Immagine che contiene testo, screenshot, clipart

Descrizione generata automaticamente

Ingegneria di Internet e Web

Progetto A.A. 2019/2020

UDP-GBN

Massimo Mazzetti

0253467

**INDICE**

[1. Traccia del Progetto 2](#_Toc119632712)

[2. Architettura e scelte progettuali 4](#_Toc119632713)

[3. Implementazione 5](#_Toc119632714)

[4. Limitazioni 6](#_Toc119632715)

[5. Piattaforma utilizzata per sviluppo e testing 7](#_Toc119632716)

[6. Esempi 8](#_Toc119632717)

[7. Valutazione Prestazioni 9](#_Toc119632718)

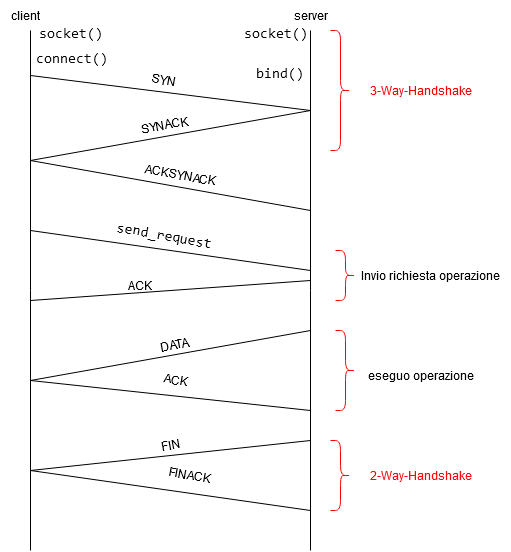
[8. Configurazione, Installazione eD Esecuzione 10](#_Toc119632719)

# Traccia del Progetto

|  |  |
| --- | --- |
|  | **Reti di Calcolatori ed Ingegneria del Web - A.A. 2019/20**  **Progetto B1: Trasferimento file su UDP**  Lo scopo de progetto è quello di progettare ed implementare in linguaggio C usando l’API del socket di Berkeley un’applicazione client-server per il trasferimento di file che impieghi il servizio di rete senza connessione (socket tipo SOCK\_DGRAM, ovvero UDP come protocollo di strato di trasporto). Il software deve permettere:  • Connessione client-server senza autenticazione;  • La visualizzazione sul client dei file disponibili sul server (comando list);  • Il download di un file dal server (comando get);  • L’upload di un file sul server (comando put);  • Il trasferimento file in modo affidabile.  La comunicazione tra client e server deve avvenire tramite un opportuno protocollo. Il protocollo di comunicazione deve prevedere lo scambio di due tipi di messaggi: messaggi di comando: vengono inviati dal client al server per richiedere l’esecuzione delle diverse operazioni; messaggi di risposta: vengono inviati dal server al client in risposta ad un comando con l’esito dell’operazione.  **Funzionalità del server**  Il server, di tipo concorrente, deve fornire le seguenti funzionalità:  • L’invio del messaggio di risposta al comando list al client richiedente; il messaggio di risposta contiene la filelist, ovvero la lista dei nomi dei file disponibili per la condivisione;  • L’invio del messaggio di risposta al comando get contenente il file richiesto, se presente, od un opportuno messaggio di errore;  • La ricezione di un messaggio put contenente il file da caricare sul server e l’invio di un messaggio di risposta con l’esito dell’operazione.  **Funzionalità del client**  I client, di tipo concorrente, deve fornire le seguenti funzionalità:  • L’invio del messaggio list per richiedere la lista dei nomi dei file disponibili;  • L’invio del messaggio get per ottenere un file  • La ricezione di un file richiesta tramite il messaggio di get o la gestione dell’eventuale errore  • L’invio del messaggio put per effettuare l’upload di un file sul server e la ricezione del messaggio di risposta con l’esito dell’operazione.  **Trasmissione affidabile**  Lo scambio di messaggi avviene usando un servizio di comunicazione non affidabile. Al fine di garantire la corretta spedizione/ricezione dei messaggi e dei file sia i client che il server implementano a livello applicativo il protocollo Go-Back N (cfr. Kurose & Ross “Reti di Calcolatori e Internet”, 7° Edizione). Per simulare la perdita dei messaggi in rete (evento alquanto improbabile in una rete locale per non parlare di quando client e server sono eseguiti sullo stesso host), si assume che ogni messaggio sia scartato dal mittente con probabilità p. La dimensione della finestra di spedizione N, la probabilità di perdita dei messaggi p, e la durata del timeout T, sono tre costanti configurabili ed uguali per tutti i processi. Oltre all’uso di un timeout fisso, deve essere possibile scegliere l’uso di un valore per il timeout adattativo calcolato dinamicamente in base alla evoluzione dei ritardi di rete osservati. I client ed il server devono essere eseguiti nello spazio utente senza richiedere privilegi di root. Il server deve essere in ascolto su una porta di default (configurabile). |

# Architettura e scelte progettuali

L'applicazione presenta un'architettura di tipo client-server.



# Implementazione

# Limitazioni

# Piattaforma utilizzata per sviluppo e testing

Lo sviluppo è avvenuto in ambiente Unix in particolare si è utilizzata una **WSL2** (Windows Subsystem for Linux) su cui è installata la distribuzione “**Ubuntu 20.04.4 LTS**”.

La versione del Kernel è “**5.10.60.1-microsoft-standard-WSL2**”.

Per la compilazione è stato utilizzato “**gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0**”.

Per la scrittura del codice è stato utilizzato il text editor “**Visual Studio Code**”.

# Esempi

# Valutazione Prestazioni

|  |  |  |  |
| --- | --- | --- | --- |
| Perdita | Finestra | Timer [ms] | Tempo esecuzione [ms] |
| 0 |  |  |  |
| 0.2 |  |  |  |
| 0.5 |  |  |  |
| 0.8 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Configurazione, Installazione ed Esecuzione