|  |  |
| --- | --- |
|  | Allievo/i: Delvai Lukas  Data: 05/05/2025 Classe:4° IB |
| **LABORATORIO DI SISTEMI E RETI**  **Titolo progetto**: Client e Server con i Socket | |

**Scopo del progetto:** realizzazione di un’applicazione già che adoperi i socket per la comunicazione tra Client e Server

**Istruzione per l’esecuzione:** per eseguire il codice prima dovrà essere compilato dall’ambiente di sviluppo che si usa una volta fatto questo sarà necessario utilizzo del terminale e dovremmo immettere questo comando telnet 127.0.0.1 1050 o al posto del terminale si può utilizzare Teraterm.

Una volta entrati si può scegliere che ricerca fare servirà per questo:

* Una ricerca campo come ad esempio comune, provincia.
* Una ricerca tipo: tutti gli alberghi che si trovano nel comune di Angera.

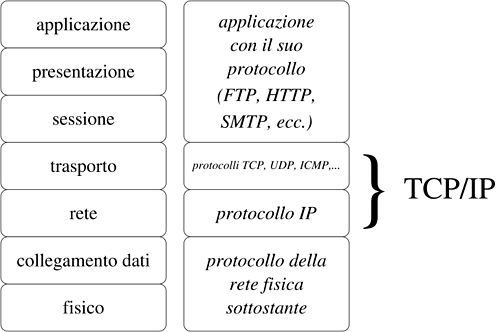
**DESCRIZIONE TEORICA**

**LAYER DI TRASPORTO**

Il layer di trasporto si occupa della comunicazione end-to-end e della trasmissione affidabile di informazioni in tutto questo non e a conoscenza del tipo di host destinazione, sul tipo di supporto nel quale viaggeranno i dati o che percorso intraprenderanno i dati:

Questo layer include due diversi tipi di protocolli:

* Transmission Control Protocol (TCP)
* User Datagram Protocol(UDP)



**TRANSMISSION CONTROL PROTOCOL (TCP)**

Il protocollo TCP viene considerato un protocollo del layer di trasporto affidabile e completo, assicurando che tutti i dati arrivano a destinazione questo avviene perché il TCP include dei campi che assicurano la consegna dei dati e divide i dati in segmenti

TCP fornisce affidabilità e controllo del flusso utilizzando queste operazioni di base:

* Numera e tiene traccia dei segmenti di dati trasmessi a un host specifico da un'applicazione specifica
* Riconoscere i dati ricevuti
* Trasmettere nuovamente i dati non riconosciuti dopo un lasso di tempo
* Sequenziare i dati che potrebbe arrivare nell'ordine sbagliato
* Trasmette i dati a una velocità tale che e anche accettabile dal destinatario

**HEADER TCP**

E un protocollo stateful il che significa che tine traccia dello stato della sessione di comunicazione siccome tiene la traccia dello stato di una sessione dovrà registrare le informazioni che ha invitato.

Header TCP e lungo 20 byte

Immagine che contiene testo, schermata, Carattere, numero

Il contenuto generato dall'IA potrebbe non essere corretto.

È utilizzato soprattutto nei casi: FTP, HTTP, SMTP, DNS e SSH

**USER DATAGRAM PROTOCOL (UDP)**

Questo protocollo e più semplice del TCP siccome non fornisce affidabilità e controllo del flusso e definito come un protocollo:

* **Connectionless** vuol dire che non richiede una connessione stabilita
* **Stateless** non tengono traccia di alcuna informazione sui pacchetti inviati
* **Best-effort** poiché non viene confermato che i dati vengano ricevuti dal dispositivo di destinazione

Le funzionalità che ha il protocollo UDP:

* I dati vengono ricostruiti nell'ordine in cui vengono ricevuti.
* Eventuali segmenti persi non vengono inviati nuovamente.
* Non è prevista la creazione di sessioni.
* L'invio non viene informato sulla disponibilità delle risorse.

**HEADER UDP**

Siccome il protocollo UDP non e affidabile come il protocollo TCP contiene meno campi nel header, infatti, ne dispone solamente di 4 campi.

Immagine che contiene testo, schermata, Carattere, linea

Il contenuto generato dall'IA potrebbe non essere corretto.

E utilizzato soprattutto nei casi: DNS, DHCP, SNMP, VoIP, Video conferenza e TFTP

Attenzione anche il TPC utilizza il DNS ma solo se la risposta e superiore a 512 byte

**SOCKET**

Nella rete ci possono tipologie di dispositivi di diverso tipo, tra cui possiamo notare i dispositivi Client che possono essere una macchina o un processo che richiedono un servizio attraverso un protocollo e invece i Server hanno lo scopo di fornire un servizio ai client.

La comunicazione che addoperanno il modello client/server e supportata attraverso i socket che sono da intendersi è come una porta che permette a due programmi di parlarsi, di solito il socket gli viene associato un indirizzo e una porta:

* Un indirizzo IP consente di invidiare in maniera univoca la posizione di un dispositivo all’interno di una rete.

È costituito da 32 bit suddivisi in 4 ottetti ognuno contenenti 8 bit e può contenere valori da 0 a 255

* I numeri di porta vengono utilizzati sia dal TCP e dal UDP vengono utilizzati per gestire più conversazioni in maniera simultanea in entrambi gli header ci sono campi di porta di origine e porta di destinazione.

Il layer di trasporto tiene traccia della porta così quando riceverà una risposta saprà inoltrala all’ applicazione corretta, IANA ha suddiviso in tre gruppi di porte

|  |  |  |
| --- | --- | --- |
| **Gruppi porta** | **Intervallo porte** | **Descrizione** |
| Porte conosciute | da 0 a 1023 | Sono assegnate a servizi comuni o nelle applicazioni come quelle web o di posta elettronica |
| Porte registrate | da 1024 a 49.151 | Vengono utilizzate da software e protocollo comuni per la comunicazione su rete |
| Porte private/dinamiche | da 49.152 a 65.535 | Note anche come porte temporanee, il client OS assegna i numeri di porta quando viene iniziata una connessione a un servizio |

Esistono due diverse modalità di connessione che può avvenire tra il client e il server:

* **Connection oriented**: il TCP/IP stabilisce un canale di comunicazione che rimane attivo fino alla fine della trasmissione o finche il canale non viene chiuso da uno dei due dispositivi.

Si predilige questa comunicazione per garantire la trasmissione dei pacchetti e la loro integrità tramite alcuni messaggi(acknowledge) questo comporta la divisione del pacchetto dei dati in frametti/datagrammi.

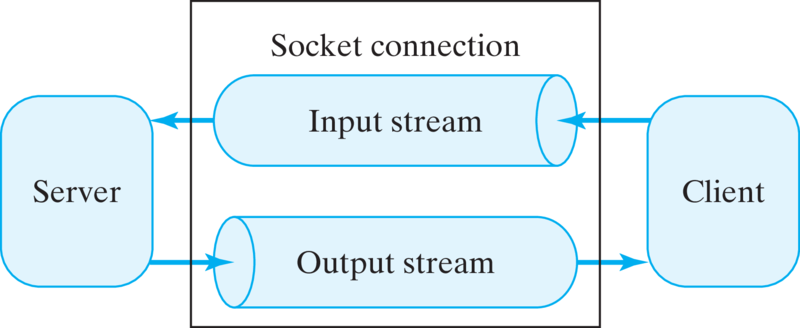
Tuttavia, con questa tecnica si sovraccarica la rete.

* **Connectionless**: e utilizzato dall’UDP non niente conto dell’integrità dei pacchetti e né della avvenuta trasmissione che svolge il client.

Questo ne consegue un minimo sovraccarico della rete.

Esistono anche due tipi di socket:

* **Strem socket**: lo stream e possibile immaginarlo come un canale dalla sua origine e la sua destinazione in maniera del tutto indipendentemente dall’origine un programma utilizza un input stream che legge i dati e un output stream per scrivere i dati.



Come si può notare dal ‘immagine il client e il server creano una connessione che addoperà due flussi uno per input e altro per l’output.

Per il lato client e una connessione che invia e che ne ricevi i dati bisogna usare la classe Java Socket invece dal il lato Server attende una connessione dal client e la classe ServerSocket

* **Datagram socket**: trasferiscono messaggi di lunghezza variabile ma senza che garantisce un ordine di arrivo dei pacchetti e non creano una connessione tra il client e server ed e supportato dal protocollo UDP.

La classe Java che usa questo protocollo è DatagramSocket. I dati sono impacchettati tramite la classe DatagramPacket.

**LATO CLIENT**

Il client crea un oggetto di tipo socket, iniziando una connessione con il server, il server deve essere già in esecuzione e pronto per ricevere questa connessione:

* Socket s = new Socket(indirizzo, porta);
* Indirizzo si intende quello del server e la porta permetti di identificare la comunicazione giusta in questo caso utilizziamo la porta 1050 dell’esercizio

Se la creazione dello stream socket ha successo, viene prodotta una connessione bidirezionale tra i due processi.

Dopo la connessione, il client crea uno stream di output con cui invierà i dati al server:

* OutputStreamWriter osw = new OutputStreamWriter(s.getOutputStream());
* BufferedWriter bw = new BufferedWriter(osw);
* PrintWriter out = new PrintWriter(bw, true);

la classe OutputStreamWriter serve a convertire un flusso di caratteri in un flusso di byte invece si e utilizzato l’oggetto bw serve a migliorare l’efficienza utilizzando un buffer di caratteri.

* Per ricevere dati dal server bisogna definire uno stream per l’input:
* InputStreamReader isr = new InputStreamReader(s.getInputStream());
* BufferedReader in = new BufferedReader(isr);

Per leggere una linea dallo stream di input del socket:

* String s = in.readLine();

Quando la connessione non è più necessaria il client deve liberare la risorsa. Per fare questo chiude lo stream di output, lo stream di input e il socket:

* out.close();
* in.close();
* s.close();

Per chiudere la connessione tra il client e il server e il metodo close ()

Altri metodi della classe Socket sono:

* InetAddress getInetAddress(): restituisce l’indirizzo IP del server
* InetAddress getLocalAddress(): restituisce l’indirizzo locale
* int getPort(): restituisce la porta remota
* int getLocalPort(): restituisce la porta locale

**CODICE**

**MAIN**

Immagine che contiene testo, schermata, software

Il contenuto generato dall'IA potrebbe non essere corretto.

* In alto importiamo le librerie che ci servano
* Definiamo una costante PORT che e la porta in cui avere la comunicazione tra server e client
* Il while permette al server di stare in esecuzione per potere accettare nuove connessioni
* Poi si crea l’istanza ServerSocket nella porta 1050
* Stampa informazioni relative al Server
* Con la chiamta accept() attende un client
* Stampa informazione relative al Client
* Siccome il Server e multithread si crea un oggetto thread con il parametro clientSocket
* E si gestiscono le varie eccezioni

**CLASSE: STRUTTURERICETTIVE**

Immagine che contiene testo, schermata, menu

Il contenuto generato dall'IA potrebbe non essere corretto.

* Aggiungo le librerie che mi servono
* Istanzio le variabili che utilizzerò

Con i relativi getters/setters e con il toString

**Immagine che contiene testo, schermata, software

Il contenuto generato dall'IA potrebbe non essere corretto.**

Questo metodo permette di leggere i file cvs gli viene passato il parametro String filepath che è il percorso in cui e posizionato il file

* Arraylist serve per contare i dati del cvs
* BufferReader e FileReader servono per la lettura del file
* Il while continua finché ci sono righe
* String [] values separa i caratteri invece -1 e per i campi vuoti
* Queste istruzioni assegnano i valori letti dal file cvs agli attributi dell'oggetto StruttureRicettive
* E ritorno data

**CLASSE: THREADSERVER**

Immagine che contiene testo, schermata, software

Il contenuto generato dall'IA potrebbe non essere corretto.

* Importo le librerie che mi serviranno in seguito
* Estendo la classe ThreadServer
* -Istanzio le variabili che usufruirò
* - E nel costruttore inizializzo il clientsocket,struttureRecittive e data con il metodo che permette la lettura del file cvs

Immagine che contiene testo, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

* InputStreamReader,BufferReader,OutputStreamWriter,BufferWriter e PrintWriter sono lo strem di input e di output
* Avvengono delle stampe dove mostrane le ricerche che puoi fare
* Successivamente ti chiede di inserire un campo e il tipo

Immagine che contiene testo, schermata, software, Software multimediale

Il contenuto generato dall'IA potrebbe non essere corretto.

* Chiamo il metodo serach
* Nel while legge la Stringa e se e STOP o END chiude tutto
* Gestisco l’eccezione

Immagine che contiene testo, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

* Ogni volta elimina i risultati precedenti
* Con un for each percorre tutte le strutture che sono presenti nella Arraylist data
* Lo switch in base a searchCampo ti indirizza a una ricerca e ti darà tutti i risultati con quella ricerca
* Infine if controlla se la results e vuoto se contiene dati li stampa