UDP AFFIDABILE

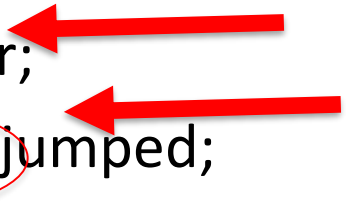
a cura di

Giuseppe Di Cosmo
Salvatore Nedia
Riccardo Chiaretti

Settembre 2017

TRASFERIMENTO DATI AFFIDABILE

- Struttura dati pkt_wnd
  - header pkt;
  - unsigned int ack;
  - int time_fd;
  - long dyn_timer;
  - unsigned char jumped;



TIMER DI RITRASMISSIONE
Creazione & Settaggio

- timerfd_create(int clockid, int flags);
  - Ritorna un file descriptor
    - utilizzato nella struttura pkt_wndw;
    - inserito nell'fd_set della salect().

- timerfd_settime (int fd, int flags, struct itimerspec new_t, struct itimerspec old_t);
  - new_t.it_value;
  - new_t.it_interval.

TIMER DI RITRASMISSIONE
Fisso & Adattativo

- Fisso:
  - impostato tramite la struttura itimerspec.

- Adattativo:
  - dyn_timer;
  - sample_rtt: RTT calcolato per i pacchetti non ritrasmessi;
  - extimated_rtt: media esponenziale ponderata dei valori sample_rtt;
  - dev_rtt: scostamento del sample_tt dalla sua media;
  - evento di timeout = ext_rtt + 4 * dev_rtt.

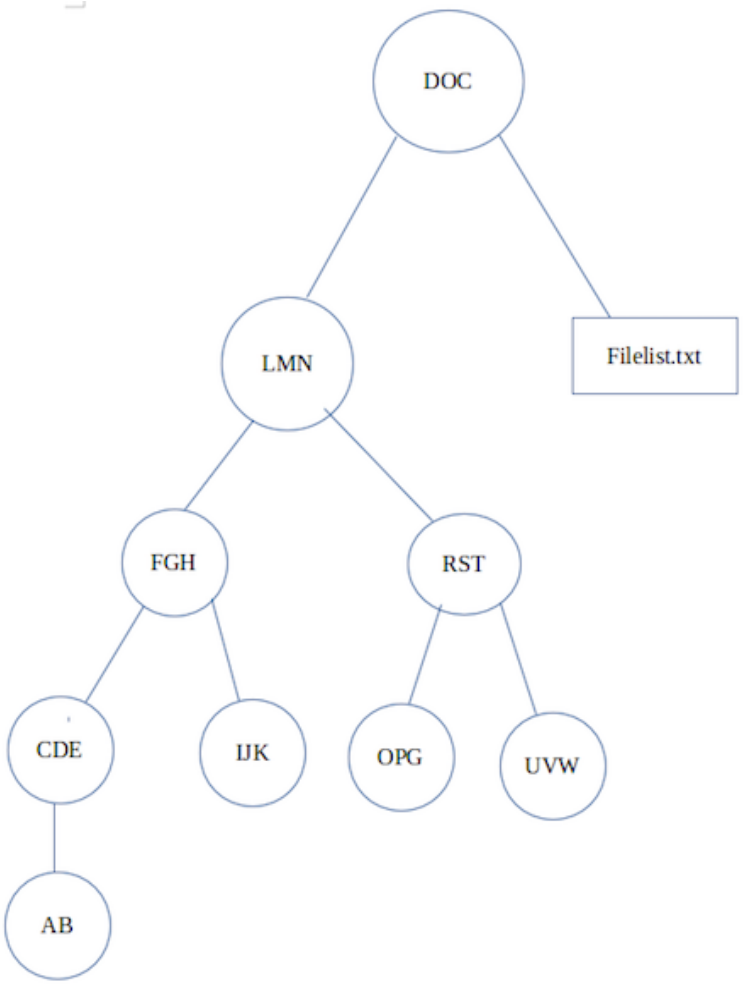
TIMER DI RITRASMISSIONE
Scelte progettuali

- Approccio meccanico e procedurale:
  - creazione;
  - settaggio;
  - evento di timeout.

- Utilizzare la select() per monitorare gli eventi di timeout.

SALVATAGGIO DEI FILE SU SERVER

- Astrazione di una struttura ad albero
  - Funzione insert() per inserire le directory
  - Funzione search() per trovare la giusta directory dato il nome di un file



Struttura del pacchetto

UDP: protocollo connectionless e non affidabile.

Come permettere queste operazioni?

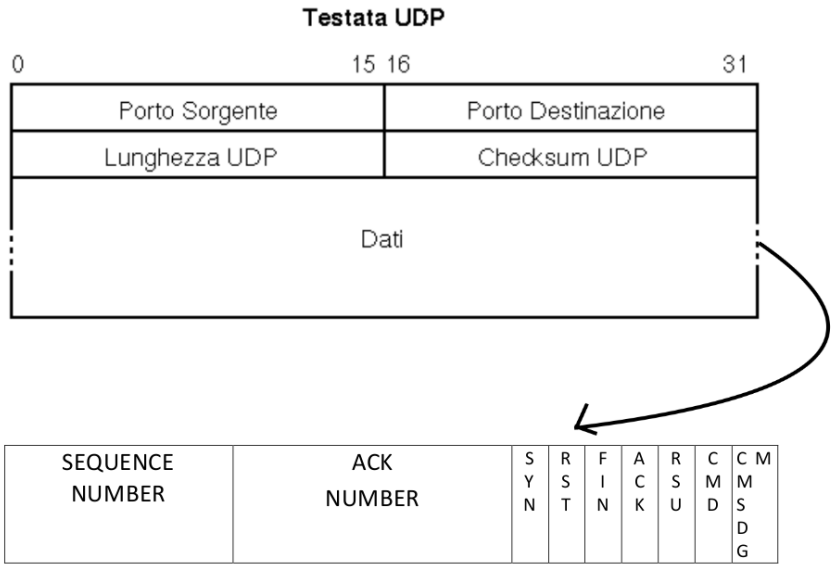
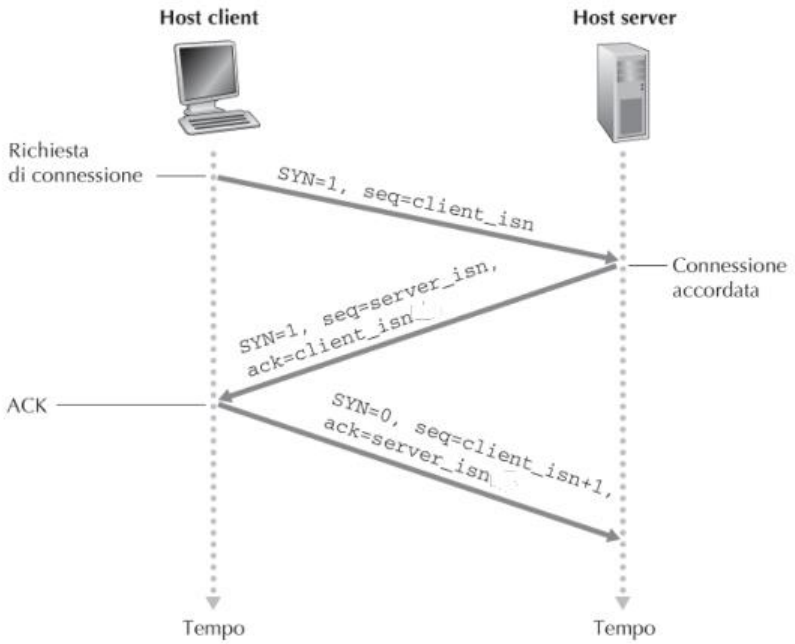


Figura 1: Struttura segmento UDP completo dell'affidabilità

Instaurazione della connessione



Comandi

- PUT: permette il caricamento di un file sul server
- GET: permette il download di un file dal server
- LIST: permette di visualizzare i file presenti sul server

Comando PUT

- PUT: Comando eseguito dal client per caricare un file sul server.
- Necessaria una fase iniziale di handshake tra client e server
- Al termine della comunicazione→ invio di un messaggio di risposta con l'esito dell'operazione.

Comando PUT: implementazione

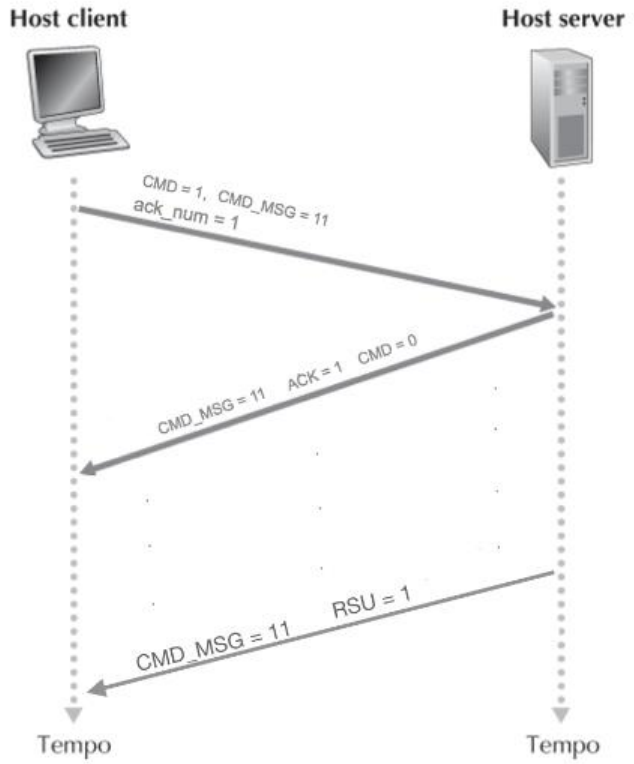


Figura 5: Scambio di pacchetti durante una PUT

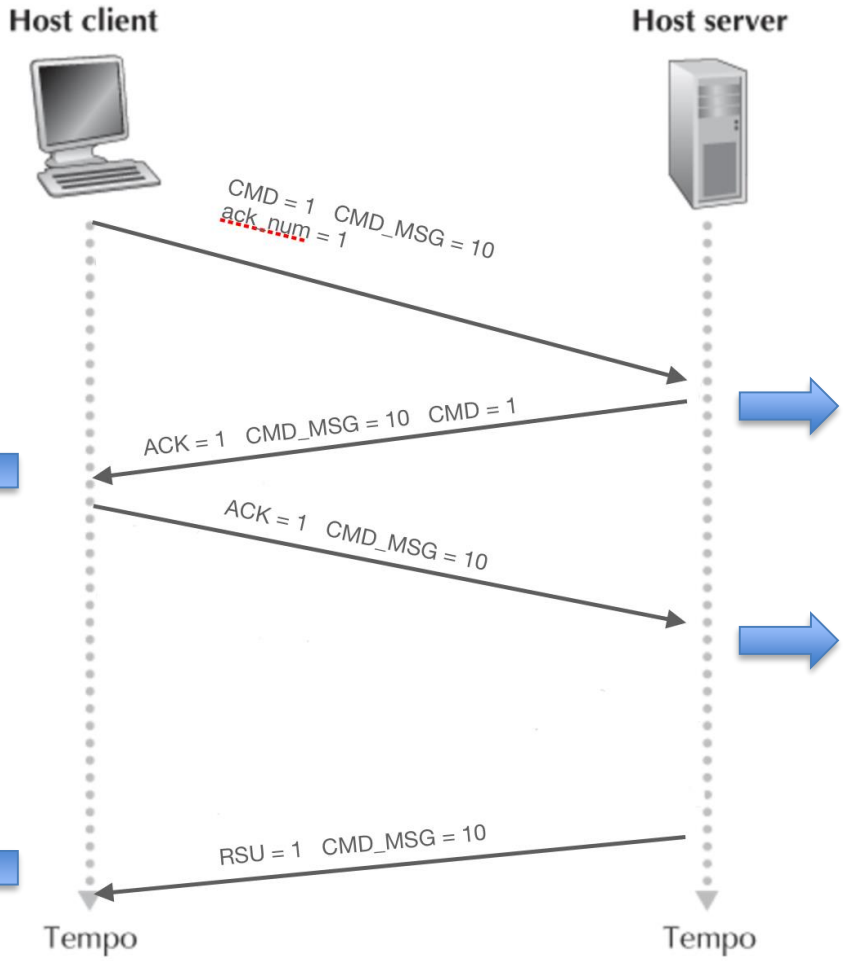
Richiesta comando di GET

- Dopo aver ricevuto dallo standard input il comando get il client invia un pacchetto di richiesta di contenente il nome del file che si vuole ottenere
- In particolare tale pacchetto ha impostati i flag di CMD a 1 e di CMD_MSG a 10 corrispondente appunto al comando get
- mentre il nome del file da scaricare viene inserito all'interno del payload

comando GET

- A differenza del comando put in questo caso lo scambio viene effettuato dopo un handsake a tre vie
- Questo perché il server mittente, cioè colui che conosce l'effettiva dimensione del file deve informare il ricevente, cioè il client, sulla dimensione delle risorse da allocare
- Proprio per questo il server ricevuta la richiesta controlla se il file richiesto è presente nella cartella doc e nel caso quest'ultimo sia presente invia un messaggio di ack al client impostando i flag di ACK a 1 e CMD_MSG a 10

comando GET: implementazione



Allocazione risorse

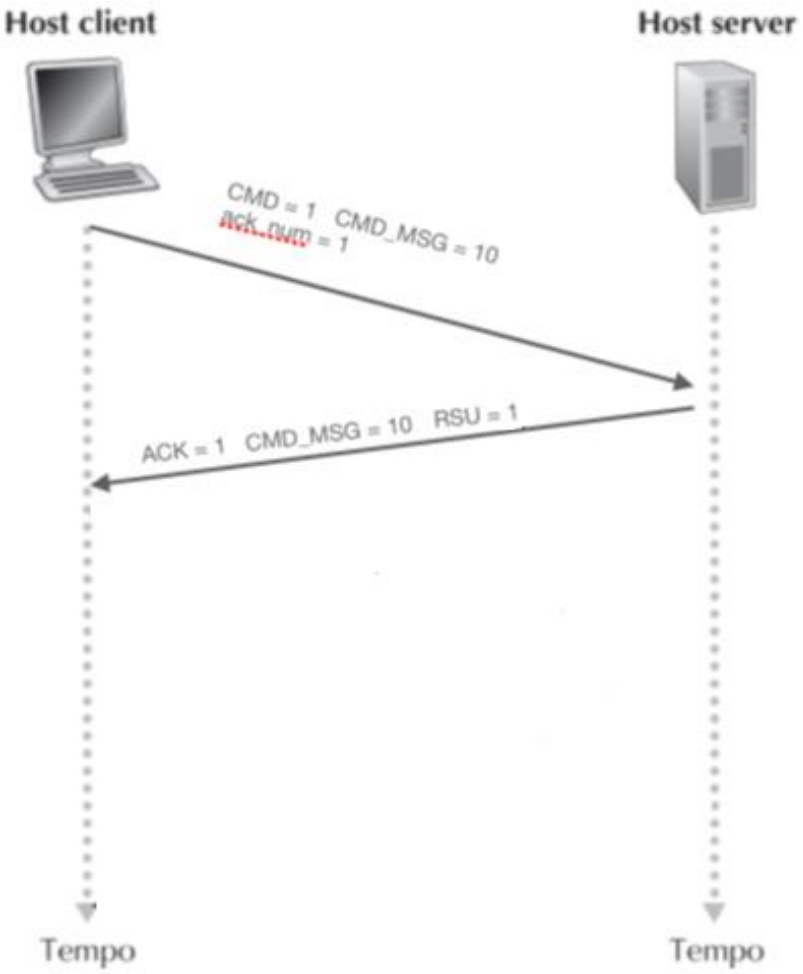
Ricerca del file e allocazione risorse

Inizio scambio file

Ricezione ultimo pacchetto , invia il riscontro al server e termina la get

comando GET: implementazione

Nel caso in cui il file richiesto dal Client non è presente il Server risponde immediatamente con un pacchetto indicante l'errore impostando i flag CMD_MSG, ACK e RSU a 1



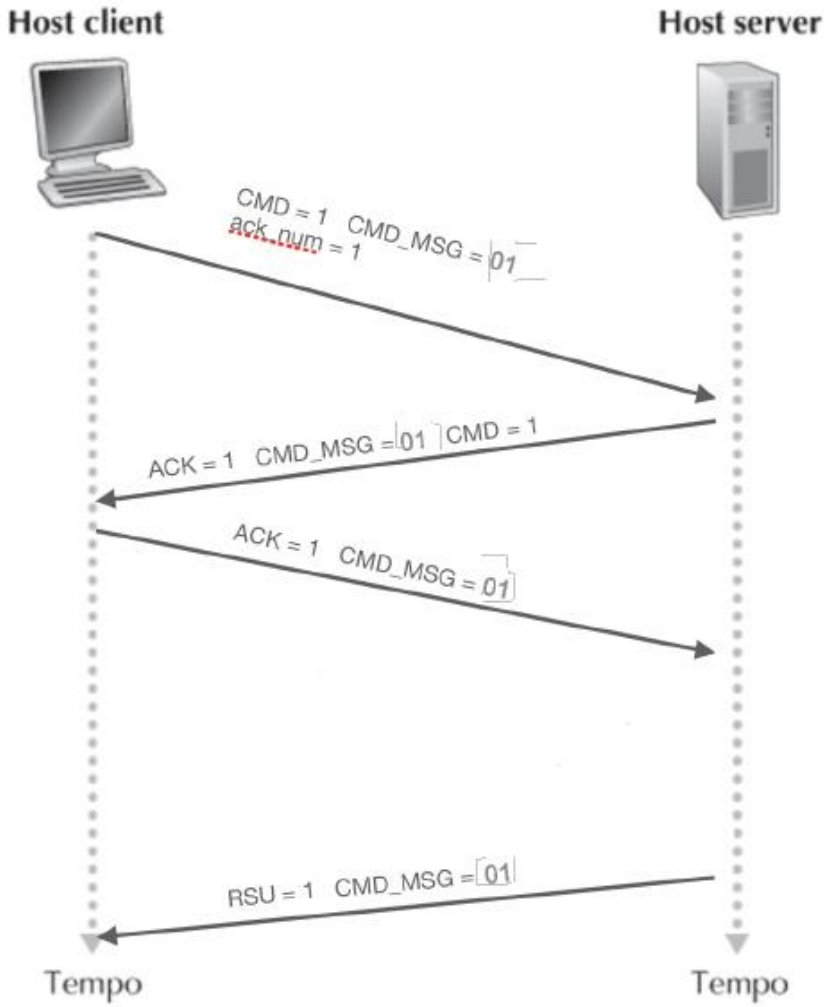
comando LIST

- Il comando di list permette al Client di visualizzare la lista dei file di cui è possibile fare la get
- Digitando il comando list il Client invia un pacchetto con tale richiesta al Server
- In particolare tale pacchetto ha impostati i flag di CMD a 1 e di CMD_MSG a 01 corrispondente appunto al comando list

comando LIST

- Anche nel comando di list si utilizza un handshake a tre vie come nel caso della get
- Infatti le due funzioni sono sostanzialmente analoghe
- L'unica differenza sta nel fatto che in questo caso il file di richiesta è sempre lo stesso, ovvero il file list contenente la lista dei documenti presenti nel Server

comando di LIST: implementazione



Chiusura della connessione

- Nel caso il client voglia terminare la connessione deve digitare da terminale il comando end
- Una volta effettuato tale comando viene inviato un pacchetto di richiesta di chiusura al server
- In particolare tale pacchetto ha impostati i flag del FIN a 1

Chiusura della connessione

- Anche in questo caso prima dell'effettiva chiusura vi è una sorta di handsake
- Infatti il server dopo aver ricevuto il pacchetto di FIN risponde al client con un pacchetto di FIN-ACK e allo stesso tempo attiva un timer per permettere a quest'ultimo di ricevere correttamente l'ack nel caso in cui dovesse perdersi
- Scaduto questo timer il Server invia un ultimo pacchetto di FIN e chiude definitivamente la connessione allo scadere del timer associato a quest'ultimo pacchetto , anche se non dovesse ricevere l'ultimo ack del Client

Chiusura della connessione

- Il Client ricevuto il FIN-ACK aspetta il FIN da parte del Server per poter effettivamente terminare la connessione
- Ricevuto quest'ultimo pacchetto invia il pacchetto di FIN-ACK e avvia una attesa temporizzata
- Infine dopo il periodo di attesa il Client chiude definitivamente la connessione e libera le risorse

Chiusura della connessione