, aggiornandola per ogni pacchetto con

$$RTT_i = \alpha\, RTT_{i-1} + (1-\alpha)\, T_{rtt}(pkt_i)$$

- Si stima poi la deviazione media

$$D_i = \alpha\, D_{i-1} + (1-\alpha)\, |RTT_i - T_{rtt}(pkt_i)|$$

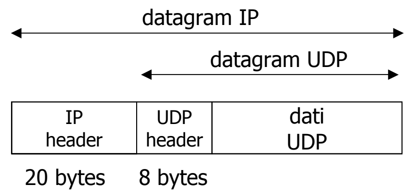
- E si sceglie

$$timeout = RTT + 4*D$$

- Ci sono altre soluzioni (es. algoritmo di Karn: raddoppio del timeout ad ogni fallimento).

UDP : Trasporto senza Connessione

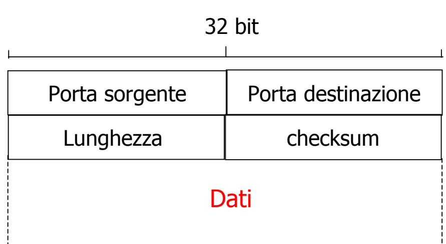
- Ogni operazione di output produce esattamente un datagram UDP che comporta l'invio di un datagram IP.



- UDP non garantisce affidabilità di consegna.
- Se il datagram eccede la **MTU** (Maximum Transfer Unit) della rete, esso viene frammentato.

Header UDP

- Preambolo (header) di un datagram UDP.



- Le **porte** UDP sono indipendenti da quelle TCP.
- La **lunghezza** in byte comprende sia i dati che l'header (≥8).
- Il **checksum** comprende anche uno pseudoheader che contiene le informazioni IP.