



**ACREDITACIÓN INSTITUCIONAL**  
*Avanzamos... ¡Es nuestro objetivo!*



## **Laboratorio Redes Implementación De Un Servicio TCP Para Cálculo De IMC Usando VB De Ubuntu Y Debian**

Cristhian David Parra Parada Código: 1232393708

Jeyson Javier Varela Suarez Código: 1002708720

Facultad de ingenierías y Arquitectura

Redes Grupo B

Docente Luz Marina Santos Jaimes

27 de octubre de 2025



SC-CER96940



*"Formando líderes para la construcción de un nuevo país en paz"*

Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750



**ACREDITACIÓN INSTITUCIONAL**  
*Avanzamos... ¡Es nuestro objetivo!*



## Laboratorio Redes Implementación De Un Servicio TCP Para Cálculo De IMC Usando VB De Ubuntu Y Debian

Como primer paso, debemos de actualizar la maquina virtual del servidor(Ubuntu) para ello, cambiado a administrador con **sudo su** y posteriormente actualizamos conectamos:

**sudo apt update**

```
[~ user] file ...
cristhian@cristhian-VirtualBox:~$ sudo su
[sudo] contraseña para cristhian:
root@cristhian-VirtualBox:/home/cristhian# sudo apt update
Obj:1 http://co.archive.ubuntu.com/ubuntu noble InRelease
Des:2 http://co.archive.ubuntu.com/ubuntu noble-updates InRe
Des:3 http://security.ubuntu.com/ubuntu noble-security InRe
Des:4 http://co.archive.ubuntu.com/ubuntu noble-backports Ir
Des:5 http://co.archive.ubuntu.com/ubuntu noble-updates/mair
41 kB]
Des:6 http://co.archive.ubuntu.com/ubuntu noble-updates/mair
kB]
Des:7 http://co.archive.ubuntu.com/ubuntu noble-updates/mair
75 kB]
Des:8 http://co.archive.ubuntu.com/ubuntu noble-updates/mair
a [15,4 kB]
Des:9 http://co.archive.ubuntu.com/ubuntu noble-updates/rest
```

Como lenguaje de programación se usara java, para ello se tiene que descargar el software de Java Development Kit (JDK) de la versión 21, usamos lo siguiente:

**sudo apt install openjdk-21-jdk -y**

```
root@cristhian-VirtualBox:/home/cristhian# sudo apt install openjdk-21-jdk -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya n
son necesarios.
 libgl1-amber-dri libglapi-mesa libllvm19
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
 ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java
 libatk-wrapper-java-jni libice-dev libpthread-stubs0-dev libsm-dev
 libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
 openjdk-21-jdk-headless openjdk-21-jre openjdk-21-jre-headless x11proto-dev
```



SC-CER96940



*"Formando líderes para la construcción de un nuevo país en paz"*

Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750



## ACREDITACIÓN INSTITUCIONAL

*Avanzamos... ¡Es nuestro objetivo!*



Se hace lo mismo con la otra maquina virtual (cliente -> debían). En Debian para acceder en modo administrador se usa el comando:

**SU -**

```
de12bi34an@vbox:~$ su -
Password:
root@vbox:~#
```

Y actualizamos el sistema => **sudo apt update**

```
root@vbox:~# sudo apt update
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://security.debian.org/debian-security trixie-security InRelease
Hit:3 http://deb.debian.org/debian trixie-updates InRelease
Get:4 http://repo.mysql.com/apt/debian trixie InRelease [22.6 kB]
Err:4 http://repo.mysql.com/apt/debian trixie InRelease
  Sub-process /usr/bin/sq returned an error code (1), error message:
  key on BCA43417C3B485DD128EC6D4B7B3B788A8D3785C is bad:
```

Instalamos Java JDK => **sudo apt install openjdk-21-jdk -y**

```
root@vbox:~# sudo apt install openjdk-21-jdk -y
Installing:
  openjdk-21-jdk

Installing dependencies:
  ca-certificates-java  libsm-dev  openjdk-21-jdk-headless
  java-common           libx11-dev openjdk-21-jre
  libatk-wrapper-java  libxau-dev openjdk-21-jre-headless
  libatk-wrapper-java-jni libxcb1-dev rpcsvc-proto
  libc-dev-bin          libxdmcp-dev uuid-dev
  libc6-dev             libxt-dev  x11proto-dev
  libcrypt-dev          linux-libc-dev xorg-sgml-doctools
  libice-dev            manpages-dev xtrans-dev

Suggested packages:
  default-jre  libsm-doc  openjdk-21-demo  fonts-ipafont-mincho
  libc-devtools libx11-doc  openjdk-21-source fonts-wqy-microhei
  glibc-doc    libxcb-doc visualvm         fonts-wqy-zenhei
```



SC-CER96940



*"Formando líderes para la construcción de un nuevo país en paz"*

Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750



## ACREDITACIÓN INSTITUCIONAL

*Avanzamos... ¡Es nuestro objetivo!*



Ahora en ambas maquinas miramos su ip con => **ip a**

En Ubuntu

```
root@cristhian-VirtualBox:/home/cristhian# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    link/ether 08:00:27:df:c9:c1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.128.9/24 brd 192.168.128.255 scope global dynamic noprefixroute
        valid_lft 82740sec preferred_lft 82740sec
```

En debían

```
root@vbox:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
    link/ether 08:00:27:79:82:a0 brd ff:ff:ff:ff:ff:ff
    altname enx0800277982a0
    inet 192.168.128.8/24 brd 192.168.128.255 scope global dynamic noprefixroute
        valid_lft forever preferred_lft forever
```

Seguidamente se hace ping de maquina a maquina para validar su comunicación

Desde Ubuntu a Debian

```
root@cristhian-VirtualBox:/home/cristhian# ping 192.168.128.8
PING 192.168.128.8 (192.168.128.8) 56(84) bytes of data:
64 bytes from 192.168.128.8: icmp_seq=1 ttl=64 time=2.40 ms
64 bytes from 192.168.128.8: icmp_seq=2 ttl=64 time=1.14 ms
64 bytes from 192.168.128.8: icmp_seq=3 ttl=64 time=3.83 ms
64 bytes from 192.168.128.8: icmp_seq=4 ttl=64 time=4.36 ms
64 bytes from 192.168.128.8: icmp_seq=5 ttl=64 time=6.81 ms
64 bytes from 192.168.128.8: icmp_seq=6 ttl=64 time=4.78 ms
64 bytes from 192.168.128.8: icmp_seq=7 ttl=64 time=4.79 ms
64 bytes from 192.168.128.8: icmp_seq=8 ttl=64 time=3.24 ms
64 bytes from 192.168.128.8: icmp_seq=9 ttl=64 time=5.31 ms
^C
--- 192.168.128.8 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 12261ms
rtt min/avg/max/mdev = 1.136/4.072/6.811/1.574 ms
root@cristhian-VirtualBox:/home/cristhian#
```



SC-CER96940



*"Formando líderes para la construcción de un nuevo país en paz"*

Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750



**ACREDITACIÓN INSTITUCIONAL**  
*Avanzamos... ¡Es nuestro objetivo!*



## Desde Debian a Ubuntu

```
root@vbox:~# ping 192.168.128.9
PING 192.168.128.9 (192.168.128.9) 56(84) bytes of data.
64 bytes from 192.168.128.9: icmp_seq=1 ttl=64 time=3.59 ms
64 bytes from 192.168.128.9: icmp_seq=2 ttl=64 time=4.30 ms
64 bytes from 192.168.128.9: icmp_seq=3 ttl=64 time=2.46 ms
64 bytes from 192.168.128.9: icmp_seq=4 ttl=64 time=2.97 ms
64 bytes from 192.168.128.9: icmp_seq=5 ttl=64 time=3.43 ms
64 bytes from 192.168.128.9: icmp_seq=6 ttl=64 time=3.58 ms
64 bytes from 192.168.128.9: icmp_seq=7 ttl=64 time=4.89 ms
64 bytes from 192.168.128.9: icmp_seq=8 ttl=64 time=4.14 ms
64 bytes from 192.168.128.9: icmp_seq=9 ttl=64 time=5.36 ms
64 bytes from 192.168.128.9: icmp_seq=10 ttl=64 time=4.05 ms
64 bytes from 192.168.128.9: icmp_seq=11 ttl=64 time=3.61 ms
64 bytes from 192.168.128.9: icmp_seq=12 ttl=64 time=3.70 ms
64 bytes from 192.168.128.9: icmp_seq=13 ttl=64 time=4.61 ms
64 bytes from 192.168.128.9: icmp_seq=14 ttl=64 time=8.87 ms
64 bytes from 192.168.128.9: icmp_seq=15 ttl=64 time=8.08 ms
```

Ahora se crea un directorio y entramos a ella en ambas  
maquinas

**mkdir tcp-ipc** (Crear directorio)

**cd tcp-ipc** (Cambiar de directorio)

En debían

```
root@vbox:~# mkdir ~/tcp-ipc
root@vbox:~# cd ~/tcp-ipc
root@vbox:~/tcp-ipc#
```

En Ubuntu

```
root@crsthian-VirtualBox:/home/crsthian# mkdir tcp-ipc
root@crsthian-VirtualBox:/home/crsthian# cd tcp-ipc
root@crsthian-VirtualBox:/home/crsthian/tcp-ipc#
```



SC-CER96940

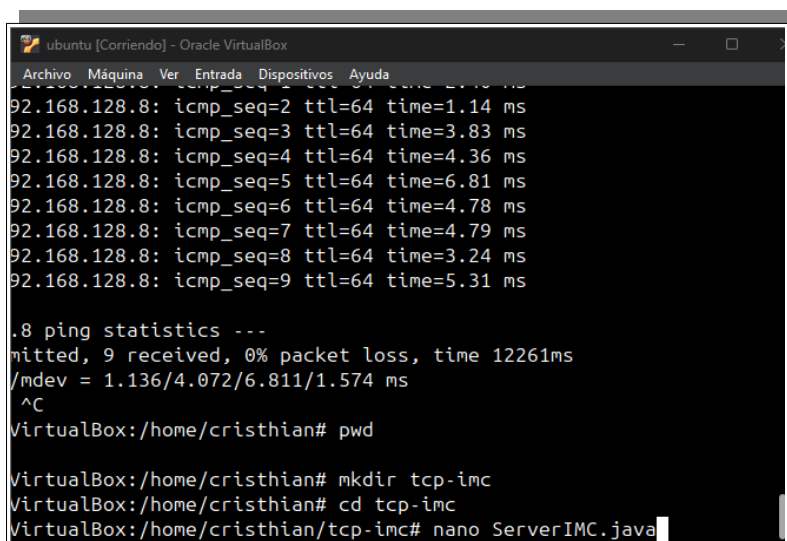


*"Formando líderes para la construcción de un nuevo país en paz"*

Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750

Ahora en el servidor(Ubuntu) se creara un archivo **ServerIMC.java** dentro de la carpeta **tcp-ipc** y se modificara con el comando

**nano ServerIMC.java**



```
VirtualBox: /home/cristhian# ping -c 9 92.168.128.8
92.168.128.8: icmp_seq=1 ttl=64 time=1.14 ms
92.168.128.8: icmp_seq=2 ttl=64 time=3.83 ms
92.168.128.8: icmp_seq=3 ttl=64 time=4.36 ms
92.168.128.8: icmp_seq=4 ttl=64 time=6.81 ms
92.168.128.8: icmp_seq=5 ttl=64 time=4.78 ms
92.168.128.8: icmp_seq=6 ttl=64 time=4.79 ms
92.168.128.8: icmp_seq=7 ttl=64 time=3.24 ms
92.168.128.8: icmp_seq=8 ttl=64 time=5.31 ms
92.168.128.8: icmp_seq=9 ttl=64 time=5.31 ms

8 ping statistics ---
8 transmitted, 9 received, 0% packet loss, time 12261ms
rtt min/avg/max/mdev = 1.136/4.072/6.811/1.574 ms
^C
VirtualBox: /home/cristhian# pwd
VirtualBox: /home/cristhian# mkdir tcp-ipc
VirtualBox: /home/cristhian# cd tcp-ipc
VirtualBox: /home/cristhian/tcp-ipc# nano ServerIMC.java
```

Y en la clase de java se coloca lo siguiente:



```
GNU nano 7.2 ServerIMC.java
1 public class ServerIMC {
2     public static void main(String[] args) {
3         int puerto = 5000;
4
5         try {
6             ServerSocket serverSocket = new ServerSocket(puerto);
7             System.out.println("Servidor IMC escuchando en el puerto " + puerto + "...");
8
9             while (true) {
10                 Socket socket = serverSocket.accept();
11                 System.out.println("Cliente conectado: " + socket.getInetAddress());
12
13                 BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
14                 PrintWriter output = new PrintWriter(socket.getOutputStream(), true);
15
16                 // Leer datos del cliente
17                 String sexo = input.readLine();
18                 int edad = Integer.parseInt(input.readLine());
19                 double peso = Double.parseDouble(input.readLine());
20                 double altura = Double.parseDouble(input.readLine());
21
22                 // Calcular IMC
23                 double imc = peso / (altura * altura);
24             }
25         } catch (IOException e) {
26             e.printStackTrace();
27         }
28     }
29 }
```



```
// Solo mostrar recomendación si es menor de edad
if (edad < 18) {
    mensaje += "⚠️Recomendación: El IMC puede no ser representativo para menores de edad.\n";
}

// Enviar la respuesta al cliente
output.println(mensaje);

socket.close();
System.out.println("Cliente desconectado.\n");
}

} catch (IOException e) {
    e.printStackTrace();
}
}
```

```
double imc = peso / (altura * altura);
String categoria;

// Determinar la categoría del IMC
if (imc <= 18.4) {
    categoria = "Bajo peso";
} else if (imc <= 24.9) {
    categoria = "Peso normal";
} else if (imc <= 29.9) {
    categoria = "Sobrepeso";
} else if (imc <= 34.9) {
    categoria = "Obesidad grado 1";
} else if (imc <= 39.9) {
    categoria = "Obesidad grado 2";
} else {
    categoria = "Obesidad grado 3";
}

// Construir respuesta
String mensaje = "Sexo: " + sexo + "\n" +
    "Edad: " + edad + " años\n" +
    "IMC: " + String.format("%.2f", imc) + "\n" +
    "Categoría: " + categoria + "\n";
```

Ahora en la maquina cliente(debían se crea el archivo **ClienteIMC.java** de la siguiente manera

```
/root/tcp-imc
root@vbox:~/tcp-imc# nano ClienteIMC.java
```

Posteriormente en la clase java se coloca lo siguiente

```
GNU nano 8.4 ClienteIMC.java
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class ClienteIMC {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Uso: java ClienteIMC <IP servidor> <puerto>");
            return;
        }

        String host = args[0];
        int puerto = Integer.parseInt(args[1]);

        try (Socket socket = new Socket(host, puerto)) {
            BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            PrintWriter output = new PrintWriter(socket.getOutputStream(), true);
            Scanner sc = new Scanner(System.in);

            // Capturar datos del usuario
            System.out.print("Ingrese su sexo (Hombre/Mujer): ");
            String sexo = sc.nextLine();

            System.out.print("Ingrese su edad: ");
            int edad = sc.nextInt();
```



**ACREDITACIÓN INSTITUCIONAL**  
*Avanzamos... ¡Es nuestro objetivo!*



```
int edad = sc.nextInt();

System.out.print("Ingrese su peso (kg): ");
double peso = sc.nextDouble();

System.out.print("Ingrese su altura (m): ");
double altura = sc.nextDouble();

// Enviar datos al servidor
output.println(sexo);
output.println(edad);
output.println(peso);
output.println(altura);

// Leer respuesta del servidor
String respuesta;
while ((respuesta = input.readLine()) != null) {
    System.out.println(respuesta);
}

} catch (IOException e) {
    System.out.println("Error: " + e.getMessage());
}

}
```

Teniendo ya los archivos lo que procede es compilar los programas con

**javac ServerIMC.java**

Desde Ubuntu

```
oot@cristhian-VirtualBox:/home/cristhian/tcp-imc# javac ServerIMC.java
oot@cristhian-VirtualBox:/home/cristhian/tcp-imc#
```

Desde Debian

```
root@vbox:~/tcp-imc# javac ClienteIMC.java
```

Ahora en el servidor(Ubuntu) hacemos que escuche por el **puerto 5000** de la siguiente forma

```
root@cristhian-VirtualBox:/home/cristhian/tcp-imc# javac ServerIMC.java
root@cristhian-VirtualBox:/home/cristhian/tcp-imc# java ServerIMC
Servidor IMC escuchando en el puerto 5000...
```



SC-CER96940



*"Formando líderes para la construcción de un nuevo país en paz"*

Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750





## ACREDITACIÓN INSTITUCIONAL

*Avanzamos... ¡Es nuestro objetivo!*



Desde el cliente(Debian) nos conectamos al servidor en donde escribimos el genero, edad el peso y la altura

```
root@vbox:~/tcp-imc# java ClienteIMC 192.168.128.9 5000
Ingrese su sexo (Hombre/Mujer): Hombre
Ingrese su edad: 17
Ingrese su peso (kg): 67
Ingrese su altura (m): 1.68
Sexo: Hombre
Edad: 17 años
IMC: 23,74
Categoría: Peso normal
⚠ Recomendación: El IMC puede no ser representativo para menores de edad.

root@vbox:~/tcp-imc# java ClienteIMC 192.168.128.9 5000
Ingrese su sexo (Hombre/Mujer): Hombre
Ingrese su edad: 19
Ingrese su peso (kg): 69
Ingrese su altura (m): 1.70
Sexo: Hombre
Edad: 19 años
IMC: 23,88
Categoría: Peso normal

root@vbox:~/tcp-imc# █
```

En el servidor se vería así de esta forma

```
root@cristhian-VirtualBox:/home/cristhian/tcp-imc# java Serv
Servidor IMC escuchando en el puerto 5000...
Cliente conectado: /192.168.128.8
Cliente desconectado.

Cliente conectado: /192.168.128.8
Cliente desconectado.

Cliente conectado: /192.168.128.8
Cliente desconectado.

Cliente conectado: /192.168.128.8
Cliente desconectado.
```



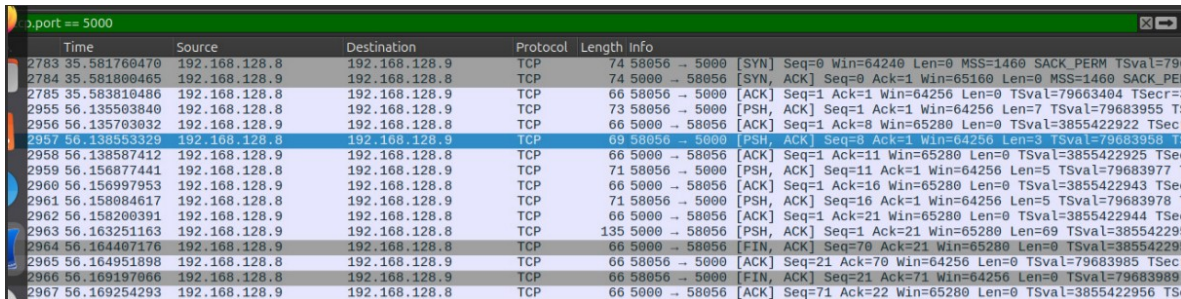
SC-CER96940



*"Formando líderes para la construcción de un nuevo país en paz"*

Universidad de Pamplona  
Pamplona - Norte de Santander - Colombia  
Tels: (7) 5685303 - 5685304 - 5685305 - Fax: 5682750

## Análisis de WireShark



Time	Source	Destination	Protocol	Length	Info
2783.35.581760470	192.168.128.8	192.168.128.9	TCP	74	58056 → 5000 [SYN, Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=79683977 TSecr=0 Win=0 Len=0]
2784.35.581800465	192.168.128.9	192.168.128.8	TCP	74	5000 → 58056 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=79683977 TSecr=0 Win=0 Len=0
2785.35.583810486	192.168.128.8	192.168.128.9	TCP	66	58056 → 5000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=79683977 TSecr=0 Win=0 Len=0
2955.56.135503840	192.168.128.8	192.168.128.9	TCP	73	58056 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=7 TSval=79683955 TSecr=0 Win=0 Len=7
2956.56.135703032	192.168.128.8	192.168.128.9	TCP	66	5000 → 58056 [ACK] Seq=1 Ack=8 Win=65280 Len=0 TSval=3855422922 TSecr=0 Win=0 Len=0
2957.56.138553329	192.168.128.8	192.168.128.9	TCP	69	58056 → 5000 [PSH, ACK] Seq=8 Ack=1 Win=64256 Len=3 TSval=79683958 TSecr=0 Win=0 Len=3
2958.56.138587412	192.168.128.9	192.168.128.8	TCP	66	5000 → 58056 [ACK] Seq=1 Ack=11 Win=65280 Len=0 TSval=3855422925 TSecr=0 Win=0 Len=0
2959.56.156877441	192.168.128.8	192.168.128.9	TCP	71	58056 → 5000 [PSH, ACK] Seq=11 Ack=1 Win=64256 Len=5 TSval=79683977 TSecr=0 Win=0 Len=5
2960.56.156997953	192.168.128.9	192.168.128.8	TCP	66	5000 → 58056 [ACK] Seq=1 Ack=16 Win=65280 Len=0 TSval=3855422943 TSecr=0 Win=0 Len=0
2961.56.158084617	192.168.128.8	192.168.128.9	TCP	71	58056 → 5000 [PSH, ACK] Seq=16 Ack=1 Win=64256 Len=5 TSval=79683978 TSecr=0 Win=0 Len=5
2962.56.158200391	192.168.128.9	192.168.128.8	TCP	66	5000 → 58056 [ACK] Seq=1 Ack=21 Win=65280 Len=0 TSval=3855422944 TSecr=0 Win=0 Len=0
2963.56.163251163	192.168.128.8	192.168.128.9	TCP	135	5000 → 58056 [PSH, ACK] Seq=1 Ack=21 Win=65280 Len=69 TSval=385542295 TSecr=0 Win=0 Len=69
2964.56.164407176	192.168.128.9	192.168.128.8	TCP	66	5000 → 58056 [FIN, ACK] Seq=70 Ack=21 Win=65280 Len=0 TSval=385542295 TSecr=0 Win=0 Len=0
2965.56.164951898	192.168.128.8	192.168.128.9	TCP	66	58056 → 5000 [ACK] Seq=21 Ack=70 Win=64256 Len=0 TSval=79683985 TSecr=0 Win=0 Len=0
2966.56.169197066	192.168.128.8	192.168.128.9	TCP	66	58056 → 5000 [FIN, ACK] Seq=21 Ack=71 Win=64256 Len=0 TSval=79683989 TSecr=0 Win=0 Len=0
2967.56.169254293	192.168.128.9	192.168.128.8	TCP	66	5000 → 58056 [ACK] Seq=71 Ack=22 Win=65280 Len=0 TSval=3855422956 TSecr=0 Win=0 Len=0

- La captura de Wireshark evidencia el correcto funcionamiento del protocolo TCP en la comunicación cliente-servidor implementada en Java.
- Se observa el proceso de *three-way handshake*, el intercambio bidireccional de datos y la confirmación de entrega mediante paquetes ACK.
- Los resultados confirman que el servidor IMC recibe correctamente los datos del cliente, realiza el cálculo y envía la respuesta sin pérdida de información, demostrando una transmisión confiable sobre TCP/IP.