**Reti di Calcolatori ed Ingegneria del Web**

**A.A. 2019/20**

**Progetto B2: Trasferimento file su UDP**

***pp. da 3 a 4 - Introduzione a UDP***

***pp. da 5 a 6 - Presentazione dell’Applicazione e manuale d’uso***

***pp. da 7 a 16 - Funzionamento dell’Applicazione***

***pp. da 17 a 22 - Esempi di File di log e Analisi***

***pp. da 23 a 29 - Test e Risultati***

*1 Struttura generale*

*1.1 Moduli*

*1.2 Implementazione del client concorrente*

*1.3 Implementazione del server concorrente*

*1.4 Implementazione del protocollo Selective Repeat*

*1.5 Parametrizzazione e Log*

*2 Connessione fra client e server e trasferimento dei dati*

*2.1 Schema generale*

*2.2 Trasferimento dei dati*

*3 Modalità di funzionamento dell’applicazione*

*3.1 Dimensione della finestra scorrevole*

*3.2 Perdita di un pacchetto sulla rete*

*3.3 Ritrasmissione di un pacchetto: RTO fisso e adattativo*

*3.4 Simulazione tempo di transito sulla rete*

*3.5 Gestione del tempo di attesa di un pacchetto e numero massimo di ritrasmissioni*

*4 Descrizione delle procedure*

*4.1 Procedure di utilità*

*4.2 Diagrammi di flusso*

*5 File di configurazione*

*6 File di log*

*6.1 File di log prodotto in trasmissione*

*6.2 File di log prodotto in ricezione*

*7 TEST*

*7.1 Test per la verifica della corretta implementazione del protocollo Selective Repeat*

*7.2 Test effettuati per verificare il corretto funzionamento di server e client concorrenti*

*7.3 Test vari*

*MANUALE*

*a. Ambiente di sviluppo e test*

*b. Moduli*

*c. Contenuto delle directory*

*d. Istruzioni per l’esecuzione dei*

*Allegato 1 Esempio file di log completo del processo in trasmissione*

*Allegato 2 Esempio file di log completo del processo in ricezione*

*Allegato 3 Codice in pdf*

*La relazione contiene:*

*• la descrizione dettagliata dell'architettura del sistema e delle scelte progettuali effettuate;*

*• la descrizione dell'implementazione;*

*• la descrizione delle eventuali limitazioni riscontrate;*

*• l'indicazione della piattaforma software usata per lo sviluppo ed il testing del sistema;*

*• alcuni esempi di funzionamento;*

*• valutazione delle prestazioni del protocollo al variare della dimensione della finestra di spedizione N,*

*della probabilità di perdita dei messaggi p, e della durata del timeout T (incluso il caso di T adattativo)*

*• un manuale per l'installazione, la configurazione e l'esecuzione del sistema.*

**Sommario**

**Panoramica UDP**

UDP è un protocollo di trasporto non orientato alla connessione e non affidabile: non garantisce né l’arrivo dei pacchetti a destinazione, né la ritrasmissione in caso di perdita d’informazione, né la corretta sequenza. Il protocollo UDP fornisce soltanto i servizi basilari del livello di trasporto come la consegna di dati da processo a processo e il controllo degli errori. Svolge una funzione di supporto di transazioni semplici tra applicativi e applicazioni realtime che tollerano perdite. Un’applicazione che usa UDP deve risolvere problemi di affidabilità, perdita di pacchetti, duplicazione, controllo di sequenza, controllo di flusso, controllo di congestione.

Un socket UDP viene identificato univocamente dalla coppia: indirizzo IP, porta di destinazione.

**Struttura generale dell’applicazione**

L’applicazione, basata su un’architettura client-server, utilizza UDP come protocollo di trasporto e permette il trasferimento di file in modo affidabile.

L’applicazione è costituita da tre moduli principali: in ***program.h*** sono definite le principali strutture dati, le variabili globali e le procedure utilizzate in ***client.c*** e ***server.c*** che contengono i relativi main, procedure e tipi di dati.

Per il funzionamento dell’applicazione sono richiesti un ***file di configurazione***, con i valori dei parametri di configurazione, e un ***file log***, per registrare le operazioni svolte e analizzare la sequenza delle azioni compiute in trasmissione e ricezione.

**Implementazione**

Nell’applicazione client e server sono di tipo concorrente. La concorrenza è stata implementata tramite la creazione di processi figli che gestiscono la connessione client/ server in base al relativo comando.

Sono stati sviluppati tre comandi principali, con le rispettive funzionalità per client e server:

• ***list*** per la visualizzazione sul client di una specifica directory;

• ***get***: per il download di un file dal server al client;

• ***put***: per l’upload di un file sul server;

Generalmente, il client crea un processo figlio che: ***\**** alloca un’area di memoria per i nomi degli oggetti delle directory che riceverà dal server (comando list); ***\**** apre il file destinazione in scrittura (comando get); ***\**** apre il file origine in lettura (comando put).

Il server riceve il messaggio CMD dal client e crea un processo figlio che rispettivamente: ***\**** alloca un’area di memoria per i nomi degli oggetti contenuti nella directory; ***\**** apre il file origine in lettura; ***\**** apre il file destinazione in scrittura. Il server invia quindi un messaggio del tipo ACK al client.

Per l’istruzione ***list***: il client risponde all’ACK con un messaggio START e si predispone alla ricezione dei dati; il server inizia il trasferimento dei dati e quando il client riceve l’ultimo pacchetto, invia il messaggio END al server.

Per l’istruzione ***get***: il client risponde con un messaggio START e si predispone alla ricezione dei dati; il server inizia il trasferimento dei dati e quando il client riceve l’ultimo pacchetto, invia il messaggio END al server.

Per l’istruzione ***put***: il server è disponibile alla ricezione; quando il client riceve il messaggio ACK, inizia il trasferimento dei dati; una volta ricevuto l’ultimo pacchetto, il server invia il messaggio END al client.

**Parametri principali**

**Manuale d’uso**

**Test**