RETI DI CALCOLATORI

Prof. PIER LUCA MONTESSORO

Facoltà di Ingegneria
Università degli Studi di Udine

 1

Nota di Copyright

 2

Lezione 25

Il livello di trasporto, TCP, UDP

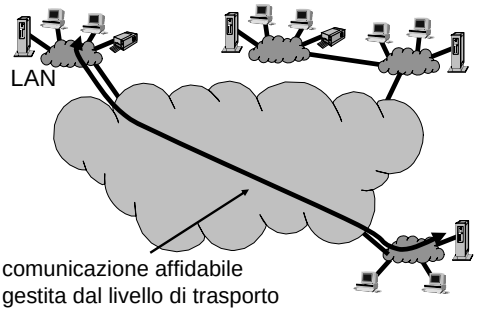
 3

Lezione 25: indice degli argomenti

- Servizi forniti ai livelli superiori
- Indirizzi del livello di trasporto
- Creare e chiudere le connessioni
- Gestire le connessioni
- TCP
- UDP
- Imbustamento multiplo: visione d'insieme

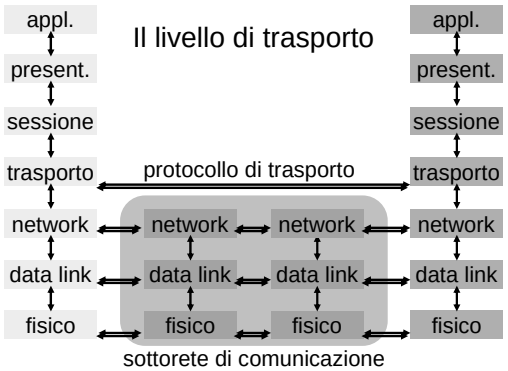
 4

Il livello di trasporto



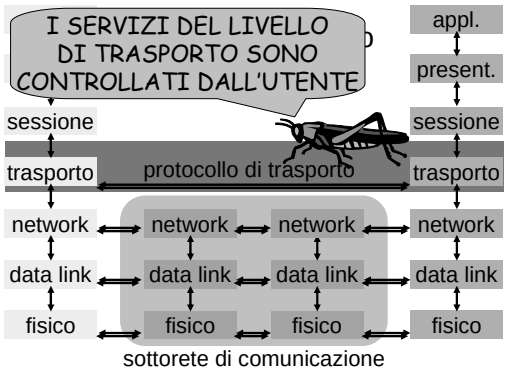
 5

Il livello di trasporto



 6

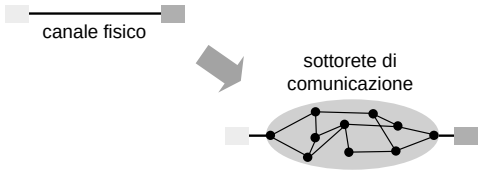
 7



 8

Problemi generali del livello di trasporto

- Simili a quelli del livello data link, ma il canale fisico è in questo caso l'intera sottorete di comunicazione



 9

Problemi generali del livello di trasporto

- Alcune conseguenze:
  - l'altra estremità della connessione può non esistere (a differenza dei canali fisici, che al limite possono essere guasti) senza che lo sappiamo
  - sono possibili ritardi di decine di secondi
  - i pacchetti possono seguire strade diverse e arrivare in ordine diverso da quello di trasmissione

 10

Servizi forniti ai livelli superiori

 11

Servizi forniti ai livelli superiori

- Fornisce un servizio efficiente, affidabile e conveniente alle applicazioni (o al livello sessione)
  - i servizi possono essere connessi o non connessi, proprio come al livello di rete
- Mette a disposizione funzioni di libreria per i programmi applicativi
- Gestisce il multiplexing del traffico di rete tra le varie applicazioni

 12

Socket per TCP (Berkeley UNIX)

- Esempio di interfaccia verso i programmi applicativi
- Si basano su un insieme di primitive per
  - rendere disponibile un punto di accesso al servizio per i client remoti
  - aprire/chiudere una connessione
  - scambiare dati

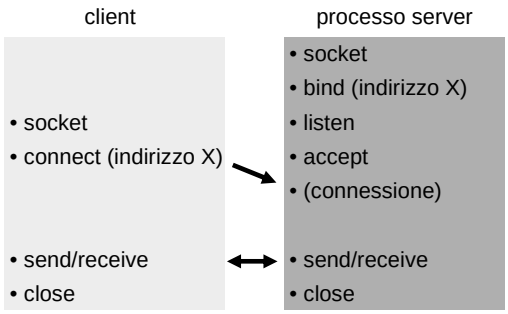
 13

Socket per TCP (Berkeley UNIX)

socket	crea un punto finale di comunicazione
bind	associa un indirizzo locale a un socket
listen	si rende disponibile ad accettare connessioni
accept	si blocca nell'attesa di una connessione
connect	tenta di stabilire una connessione
send	invia dati sulla connessione
receive	riceve dati dalla connessione
close	chiude la connessione

 14

Socket per TCP (Berkeley UNIX)



 15

Indirizzi del livello di trasporto

 16

Indirizzi del livello di trasporto

- TSAP: Transport Service Access Point
- In Internet:
  - coppie <indirizzo IP, porta locale>
  - esempio:
    - 158.110.1.2:23 (porta 23: telnet)
- Definiscono punti di accesso presso cui i processi possono attendere le connessioni

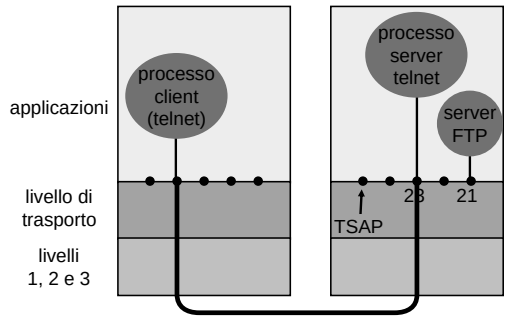
 17

Indirizzi del livello di trasporto

- Come conscerli?
  - noti a priori (le "well known ports" di TCP)
  - elencati in un "name server" o "directory server"

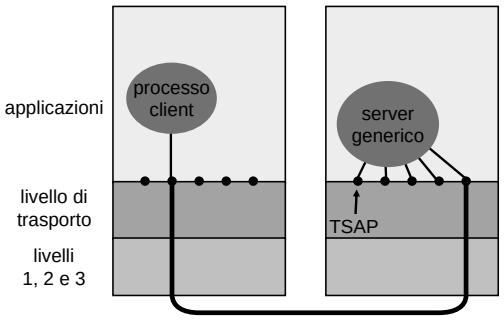
 18

Indirizzi del livello di trasporto



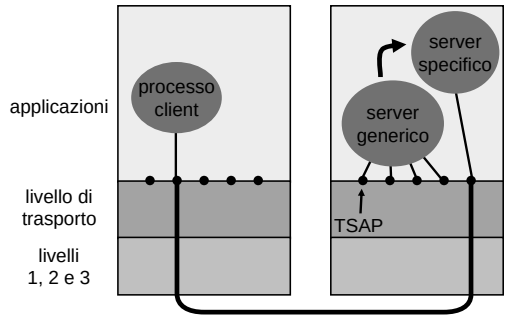
 19

Server generico



 20

Server generico

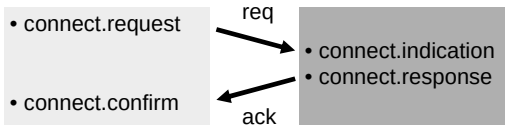


 21

Creare e chiudere le connessioni

 22

Creare una connessione



- Problema:
  - se gli ack non arrivano in tempo, il mittente rispedisce i pacchetti, che arrivano duplicati, aprendo una nuova connessione parallela

 23

Creare una connessione

- È necessario ridurre il massimo tempo di vita di un pacchetto (questo richiede un supporto da parte del livello rete, per esempio, il “time to live” di IP)
- A questo punto ogni connessione può numerare le proprie TPDU con numeri di sequenza differenti (ad esempio estratti da un orologio)
- Quando i numeri di sequenza vengono riutilizzati, le vecchie TPDU sono estinte

 24

Chiudere una connessione

- Chiusura asimmetrica
  - la chiusura della connessione avviene contemporaneamente in entrambe le direzioni
  - è simile a quella del sistema telefonico
  - possono essere persi dati in transito e non ancora arrivati al momento della chiusura della connessione

 25

Chiudere una connessione

- Chiusura simmetrica
  - ogni direzione della connessione è chiusa indipendentemente dall'altra (messaggi DISCONNECT.REQUEST indipendenti nelle due direzioni)

 26

Chiudere una connessione

- Problema: come sapere quando l'attività è terminata?

  (una TPDU in arrivo può richiedere la trasmissione di nuovi dati)
  - decisione unilaterale → possibile perdita di dati
  - protocollo per riscontro → possibile attesa infinita (hang)

 27

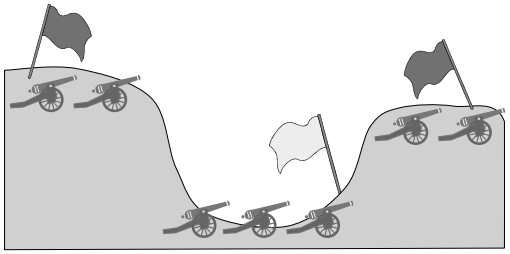
Chiudere una connessione

 28

Chiudere una connessione

 29

Il problema dei due eserciti



 30

Il problema dei due eserciti

 31

Un esempio di timeout

 32

Gestire le connessioni

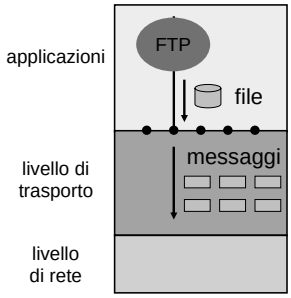
 33

Frammentazione

- I dati ricevuti dalle applicazioni devono essere suddivisi in messaggi di dimensione adeguata alle capacità del livello network

 34

Frammentazione



 35

Controllo di flusso

- Protocolli sliding window
- Più complesso che nel caso del livello data link
  - il livello di trasporto deve gestire più connessioni contemporanee
  - sono necessarie opportune strategie di gestione dei buffer per memorizzare i messaggi

 36

Gestione dei guasti

- Perdita di pacchetti
  - gestita tramite i messaggi di riscontro, timeout e ritrasmissioni
- Guasto temporaneo dell'host di destinazione
  - quando riprende a funzionare reinizializza le tabelle e perde memoria delle connessioni
  - richiede tecniche di logging sofisticate simili a quelle dei sistemi transazionali

 37

Il livello di trasporto in Internet: TCP e UDP

 38

Porte

- Numeri interi scritti su 16 bit
- Sono i TSAP del TCP
- Definiscono i "punti" in cui i processi server attendono le connessioni e i dati
- RFC 1700 ("Assigned numbers") definisce le "well-known ports": valori inferiori a 1024 (originariamente il limite era posto a 255) che sono riservate a processi di sistema

 39

Esempi di well-known ports (RFC 1700)

- FTP: 21
- Telnet: 23
- SMTP: 25
- DNS: 53
- WWW-HTTP: 80
- POP3: 110
- IMAP2: 143
- SNMP: 161

 40

TCP Transmission Control Protocol

- Gestisce connessioni full duplex punto-punto
  - no multicast né broadcast
- Fornisce alle applicazioni un flusso di dati affidabile
  - i confini delle operazioni di scrittura del processo trasmittente non vengono conservati
- Se necessario, i dati sono divisi in segmenti

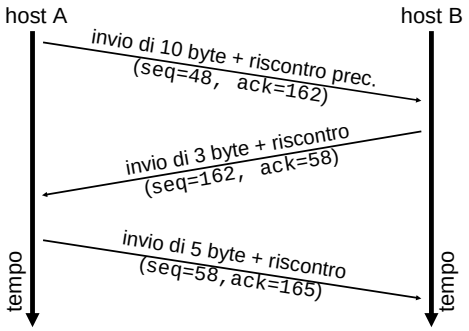
 41

TCP: numeri di sequenza

- La numerazione non è orientata ai segmenti, ma al byte nel flusso continuo di dati
- Il numero di sequenza di un segmento è sempre il numero di sequenza del primo byte contenuto nel campo dati
- Nei messaggi di riscontro del protocollo TCP viene sempre indicato il numero di sequenza del byte successivo che il ricevitore sta aspettando (rispetto all'ultimo ricevuto in ordine)

 42

TCP: numeri di sequenza



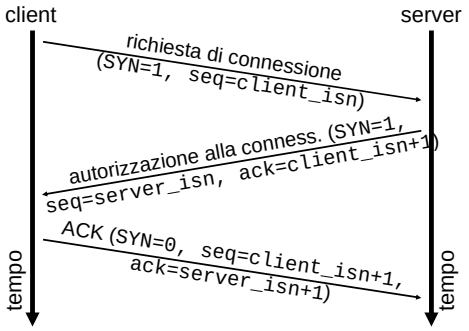
 43

TCP: apertura della connessione (handshake a tre vie)

1. Il client spedisce un *segmento SYN* (flag **`SYN=1`**) notificando il suo numero iniziale di sequenza (**`client_isn`**)
2. Il server risponde autorizzando la connessione: **`SYN=1`**, notifica il proprio numero iniziale di sequenza (**`server_isn`**) e conferma il segmento SYN del client (**`ack=client_isn+1`**)
3. Il client conferma la ricezione del segmento SYN del server: **`SYN=0`, `seq=client_isn+1`, `ack=server_isn+1`**

 44

TCP: apertura della connessione



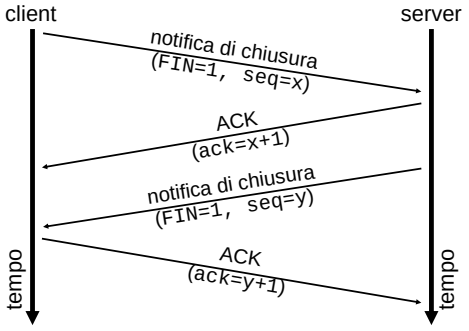
 45

TCP: chiusura della connessione (handshake a quattro vie)

1. L'host A spedisce all'host B un *segmento FIN* (flag **`FIN=1`**) notificando l'intenzione di chiudere la connessione
2. L'host B invia un riscontro e la connessione viene chiusa in un verso; B può continuare a inviare dati
3. Quando l'host B termina la trasmissione invia un segmento FIN
4. L'host A invia un messaggio di riscontro

 46

TCP: chiusura della connessione



 47

TCP: controllo di flusso, congestione, ed errori di trasmissione

- Gestisce il controllo di flusso e gli errori di trasmissione (o perdita di pacchetti) con lo stesso meccanismo
  - protocollo sliding window con numerazione orientata al byte (i riscontri, come visto, contengono il numero di sequenza del prossimo byte atteso)

 48

TCP: segmenti ricevuti fuori ordine

- Il TCP non usa messaggi espliciti di NACK
- Se riceve un segmento fuori ordine (buco nella sequenza di dati) invia un duplicato dell'ACK per l'ultimo segmento in ordine ricevuto
- Il trasmettitore, quando riceve tre ACK duplicati, li interpreta come NACK per il segmento seguente ("ritrasmissione veloce", RFC 2581)

 49

TCP: finestra di ricezione

- La dimensione della finestra di ricezione varia in funzione della dimensione del buffer (fissa) e i dati ricevuti ma non ancora prelevati dal processo applicativo
- La dimensione della finestra di ricezione viene comunicata al trasmettitore nel campo "window size" dei segmenti inviati

 50

TCP: finestra di trasmissione

- La dimensione della finestra di trasmissione viene modificata dinamicamente in funzione della capacità della rete e del ricevente

```
finestra di trasmissione =
      min (finestra di ricezione,
           finestra di congestione)
```

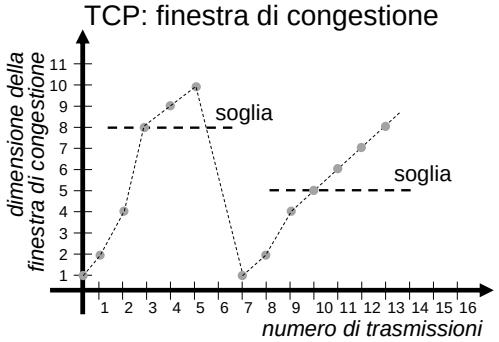
 51

TCP: finestra di congestione

- Inizialmente a 1, man mano che arrivano riscontri cresce esponenzialmente fino ad un valore di soglia predefinito, poi linearmente)
- Se un riscontro non arriva entro il timeout, il valore di soglia viene abbassato alla metà dell'attuale valore della finestra di congestione e questa riparte da 1

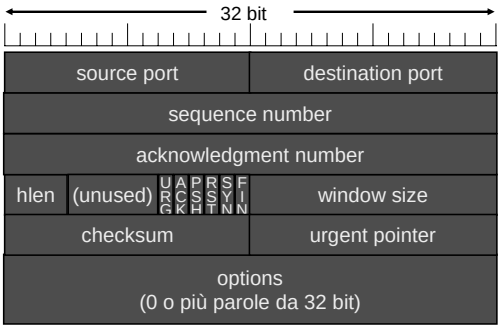
 52

TCP: finestra di congestione



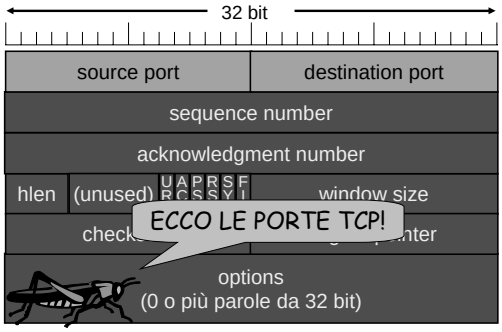
 53

Header TCP

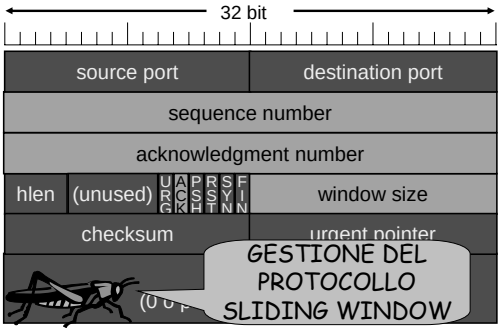


 54

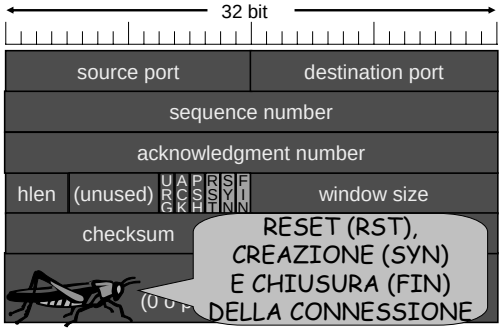
Header TCP



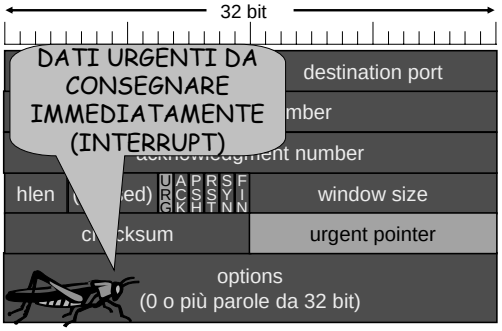
Header TCP



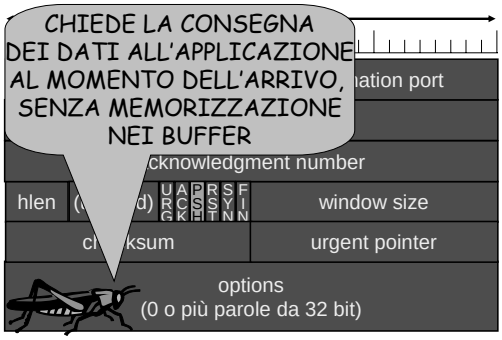
Header TCP



Header TCP



Header TCP



UDP
User Data Protocol

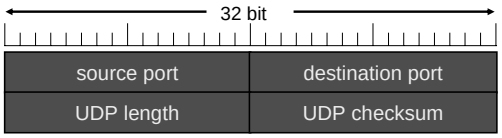
- Protocollo non connesso
- Si limita ad aggiungere ad IP le funzionalità di:
  - multiplexing dei dati tra le diverse applicazioni (gestione delle porte)
  - generazione e verifica della checksum per l'integrità dei dati

UDP
User Datagram Protocol

- Più snello ed efficiente di TCP
- Applicabile nelle seguenti condizioni:
  - si opera su rete locale
  - l'applicazione mette tutti i dati in un singolo pacchetto
  - non è importante che tutti i pacchetti arrivino a destinazione (es. multimedia)
  - l'applicazione gestisce meccanismi di ritrasmissione

 61

Header UDP

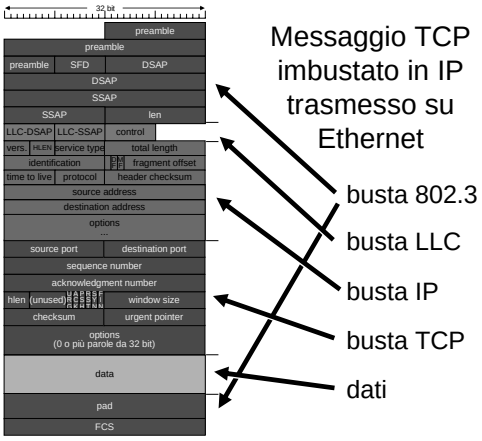


 62

Imbustamento multiplo: visione d'insieme

 63

Messaggio TCP imbustato in IP trasmesso su Ethernet



Esempio: telnet (apertura di connessione)

```
ADDR   HEX
0000   08 00 2B 00 9A 7D 08 00 5A 1A 60 BE 00 2F 06 06
0010   03 45 00 00 2C 58 E3 00 00 3C 06 18 53 82 C0 04
0020   12 82 C0 04 04 04 14 00 17 33 A8 26 20 56 67 24
0030   01 60 12 40 00 72 2D 00 00 02 04 05 AB
```

 65

Esempio: telnet (apertura di connessione)

```
ADDR   HEX
0000   08 00 2B 00 9A 7D 08 00 5A 1A 60 BE 00 2F 06 06
0010   03 45 00 00 2C 58 E3 00 00 3C 06 18 53 82 C0 04
0020   12 82 C0 04 04 04 14 00 17 33 A8 26 20 56 67 24
0030   01 60 12 40 00 72 2D 00 00 02 04 05 AB

DLC: ----- DLC Header -----
DLC: Destination = Station 08-00-2B-00-9A-7D
DLC: Source = Station 80-00-5A-08-00-5A
DLC: 802.3 len = 47 byte

LLC: ----- LLC Header -----
LLC: DSAP = 06 (IP), SSAP = 06 (IP)
LLC: Unnumbered frame: UI
```

 66

Esempio: telnet (apertura di connessione)

```
ADDR   HEX
0000   08 00 2B 00 9A 7D 08 00 5A 1A 60 BE 00 2F 06 06
0010   03 45 00 00 2C 58 E3 00 00 3C 06 18 53 82 C0 04
0020   12 82 C0 04 04 04 14 00 17 33 A8 26 20 56 67 24
0030   01 60 12 40 00 72 2D 00 00 02 04 05 AB

     IP: ----- IP Header -----
     IP: Version = 4, header length = 20 bytes
     IP: Type of service = 00
     IP: 000. .... = routine
     IP: ...0 .... = normal delay
     IP: .... 0... = normal throughput
     IP: .... .0.. = normal reliability
     IP: Total length = 44 bytes
     IP: Identification = 22755
     IP: Flags = 0X
     IP: .0.. .... = may fragment
     IP: ..0. .... = last fragment
     IP: Fragment offset = 0 bytes
```


Esempio: telnet (apertura di connessione)

```
ADDR   HEX
0000   08 00 2B 00 9A 7D 08 00 5A 1A 60 BE 00 2F 06 06
0010   03 45 00 00 2C 58 E3 00 00 3C 06 18 53 82 C0 04
0020   12 82 C0 04 04 04 14 00 17 33 A8 26 20 56 67 24
0030   01 60 12 40 00 72 2D 00 00 02 04 05 AB

     IP: ----- IP Header -----

     (segue)

     IP: Time to live = 60 seconds/hops
     IP: Protocol = 6 (TCP)
     IP: Header checksum = 1853 (correct)
     IP: Source address = [130.192.4.18]
     IP: Destination address = [130.192.4.4]
     IP: No options
```


Esempio: telnet (apertura di connessione)

```
ADDR   HEX
0000   08 00 2B 00 9A 7D 08 00 5A 1A 60 BE 00 2F 06 06
0010   03 45 00 00 2C 58 E3 00 00 3C 06 18 53 82 C0 04
0020   12 82 C0 04 04 04 14 00 17 33 A8 26 20 56 67 24
0030   01 60 12 40 00 72 2D 00 00 02 04 05 AB

     TCP: ----- TCP header -----
     TCP: Source port = 1044
     TCP: Destination port = 23 (Telnet)
     TCP: Initial sequence number = 866657824
     TCP: Acknowledgment number = 1449600001
     TCP: Data offset = 24 bytes
     TCP: Flags = 12
     TCP: ..0. .... = (No urgent pointer)
     TCP: ...1 .... = Acknowledgment
     TCP: .... 0... = (No push)
     TCP: .... .0.. = (No reset)
     TCP: .... ..1. = SYN
     TCP: .... ...0 = (No FIN)
```


Esempio: telnet (apertura di connessione)

```
ADDR   HEX
0000   08 00 2B 00 9A 7D 08 00 5A 1A 60 BE 00 2F 06 06
0010   03 45 00 00 2C 58 E3 00 00 3C 06 18 53 82 C0 04
0020   12 82 C0 04 04 04 14 00 17 33 A8 26 20 56 67 24
0030   01 60 12 40 00 72 2D 00 00 02 04 05 AB

     TCP: ----- TCP header -----

     (segue)

     TCP: Window = 16384
     TCP: Checksum = 722D (correct)
     TCP: (void urgent pointer)
     TCP: Options follow
     TCP: Maximum segment size = 1451
```


Bibliografia

- “Reti di Computer”
  - Capitolo 6
- Libro “Reti locali: dal cablaggio all’internetworking”
  contenuto nel CD-ROM omonimo
  - Capitolo 16

Come contattare il prof. Montessoro

E-mail: montessoro@uniud.it
Telefono: 0432 558286
Fax: 0432 558251
URL: www.montessoro.it

