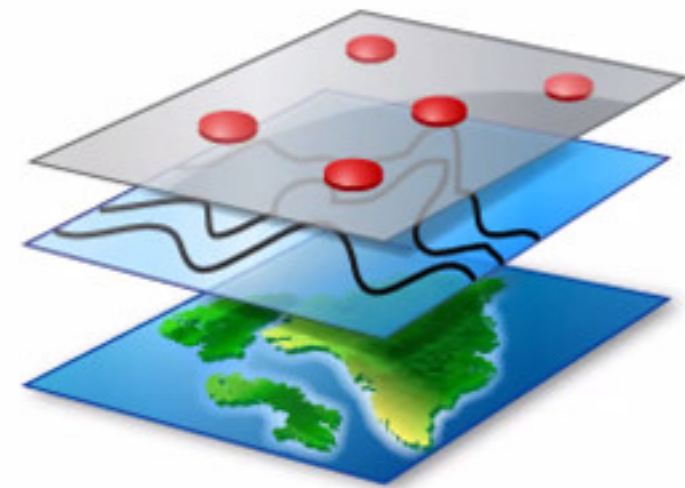
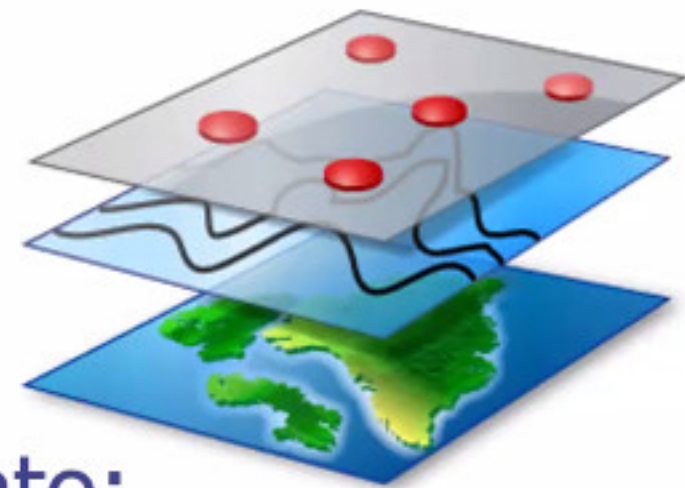


Lo Strato Data Link



- ◆ Dopo lo strato fisico, cominciamo a parlare dello strato immediatamente sovrastante, quello che si occupa del collegamento dati
- ◆ Essenzialmente, si occupa del problema della codifica e decodifica di un flusso di dati

Lo Strato Data Link

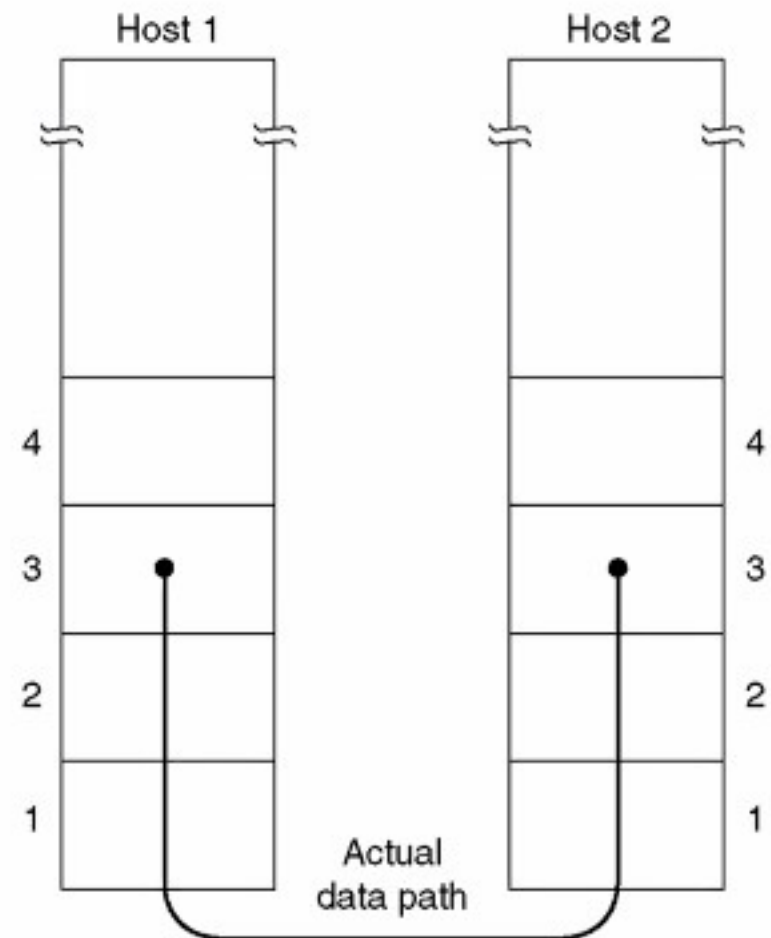
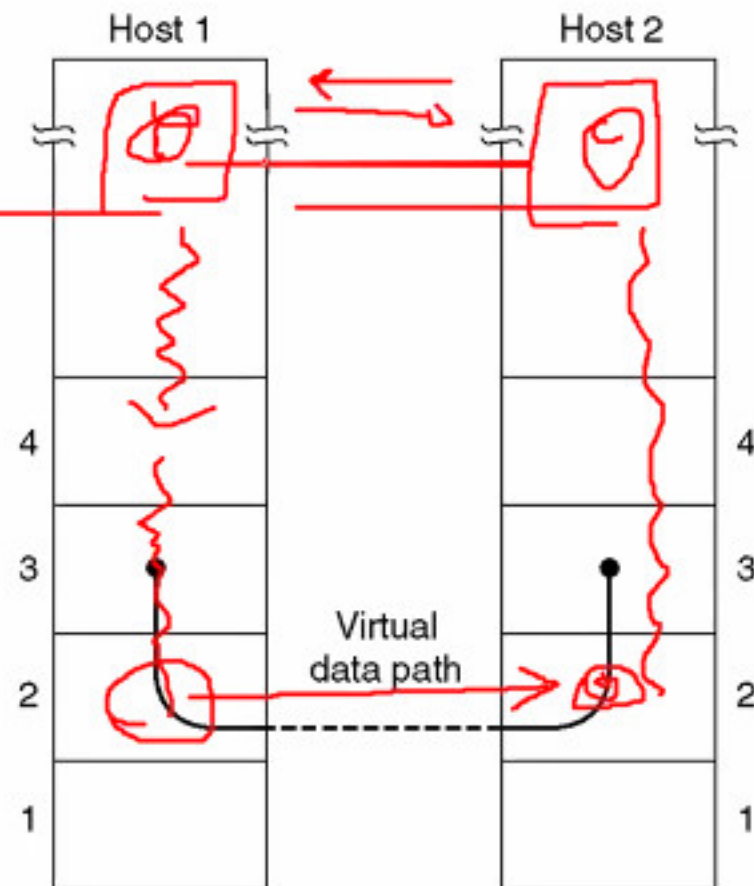


- ◆ Più in dettaglio, questo strato:
- ◆ Fornisce una interfaccia allo strato di rete (network layer) sovrastante
- ◆ Si occupa degli errori di trasmissione (**error control**)
- ◆ Regola il flusso di dati a seconda delle capacità della rete e del ricevente (**flow control**)



Si usa l'interfaccia di rete

- ◆ Che fornisce i servizi necessari al network layer per la trasmissione dati



Varie possibilità

- ◆ Ci sono ovviamente vari modi in cui tale servizio può essere offerto

Unacknowledged connectionless

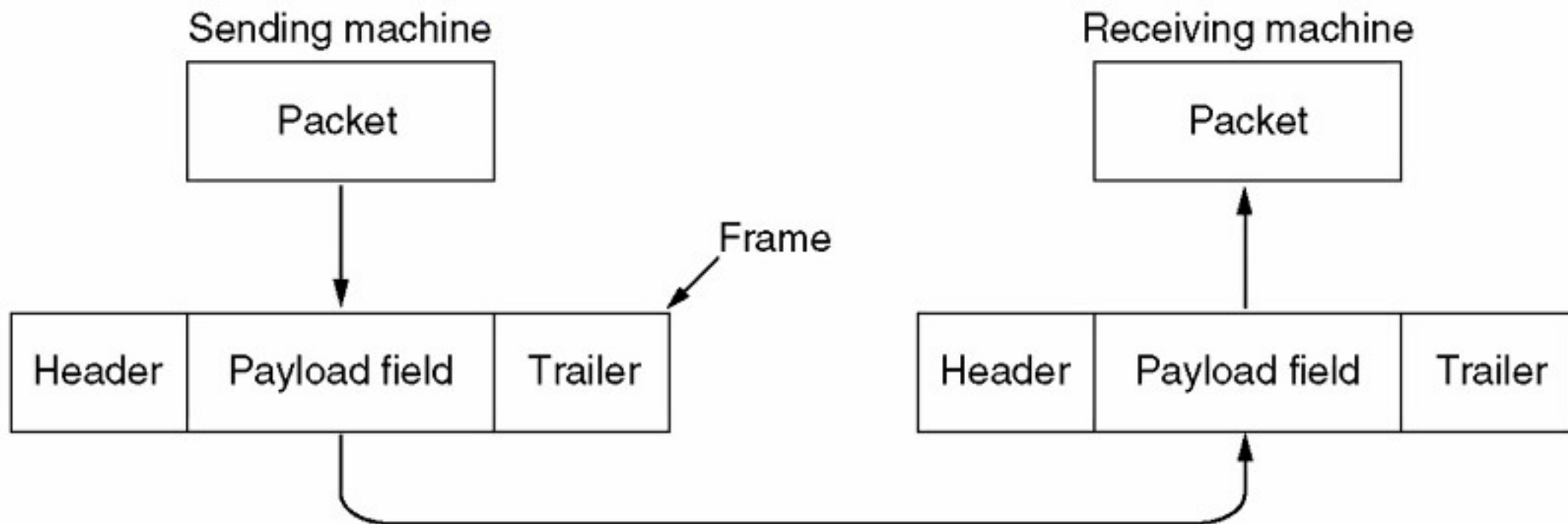
- ◆ In questo servizio, i pacchetti vengono inviati senza aspettare conferma di una eventuale ricevuta (***unacknowledged***),
- ◆ ... e tantomeno senza stabilire una connessione dedicata (***connectionless***)
- ◆ Utile ad esempio per ***voce/streaming media***, o quando il canale è ***molto affidabile***

Acknowledged connectionless

- ◆ L'altro servizio analogo è come il precedente, solo che questa volta i pacchetti ricevuti vengono "confermati" (***acknowledged***) con ricevuta di ritorno

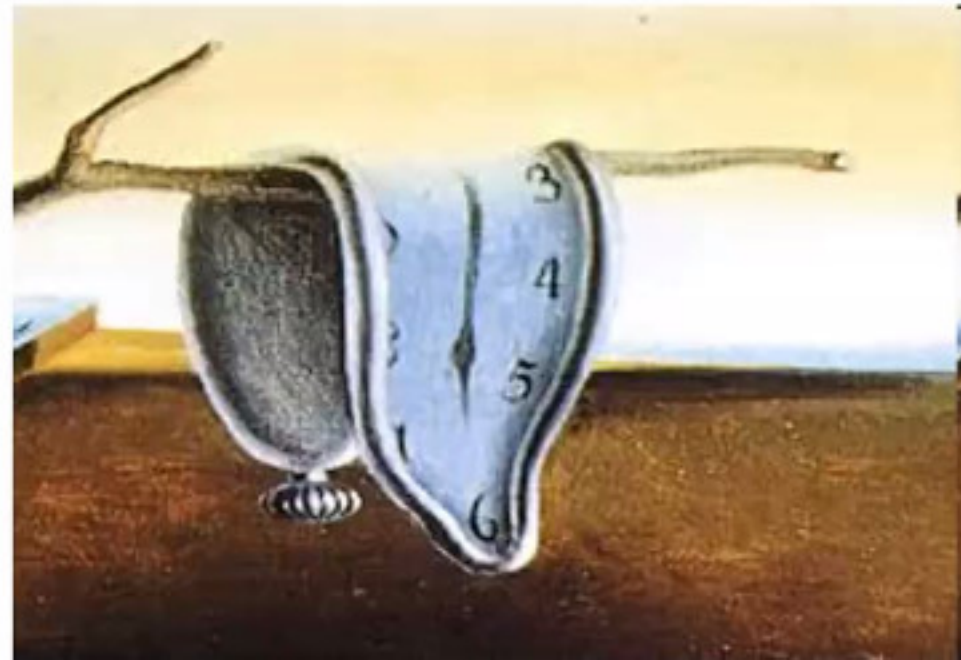
La codifica: framing

- ◆ L'approccio classico è di prendere dei pacchetti dati dallo strato superiore (network), e codificarli in appositi **frames**



Il problema del contorno...

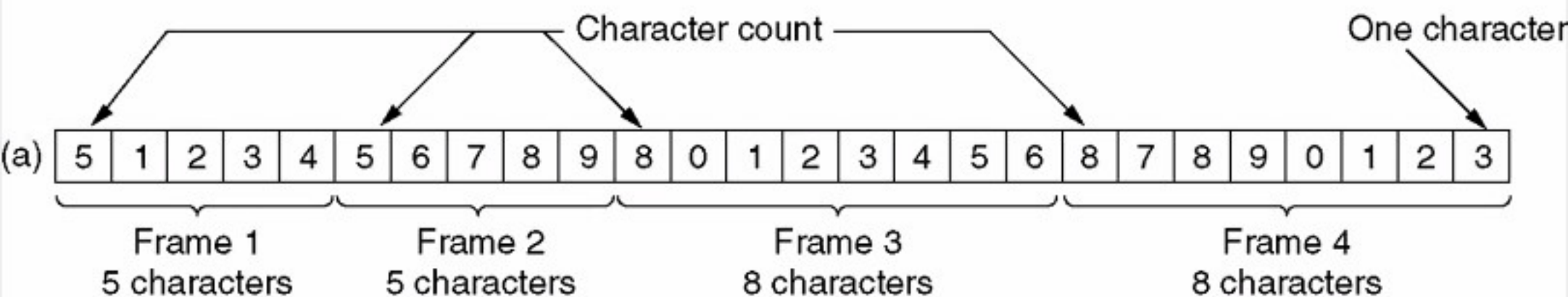
- ◆ Uno dei problemi della trasmissione dei frame è accorgersi di dove un frame *inizia* e *finisce*
- ◆ Si potrebbe usare la sincronizzazione degli orologi, ma è impraticabile per vari (abbastanza ovvi) motivi, quindi si sono scelte altre strade



Il metodo del **character count**

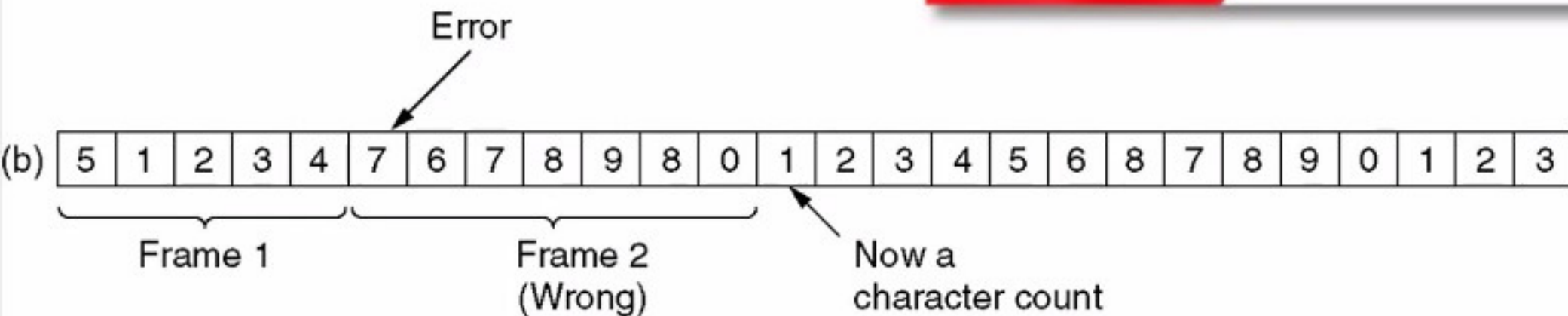
- ◆ Semplicemente, mettiamo nell'header l'informazione ***del numero di caratteri*** che costituiscono il corpo dati (il ***payload***)

Esempio



Problemi...

◆ Se c'è un errore...
disastro!



Altri metodi

- ◆ Il character count è stato uno dei ***primi*** metodi (derivati appunto dai linguaggi di programmazione...)
- ◆ Ma l'ambiente delle reti, dove ci possono essere ***errori*** molto più ***frequentemente*** che non su un desktop, ha imposto l'uso di tecniche ***diverse*** e sempre più sofisticate



Il metodo dei **flag bytes**

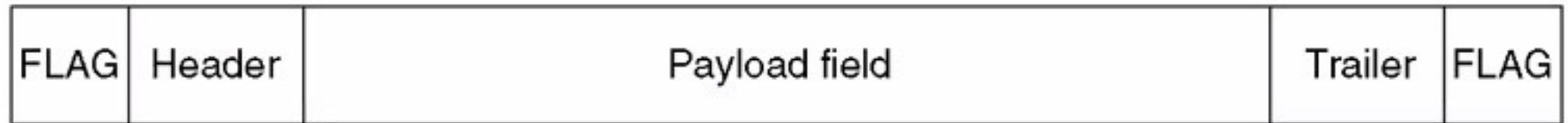
- ◆ Il vero problema del character count è che perdiamo la “sincronizzazione” se c’è un errore
- ◆ → usiamo un byte speciale (***flag byte***) per segnalare l’inizio e la fine di ogni frame
- ◆ (Analogo delle virgolette nei linguaggi di programmazione...)



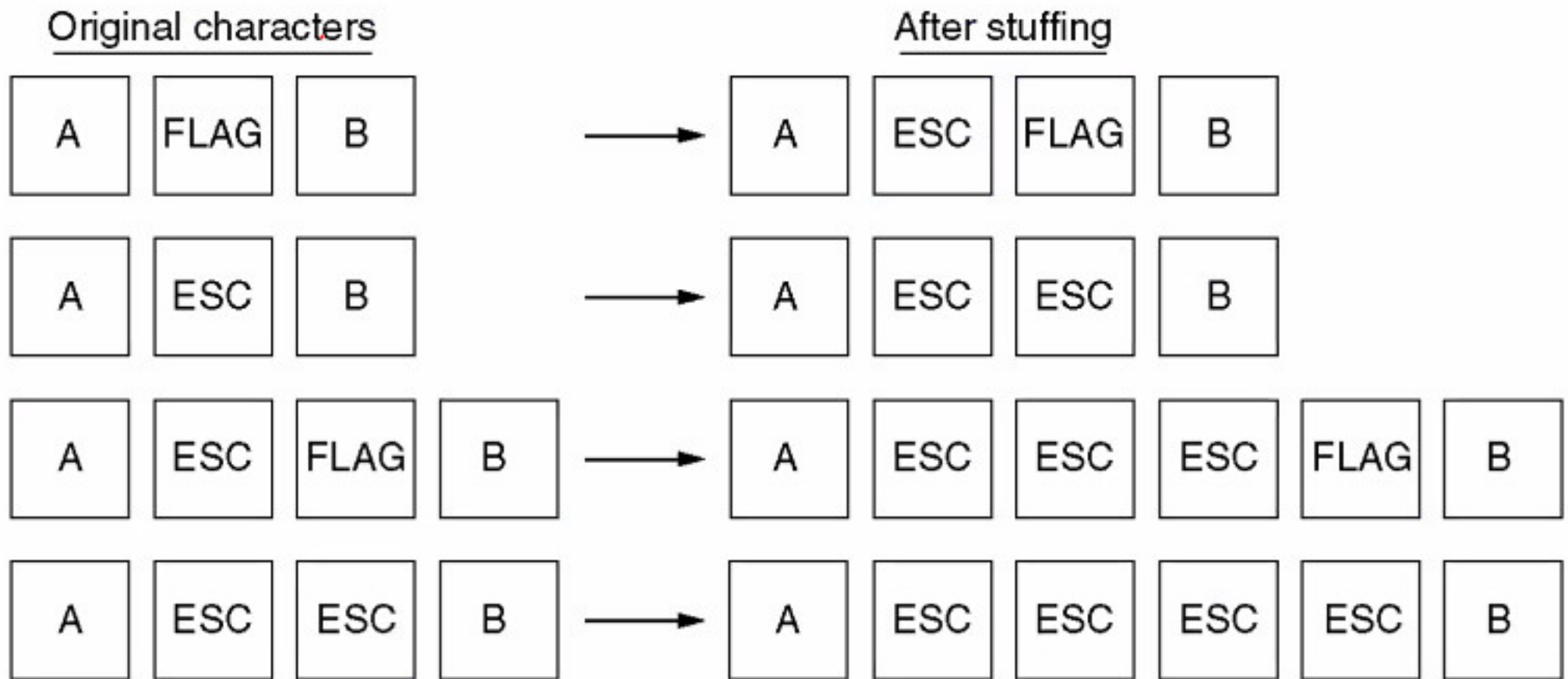
Flag bytes (cont.)

- ◆ Come per le virgolette, serve un metodo che ci permetta di fare ***l'escaping*** delle virgolette stesse, in modo da poter trasmettere qualunque messaggio
- ◆ Nel gergo delle reti, la tecnica si chiama **byte stuffing** (o **character stuffing**)

Esempio




(a)



(b)

Problema



- ◆ Un problema è dovuto all'analogo dei ***linguaggi di programmazione*** quando si è passati dai caratteri ASCII (bytes) a caratteri più globali (UNICODE):

- ◆ usare ***bytes*** o in ogni caso ***grandezze fisse*** prima o poi non va più bene

Il bit stuffing

- ◆ E' l'analogo del byte stuffing, stavolta fatto a livello di bit
- ◆ Ad esempio, si prende come "flag" 01111110 (0 - sei 1 - 0)
- ◆ Escaping: Se lo stream di bits ha cinque 1 di fila, allora dopo il quinto bit viene inserito uno 0

Esempio

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Stuffed bits

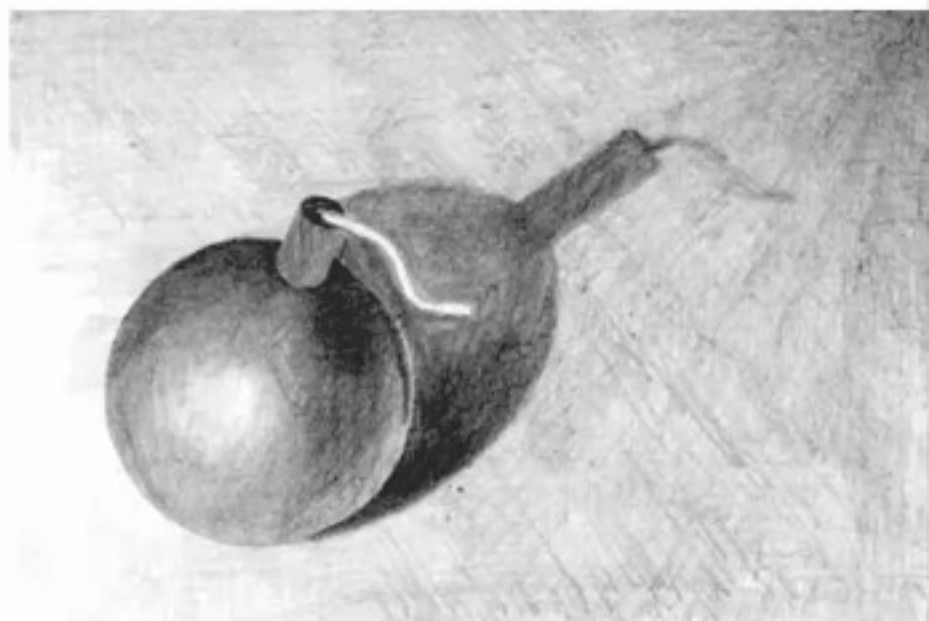
Passiamo ora...

- ◆ ... ad un altro aspetto del data layer, molto importante!



Gli ERRORI

- ◆ Se tecniche come quelle che abbiamo visto ci permettono di sapere ***come identificare*** un frame sulla rete, resta in ogni caso il problema più grave: come comportarsi quando ci sono degli ***errori***



ERROR CONTROL

- ◆ Ci sono essenzialmente due strategie che possiamo seguire:
- ◆ Una è fare **error detection**, cioè sviluppare tecniche che ci dicano se un frame ha subito degli errori durante la trasmissione