RELAZIONE TECNICA

OGETTO

socket UDP

ALUNNO

Topcii Daniil

DOCENTI

Crafa Nicola

Conte Alberto

SOCKET UDP

Per la creazione di un socket udp è stato proposto come esercizio la creazione di un server che accettasse pacchetti UDP dai client e rispondesse con l'orario attuale del server, per svolgere l'esercitazione si sono creati due file:

SERVER e CLIENT.

SERVER:

- Main, qui vengono svolete le funzioni principali come:
  - la carica dei contatori,
  - recezione dei pacchetti UDP,
  - estrazione dell'indirizzo ip del client,
  - conta degli utilizzi per client
  - invio del pacchetto UDP con l orario

CODICE:

```
public static void main(String[] args) {
        // Crea il socket del server
        DatagramSocket serverSocket = null;
        try {
            serverSocket = new DatagramSocket(PORT);

            // Carica i contatori di utilizzo dal file
            Map<InetAddress, Integer> usageCount = loadUsageCount();

            while (true) {
                // Crea il buffer per ricevere il pacchetto UDP
                byte[] receiveBuffer = new byte[BUFFER_SIZE];
                DatagramPacket receivePacket = new
DatagramPacket(receiveBuffer, receiveBuffer.length);

                // Ricevi il pacchetto UDP dal client
                serverSocket.receive(receivePacket);

                // Estrai l'indirizzo IP e il numero di porta del client
                InetAddress clientAddress = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();

                // Incrementa il contatore di utilizzo per il client
                int count = usageCount.getOrDefault(clientAddress, 0) + 1;
                usageCount.put(clientAddress, count);
```

```
                // Se il client ha superato il limite di utilizzo gratuito,
                // invia un messaggio per avvertire che il servizio è a
pagamento
                String response;
                if (count > FREE_SERVICE_LIMIT) {
                    response = "Servizio a pagamento";
                } else {
                    // Invia la data e l'ora correnti al client
                    Date now = new Date();
                    response = now.toString();
                }

                // Crea il buffer per inviare il pacchetto UDP
                byte[] sendBuffer = response.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length, clientAddress, clientPort);

                // Invia il pacchetto UDP al client
                serverSocket.send(sendPacket);

                // Salva i contatori di utilizzo nel file
                saveUsageCount(usageCount);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            // Chiudi il socket del server se è stato aperto
            if (serverSocket != null) {
                serverSocket.close();
            }
        }
    }
```

- loadUsageCount
  - controllo dell'esistenza del file archivio in caso contrario la sua creazione
  - lettura del file

CODICE:

```
private static Map<InetAddress, Integer> loadUsageCount() throws IOException
{
        Map<InetAddress, Integer> usageCount = new HashMap<>();
        Path path = Paths.get(USAGE_COUNT_FILE);
        if (Files.exists(path)) {
            for (String line : Files.readAllLines(path)) {
```

```
                String[] parts = line.split(",");
                InetAddress address = InetAddress.getByName(parts[0]);
                int count = Integer.parseInt(parts[1]);
                usageCount.put(address, count);
            }
        } else {
            // Crea il file dei contatori di utilizzo se non esiste
            Files.createFile(path);
        }
        return usageCount;
    }
```

- saveUsageCount
  - scrittura nel file

CODICE:

```
private static void saveUsageCount(Map<InetAddress, Integer> usageCount)
throws IOException {
        Path path = Paths.get(USAGE_COUNT_FILE);
        List<String> lines = new ArrayList<>();
        for (Map.Entry<InetAddress, Integer> entry : usageCount.entrySet()) {
            InetAddress address = entry.getKey();
            int count = entry.getValue();
            lines.add(address.getHostAddress() + "," + count);
        }
        Files.write(path, lines, StandardOpenOption.CREATE,
StandardOpenOption.TRUNCATE_EXISTING);
    }
```

CLIENT:

- main che svolge le seguenti funzioni:
  - creazione del socket
  - invio del pacchetto UDP
  - ricezione del pacchetto UDP
  - estrazione del contenuto del pachetto
  - stampa a video del risultato

CODICE:

```
public static void main(String[] args) {
        DatagramSocket clientSocket = null;
        try {
            // Crea il socket del client
            clientSocket = new DatagramSocket();

            // Invia un pacchetto UDP vuoto al server
            byte[] sendBuffer = new byte[0];
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length,
                    InetAddress.getLocalHost(), PORT);
            clientSocket.send(sendPacket);

            // Crea il buffer per ricevere il pacchetto UDP dal server
            byte[] receiveBuffer = new byte[BUFFER_SIZE];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);

            // Ricevi il pacchetto UDP dal server
            clientSocket.receive(receivePacket);

            // Estrai il messaggio dal pacchetto UDP
            String message = new String(receivePacket.getData(), 0,
receivePacket.getLength());

            // Stampa il messaggio ricevuto dal server
            System.out.println(message);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (clientSocket != null) {
                clientSocket.close();
            }
        }
    }
```

Come richiesto dalla traccia dopo la decima connessione non verrà inviata la data ma soltanto che il servizio è diventato a pagamento.