Examen Modelo – Programación Distribuida

Fecha: 6/11/2025

Examen

1. Sockets (20 puntos)

Implemente en C un sistema cliente–servidor denominado `PingPong`. **Servidor:** escucha en el puerto `5000` y responde con el mensaje `"pong"` cada vez que recibe `"ping"`. **Cliente:** se conecta al servidor, envía el mensaje `"ping"` y muestra la respuesta recibida. Utilice las llamadas estándar de Berkeley Sockets (`socket()`, `bind()`, `listen()`, `accept()`, `connect()`, `read()`, `write()`, `close()`) y describa brevemente el flujo de ejecución.

2. gRPC (40 puntos)

Diseñe un archivo `.proto` (versión 3) que defina un servicio `CalculatorService` con los siguientes métodos:

- **Add:** recibe dos números y devuelve su suma.
- **SumStream:** recibe una secuencia de números y devuelve la suma total al finalizar.

Incluya la línea `syntax = "proto3";`, defina los mensajes necesarios (*requests* y *responses*) e indique el tipo de RPC correspondiente a cada método. No escriba código del cliente ni del servidor.

3. JADE (40 puntos)

Programe un agente móvil en Java denominado `SumadorAgente` que calcule la suma de todos los números almacenados en un archivo de texto ubicado en una computadora remota. El agente debe:

1. Migrar desde el contenedor de origen al contenedor remoto indicado.
2. Leer el archivo localmente en el host remoto.
3. Volver al contenedor de origen y mostrar la suma total.

Condiciones: Java 17 y JADE 4.5 ya importados. Plataformas conectadas: `Main-Container` (host A) y `Remoto-1` (host B). Archivo remoto: `/home/alumno/data/nums.txt`. El agente se lanza con tres argumentos: `hostDestino`, `containerDestino` y `rutaRelativa`. Suponga happy path (sin errores de red ni permisos).

Anexo A — Funciones de Sockets

A.1 Funciones típicas en C (BSD Sockets)

`socket(domain, type, protocol)`	Crea un descriptor de socket (por ejemplo, `AF_INET`, `SOCK_STREAM`).
`bind(sockfd, addr, addrlen)`	Asocia el socket a una dirección IP y puerto locales.
`listen(sockfd, backlog)`	Pone el socket en modo escucha, habilitándolo para aceptar conexiones.
`accept(sockfd, addr, addrlen)`	Acepta una conexión entrante y devuelve un nuevo descriptor para el cliente.
`connect(sockfd, addr, addrlen)`	Solicita conexión a un servidor.
`read(sockfd, buf, nbytes)`	Lee datos del socket (bloqueante si no hay datos).
`write(sockfd, buf, nbytes)`	Envía datos a través del socket.
`close(sockfd)`	Cierra el descriptor del socket.
`htons/htonl, ntohs/ntohl`	Conversión hostred (endianess) para puertos y direcciones.
`inet_pton / inet_ntop`	Conversión entre formato texto y binario de direcciones IP.
`gethostbyname(), getaddrinfo()`	Resolución de nombres de host a direcciones IP.
`setsockopt(), getsockopt()`	Configura o consulta opciones del socket (por ejemplo, `SO_REUSEADDR`).
`shutdown(sockfd, how)`	Cierra parcialmente el canal (solo lectura, solo escritura o ambos).

A.2 Clases y métodos en Java (Sockets TCP)

`ServerSocket(int port)`	Crea un socket servidor que escucha en el puerto indicado.
`accept()`	Espera (bloqueante) una conexión entrante y devuelve un objeto `Socket`.
`Socket(String host, int port)`	Crea un socket cliente conectado al servidor indicado.
`getInputStream()`	Devuelve el flujo de entrada (lectura) del socket.
`getOutputStream()`	Devuelve el flujo de salida (escritura) del socket.
`InputStream.read(byte[])`	Lee datos del socket (bloqueante).
`OutputStream.write(byte[])`	Envía datos a través del socket.
`close()`	Cierra la conexión y libera los recursos.

Resolución Modelo

1) Sockets en C — Ping/Pong con `read`/`write`

Servidor:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <stdio.h>

int main() {
  int sockfd = socket(AF_INET, SOCK_STREAM, 0);

  struct sockaddr_in serv;
  memset(&serv, 0, sizeof(serv));
  serv.sin_family = AF_INET;
  serv.sin_addr.s_addr = INADDR_ANY;
  serv.sin_port = htons(5000);

  bind(sockfd, (struct sockaddr*)&serv, sizeof(serv));
  listen(sockfd, 5);

  int cfd = accept(sockfd, NULL, NULL);
  char buf[256]; memset(buf, 0, sizeof(buf));
  int n = read(cfd, buf, sizeof(buf)-1);
  if (n > 0 && strcmp(buf, "ping") == 0) {
      write(cfd, "pong", 4);
  }

  close(cfd);
  close(sockfd);
  return 0;
}
```

Cliente:

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>
```

```
int main() {
  int sockfd = socket(AF_INET, SOCK_STREAM, 0);
  struct sockaddr_in serv;
  memset(&serv, 0, sizeof(serv));
  serv.sin_family = AF_INET;
  serv.sin_port = htons(5000);
  inet_pton(AF_INET, "127.0.0.1", &serv.sin_addr);

  connect(sockfd, (struct sockaddr*)&serv, sizeof(serv));

  write(sockfd, "ping", 4);
  char buf[256]; memset(buf, 0, sizeof(buf));
  read(sockfd, buf, sizeof(buf)-1);
  printf("Respuesta: %s\n", buf);

  close(sockfd);
  return 0;
}
```

2) gRPC — Archivo `calculator.proto`

```
syntax = "proto3";

package ar.edu.unlp.tp2.grpc;

option java_multiple_files = true;
option java_package = "ar.edu.unlp.tp2.grpc";
option go_package = "tp2/grpc;grpc";

// ======== Mensajes ========

message AddRequest { int64 a = 1; int64 b = 2; }
message AddResponse { int64 result = 1; }
message Number      { int64 value = 1; }
message SumResponse { int64 total = 1; }

// ======== Servicio ========

service CalculatorService {
  rpc Add(AddRequest) returns (AddResponse);          // Unary
  rpc SumStream(stream Number) returns (SumResponse); // Client-streaming
}
```

3) JADE — Agente `SumadorAgente`

```
import jade.core.Agent;
import jade.core.Location;
import jade.core.ContainerID;
import java.io.BufferedReader;
import java.io.FileReader;

public class SumadorAgente extends Agent {
    private String hostDestino;
    private String containerDestino;
    private String rutaRelativa;
    private Location origen;
    private long suma = 0L;

    @Override
    protected void setup() {
        Object[] args = getArguments();
        hostDestino = args[0].toString();
        containerDestino = args[1].toString();
        rutaRelativa = args[2].toString();
        origen = here();

        ContainerID destino = new ContainerID(containerDestino, hostDestino);
        doMove(destino);
    }

    @Override
    protected void afterMove() {
        if (!here().getID().equals(origen.getID())) {
            String ruta = "/home/alumno/data/" + rutaRelativa;
            FileReader fr = null;
            BufferedReader br = null;
            try {
                fr = new FileReader(ruta);
                br = new BufferedReader(fr);
                String line;
                while ((line = br.readLine()) != null) {
                    line = line.trim();
                    if (!line.isEmpty()) {
                        suma += Long.parseLong(line);
                    }
                }
            } catch (Exception e) {
                System.out.println("Error leyendo archivo.");
            } finally {
                try { if (br != null) br.close(); } catch (Exception ignore) {}
                try { if (fr != null) fr.close(); } catch (Exception ignore) {}
```

```
            }
            doMove(new ContainerID(origen.getName(), null));
        } else {
            System.out.println("Suma total = " + suma);
            doDelete();
        }
    }
}
```