

Trabajo TCP - Calculadora Remota

Yessica Tatiana Gualteros Ayala - 1123530064
Leidi Tatiana Nieto Goyeneche - 1048823584

Luz Marina Santos Jaimes
Docente

Universidad de Pamplona
Facultad de Ingeniería y Arquitectura
Redes
Pamplona, Colombia
2025

Trabajo TCP - Calculadora Remota

Objetivo General: Reforzar los conocimientos del protocolo TCP mediante el desarrollo de una calculadora remota que utilice sockets tcp.

Descripción: Desarrollar una aplicación C/S empleando programación en sockets tcp con cualquier lenguaje de programación. Los procesos servidor y cliente deben estar en la misma red en máquinas diferentes.

Servidor: Escuchando permanentemente conexiones de clientes que solicitan realizar una operación matemática entre: +, -, * y /. El servidor recibe los operandos y operador y procede a realizar la operación, y a continuación devuelve el resultado al proceso cliente.

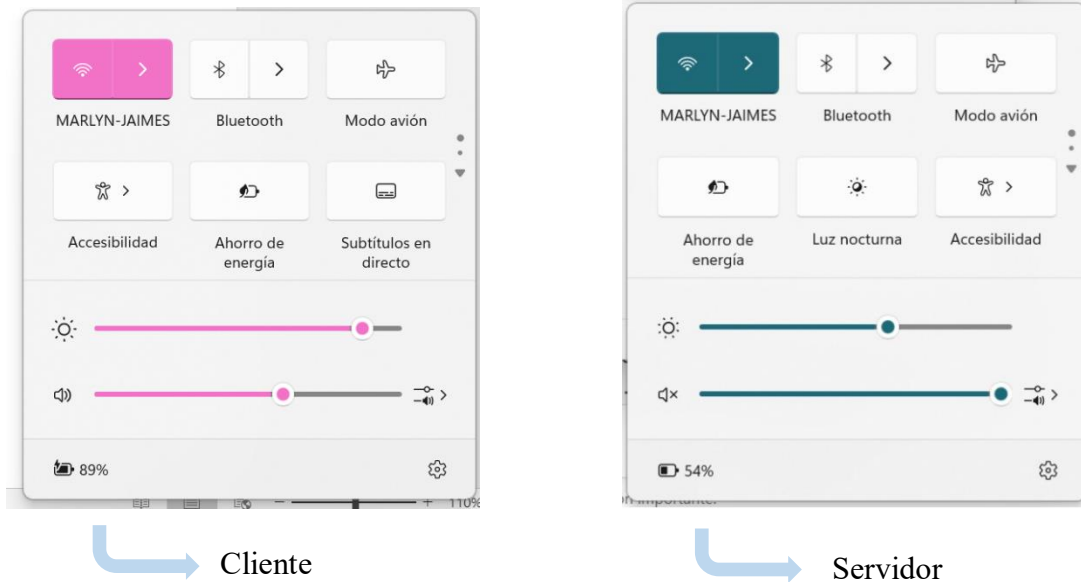
Cliente: El cliente solicita al usuario dos números (operandos) y la operación (+, -, * y /). A continuación, se conecta con el servidor para enviarle la información necesaria para que procese y devuelva el resultado. Una vez devuelto el resultado por parte del servidor se muestra en pantalla del cliente y se cierra la conexión con el servidor.

Entregable del trabajo:

1. Código fuente completo, documentado y funcional (subido en GitHub).
2. Archivo readme.txt con explicación del funcionamiento de la calculadora remota.
3. Documento wireshark con captura de mensajes intercambiados entre cliente y servidor de la captura remota.
4. Enviar al correo sólo enlace del trabajo a GitHub con los integrantes del mismo.

Configuración de Máquinas

1. Conecta ambas computadoras a la misma red local (mismo Wi-Fi o LAN).



2. Verifica que tienen Python instalado.

```
Microsoft Windows [Versión 10.0.26100.6899]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\tatia>python --version
Python 3.11.9
```

Cliente

```
Microsoft Windows [Versión 10.0.26100.4349]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\tatia>python --version
Python 3.11.9
```

Servidor

3. En la PC Servidor, abre una consola y ejecuta:

```
Microsoft Windows [Versión 10.0.26100.4349]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\tatia>ipconfig

Configuración IP de Windows

Adaptador de Ethernet Ethernet 3:

    Sufijo DNS específico para la conexión. . . :
    Vínculo dirección IPv6 local. . . . . : fe80::f228:63a8:b6e1:ab59%4
    Dirección IPv4. . . . . : 192.168.56.1
    Máscara de subred. . . . . : 255.255.255.0
    Puerta de enlace predeterminada. . . . . :
```

4. Verifica que ambas computadoras pueden comunicarse:

- Desde la PC cliente:

```
C:\>ping 192.168.1.8

Haciendo ping a 192.168.1.8 con 32 bytes de datos:
Respuesta desde 192.168.1.8: bytes=32 tiempo=41ms TTL=64
Respuesta desde 192.168.1.8: bytes=32 tiempo=605ms TTL=64
Respuesta desde 192.168.1.8: bytes=32 tiempo=96ms TTL=64
Respuesta desde 192.168.1.8: bytes=32 tiempo=233ms TTL=64

Estadísticas de ping para 192.168.1.8:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 41ms, Máximo = 605ms, Media = 243ms
```

- Desde la PC servidor:

```
C:\Windows\System32>ping 192.168.1.10

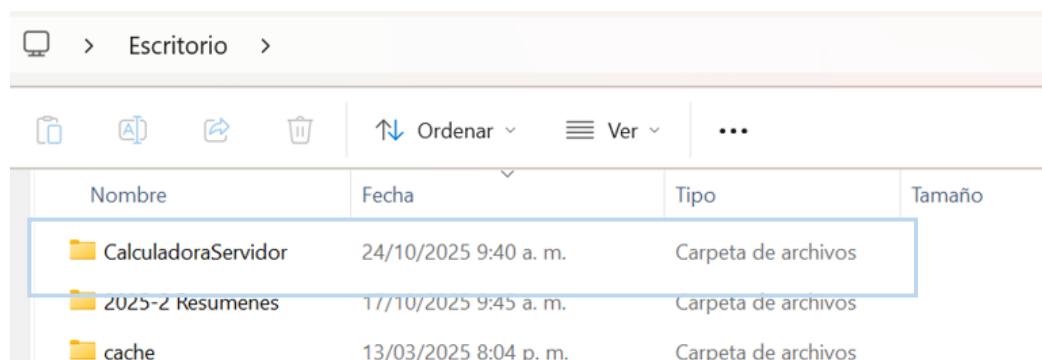
Haciendo ping a 192.168.1.10 con 32 bytes de datos:
Respuesta desde 192.168.1.10: bytes=32 tiempo=166ms TTL=128
Respuesta desde 192.168.1.10: bytes=32 tiempo=234ms TTL=128
Respuesta desde 192.168.1.10: bytes=32 tiempo=3ms TTL=128
Respuesta desde 192.168.1.10: bytes=32 tiempo=7ms TTL=128

Estadísticas de ping para 192.168.1.10:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 3ms, Máximo = 234ms, Media = 102ms

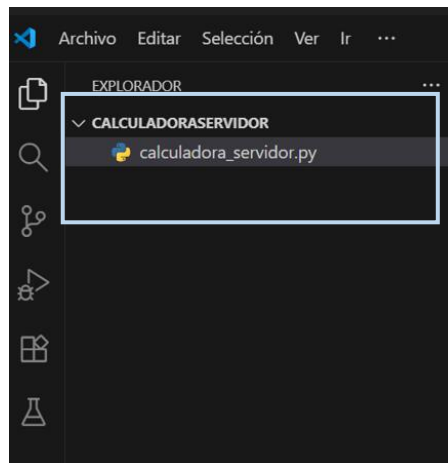
C:\Windows\System32>
```

5. Preparar la pc servidor:

- En el Escritorio, crea una carpeta llamada “CalculadoraServidor”:



- Dentro de esa carpeta, abre Visual Studio Code.



- Código:

```
calculadora_servidor.py X
calculadora_servidor.py > ...
1  import socket # Importamos la librería necesaria para manejar sockets
2
3  # CONFIGURACIÓN DEL SERVIDOR
4  HOST = "0.0.0.0" # Escucha en todas las interfaces de la red disponible.
5  PORT = 5000      # Puerto donde escuchará el servidor
6
7  # CREACIÓN DEL SOCKET TCP
8
9  # AF_INET -> Familia de direcciones IPv4
10 # SOCK_STREAM -> Tipo de socket TCP (orientado a conexión)
11 servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12 servidor.bind((HOST, PORT)) # Asociamos el socket con la dirección IP y el puerto definidos
13 servidor.listen(1) # Ponemos el socket en modo escucha, máximo 1 cliente en cola
14 print(f"Servidor escuchando en {HOST}:{PORT}...")
15
16 # CICLO PRINCIPAL DEL SERVIDOR
17
18 # Este ciclo permite atender múltiples clientes uno tras otro
19 while True:
20     # Esperamos a que un cliente se conecte
21     conn, addr = servidor.accept()
22     print(f"Conexión establecida con: {addr}")
23
24     data = conn.recv(1024).decode() # Recibimos los datos enviados por el cliente (máximo 1024 bytes)
25     # Si no se recibe nada, se sale del bucle
26     if not data:
27         break
```

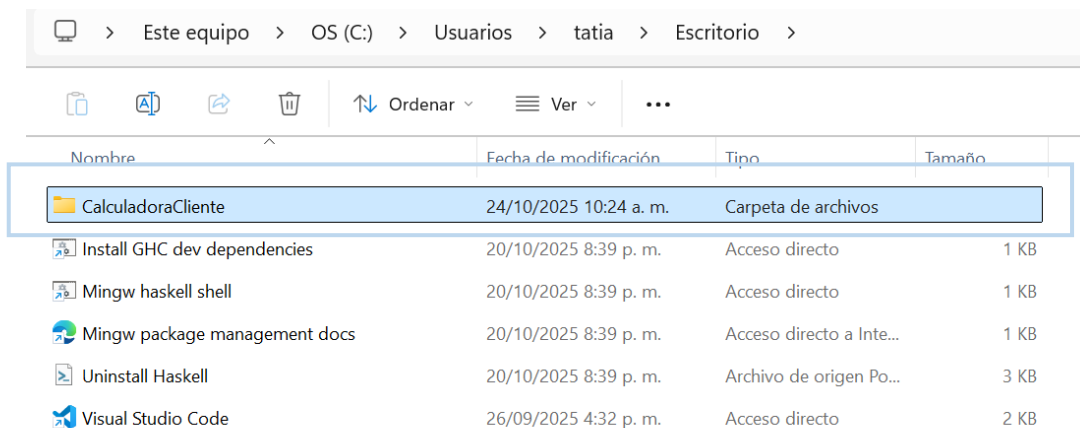
```

28
29     # Dividimos la cadena recibida (por ejemplo: "5 + 3")
30     partes = data.split()
31     num1 = float(partes[0]) # Primer número
32     operador = partes[1]   # Operador (+, -, *, /)
33     num2 = float(partes[2]) # Segundo número
34
35
36     # PROCESAMIENTO DE LA OPERACIÓN
37
38     if operador == '+':
39         resultado = num1 + num2
40     elif operador == '-':
41         resultado = num1 - num2
42     elif operador == '*':
43         resultado = num1 * num2
44     elif operador == '/':
45         # Evitar división por cero
46         resultado = num1 / num2 if num2 != 0 else "Error: división por cero"
47     else:
48         resultado = "Operador inválido"
49
50     # ENVÍO DEL RESULTADO AL CLIENTE
51
52     conn.send(str(resultado).encode())
53     print(f"Resultado enviado al cliente: {resultado}")
54     conn.close() # Cerramos la conexión con ese cliente

```

6. Preparar la pc cliente:

- En el Escritorio, crea una carpeta llamada “CalculadoraCliente”:



- Dentro de esa carpeta, abre Visual Studio Code.

```
calculadora_cliente.py X
calculadora_cliente.py > ...
1  import socket                # Librería estándar 'socket' para crear sockets TCP/IP.
2  import tkinter as tk         # Importa tkinter para crear la interfaz gráfica (alias 'tk' para abreviar).
3  from tkinter import messagebox # Importa el submódulo messagebox para mostrar diálogos (errores, avisos).
4
5  # -----
6  # Configuración de red
7  # -----
8  SERVER_IP = "192.168.1.8"     # IP del servidor.
9  PORT = 5000                  # Puerto TCP donde el servidor está escuchando (debe coincidir con el servidor).
10
11 # -----
12 # Función que envía la operación y recibe el resultado
13 # -----
14 def calcular():
15     """
16     6. Función llamada cuando el usuario hace clic en "Calcular".
17     Reúne los datos de la interfaz, valida, abre conexión TCP, envía el mensaje,
18     recibe el resultado y lo muestra en la etiqueta de resultado.
19     """
20
21     num1 = entrada_num1.get()   # Lee el texto actual del campo de entrada 'entrada_num1'.
22     num2 = entrada_num2.get()   # Lee el texto actual del campo de entrada 'entrada_num2'.
23     operador = operacion.get()  # Obtiene el operador seleccionado en el OptionMenu ('+', '-', '*', '/').
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
#Validación básica: asegurar que los campos no estén vacíos.
if not num1 or not num2:
    messagebox.showwarning("Datos incompletos", "Debe ingresar ambos números.")
    return # Sale de la función si faltan datos.
# Validación de formato: intentar convertir a float para asegurarnos que sean números.
try:
    float(num1) #Intento de conversión para validar; no guardamos el valor convertido aún.
    float(num2)
except ValueError:
    # Si la conversión falla lanzamos un aviso al usuario y no continuamos.
    messagebox.showwarning("Valor inválido", "Los valores deben ser números (ej: 3.5 o 7).")
    return
# Construimos el mensaje con el formato que espera el servidor: "num1 operador num2"
mensaje = f"{num1} {operador} {num2}"
# Intento de conexión y comunicación dentro de un bloque try/except para capturar errores de red.
try:
    cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Crea un socket IPv4 (AF_INET) orientado a conexión/TCP (SOCK_STREAM).
    cliente.connect((SERVER_IP, PORT))
    # Inicia la conexión TCP con el servidor usando la IP y el puerto configurados.
    # Si el servidor no responde o la IP es incorrecta, esto lanzará una excepción.
```

```

54     cliente.send(mensaje.encode())
55     # Envía el mensaje al servidor codificado a bytes (UTF-8 por defecto).
56     # .encode() convierte la cadena a bytes para la transmisión.
57
58     resultado = cliente.recv(1024).decode()
59     # Recibe hasta 1024 bytes de respuesta del servidor y los decodifica a string.
60     # Asumimos que el servidor envía una respuesta corta (resultado de la operación).
61
62     etiqueta_resultado.config(text=f"Resultado: {resultado}")
63     # Actualiza la etiqueta en la interfaz con el resultado recibido.
64
65     cliente.close()
66     # Cierra el socket para liberar recursos y terminar la conexión TCP.
67 except Exception as e:
68     # Si ocurre cualquier excepción (conexión rechazada, timeout, etc.), mostramos un error.
69     messagebox.showerror("Error de conexión", f"No se pudo conectar al servidor.\n\nDetalle: {e}")
70
71 # -----
72 # Construcción de la interfaz gráfica
73 # -----
74 ventana = tk.Tk() # Crea la ventana principal de la aplicación.
75 ventana.title("Calculadora Remota TCP") # Título de la ventana.
76 ventana.geometry("320x280") # Tamaño inicial de la ventana.
77 ventana.resizable(False, False) # Evita que el usuario cambie el tamaño.
78 ventana.configure(bg="#F3E6F8")

```

```

80 # Etiqueta y campo para el primer número
81 tk.Label(ventana, text="Número 1:", bg="#F3E6F8", anchor="w").pack(padx=10, pady=(12,2), fill="x")
82 entrada_num1 = tk.Entry(ventana) # Campo de entrada para el primer número.
83 entrada_num1.pack(padx=10, pady=(0,8), fill="x")
84
85 # Etiqueta y campo para el segundo número
86 tk.Label(ventana, text="Número 2:", bg="#F3E6F8", anchor="w").pack(padx=10, pady=(6,2), fill="x")
87 entrada_num2 = tk.Entry(ventana) # Campo de entrada para el segundo número.
88 entrada_num2.pack(padx=10, pady=(0,8), fill="x")
89
90 # Etiqueta y menú para seleccionar la operación
91 tk.Label(ventana, text="Operación:", anchor="w").pack(padx=10, pady=(6,2), fill="x")
92 operacion = tk.StringVar(value="+") # Variable de control que guarda la opción seleccionada; valor inicial '+'.
93 menu_operacion = tk.OptionMenu(ventana, operacion, "+", "-", "*", "/")
94
95 # OptionMenu crea un desplegable con las cuatro operaciones y lo enlaza a 'operacion'.
96 menu_operacion.pack(padx=10, pady=(0,8), fill="x")
97
98 # Botón principal que ejecuta la función 'calcular' al hacer clic
99 boton_calcular = tk.Button(ventana, text="Calcular", command=calcular)
100 boton_calcular.pack(padx=10, pady=(6,10), fill="x")
101

```

```

102 # Etiqueta del resultado
103 etiqueta_resultado = tk.Label(
104     ventana,
105     text="Resultado:",
106     bg="#F3E6F8",
107     font=("Arial", 10, "bold"),
108     fg="black"
109 )
110 etiqueta_resultado.pack(padx=10, pady=(10,6), fill="x")
111
112 # Inicia el loop principal de la interfaz; la aplicación queda a la espera de eventos (clicks, entradas).
113 ventana.mainloop()
114

```


7. Ejecutar:

- En la cmd del servidor:

```
cd Escritorio\CalculadoraServidor  
python calculadora_servidor.py
```

```
Directorio de C:\Users\tatia\Desktop  
24/10/2025 10:22 a. m. <DIR> .  
20/10/2025 10:14 p. m. <DIR> ..  
09/03/2025 06:44 p. m. <DIR> 2024  
22/06/2025 06:52 p. m. <DIR> 2025  
21/10/2025 10:42 a. m. <DIR> 2025-2 Resumenes  
13/03/2025 08:04 p. m. <DIR> cache  
24/10/2025 09:45 a. m. <DIR> CalculadoraServidor  
13/10/2025 03:47 p. m. <DIR> Chii  
12/11/2024 06:14 a. m. 4.739 debug.log  
25/09/2024 02:53 p. m. 109.732.304 draw.io-24.7.8-windows-installer.exe  
06/10/2023 05:25 a. m. 1.107 Eclipse IDE for Java Developers - 2023-09.lnk  
29/05/2025 06:16 p. m. 843.439.680 ideaIC-2025.1.1.1.exe  
20/10/2025 08:36 p. m. 709 Install GHC dev dependencies.lnk  
01/10/2024 09:15 p. m. <DIR> JUEGO 3 SEMESTRE IMAGENES  
27/10/2023 06:11 p. m. 2.161 Liberar Espacio.lnk  
21/02/2025 08:14 p. m. 2.361 Microsoft Edge.lnk  
20/10/2025 08:36 p. m. 463 Mingw haskell shell.lnk  
20/10/2025 08:36 p. m. 135 Mingw package management docs.url  
15/09/2025 07:14 p. m. 1.182.435.288 OllamaSetup.exe  
16/10/2024 02:02 p. m. 388.570.408 postgresql-17.0-1-windows-x64.exe  
15/06/2025 02:55 p. m. <DIR> Tatiana Nieto  
20/10/2025 08:36 p. m. 2.650 Uninstall Haskell.ps1  
13/12/2024 01:55 p. m. 1.454 Visual Studio Code.lnk  
13 archivos 2.524.193.459 bytes  
10 dirs 195.582.320.640 bytes libres  
  
C:\Users\tatia\Desktop>cd CalculadoraServidor  
  
C:\Users\tatia\Desktop\CalculadoraServidor>python calculadora_servidor.py  
Servidor escuchando en 0.0.0.0:5000...
```

- En la cmd del Cliente:

```
cd Escritorio\CalculadoraCliente  
python calculadora_cliente.py
```

```
PS C:\Users\tatia> cd Escritorio  
PS C:\Users\tatia\Escritorio> cd CalculadoraCliente  
PS C:\Users\tatia\Escritorio\CalculadoraCliente> python calculadora_cliente.py
```

Calculadora Remota TCP

Número 1:

2

Número 2:

8

Operación:

+

Calcular

Resultado: 10.0

- CMD del pc Servidor

```

C:\Users\tatia>cd Desktop
C:\Users\tatia\Desktop>cd CalculadoraServidor
C:\Users\tatia\Desktop\CalculadoraServidor>python calculadora_servidor.py
Servidor escuchando en 0.0.0.0:5000...
Conexión establecida con: ('192.168.1.10', 64239)
Resultado enviado al cliente: 10.0

```

8. Captura En Wireshark

captura.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

Aplique un filtro de visualización ... <Ctrl-/>

No.	Time	Sour	Dest	Protocol	Length	Info
350	25.497651	Sa...	Br...	ARP	68	Who has 192.168.100.1? Tell 192.168.100.2
351	26.726524	19...	19...	UDP	77	47802 → 15600 Len=35
352	27.034825	19...	22...	UDP	217	8001 → 8001 Len=175
353	27.400551	19...	19...	NBNS	92	Name query NB WORKGROUP<1d>
354	27.640032	Sa...	Br...	ARP	68	Who has 192.168.100.1? Tell 192.168.100.2
355	28.038411	19...	10...	TLSv1.2	107	Application Data
356	28.083292	19...	19...	TCP	66	80 → 36179 [FIN, ACK] Seq=159910 Ack=2146 Win=18816 Len=0 TSval=2972497789 TSecr=3922294
357	28.083468	19...	19...	TCP	66	36179 → 80 [FIN, ACK] Seq=2146 Ack=159911 Win=241792 Len=0 TSval=393533 TSecr=2972497789
358	28.136787	10...	19...	TLSv1.2	107	Application Data
359	28.136837	19...	10...	TCP	66	43470 → 443 [ACK] Seq=42 Ack=42 Win=342 Len=0 TSval=393546 TSecr=1969951825
360	28.167009	19...	19...	TCP	66	80 → 36179 [ACK] Seq=159911 Ack=2147 Win=18816 Len=0 TSval=2972497875 TSecr=393533
361	28.879443	19...	22...	UDP	217	8001 → 8001 Len=175
362	29.038677	19...	15...	TLSv1.2	107	Application Data
363	29.086892	15...	19...	TLSv1.2	107	Application Data

> Frame 179: Packet, 405 bytes on wire (3240 bits), 405 bytes captured (3240 bits) on

> Ethernet II, Src: Arcadyan_d6:13:16 (e4:3e:d7:d6:13:16), Dst: HonHaiPrecis_cd:db:51

> Internet Protocol Version 4, Src: 190.25.226.68, Dst: 192.168.100.15

> Transmission Control Protocol, Src Port: 80, Dst Port: 36179, Seq: 36683, Ack: 2146

Source Port: 80

Destination Port: 36179

[Stream index: 2]

[Stream Packet Number: 64]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 339]

Sequence Number: 36683 (relative sequence number)

Sequence Number (raw): 1230557178

[Next Sequence Number: 37022 (relative sequence number)]

Acknowledgment Number: 2146 (relative ack number)

0000 78 e4 00 cd db 51 e4 3e d7 d6 13 16 08 00 45 68 x...Q>-----Eh

0010 01 87 ca 85 40 00 32 06 b7 6d be 19 e2 44 c0 a8 @.2...m...D..

0020 64 0f 00 50 8d 53 49 58 cf fa 46 9e 8f d9 80 18 d..P..SIX...F...

0030 00 93 a3 3d 00 00 01 01 00 0a b1 2c a3 2a 00 05:.....*

0040 fc 0f 0a 85 3a cc c6 0c 11 48 73 db 6a ae a9 41:..Hs n...A

0050 65 4d f4 59 64 89 d4 87 27 9f 4d b9 b8 1c d7 27 eM.Yd... 'M...-'

0060 cf 1d 01 93 89 40 39 6d 08 5e 45 07 3d 00 b4 cc@9m...^E...-

0070 d6 68 7a b9 28 9d e7 57 54 57 b5 0d 40 1b b1 11 .hz...W TW...@...

0080 4a f1 8a 1f 01 9e 8c ae 61 4d 3d 5e 6a 71 71 0a J.....aM="jqj...

0090 15 40 83 f3 90 2b a0 fc ca a9 81 9c 3d b4 d0 13 @.....:.....-

00a0 d1 b9 9f 47 c0 39 11 13 94 e2 92 e1 52 d1 f5 bb69.....R...

00b0 f3 91 a5 e5 07 c9 74 81 f2 22 77 60 0f 15 a8 a5t... "wk....

00c0 ab f5 ee 8c b7 e0 5d 7a c5 2a 36 c1 23 14 c3 2f]z...*6 #.../

00d0 a1 e2 c3 a5 00 9f a0 fc 9b 3b 5f 66 cb 0b da 68 @.....:..f...h

00e0 ae 49 3c 9e d3 2a 1e 85 00 5b 0f 0f db 00 79 65 .I<...*...[...ye

00f0 95 13 8f 37 89 e6 6a c5 86 9b e7 07 e2 21 78 cd7...j.....[X-

0100 96 39 b6 80 5b 76 bd fc 22 a5 0a 9a cb 29 f6 b7 .9...[v... ".....

0110 ed df cb 10 f2 d3 2f bf fc 14 42 95 b7 ff 16 eb/...B.....

- Filtrar por el puerto 5000

Tcp port == 5000

calculadora_remota.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port == 5000

No.	Time	Source	Destination	Protocol	Length	Info
43	24.580464	192.168.1.10	192.168.1.8	TCP	66	55712 → 5000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
44	24.580837	192.168.1.8	192.168.1.10	TCP	66	5000 → 55712 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
45	24.586660	192.168.1.10	192.168.1.8	TCP	54	55712 → 5000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
46	24.586660	192.168.1.10	192.168.1.8	TCP	60	55712 → 5000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=6
47	24.588599	192.168.1.8	192.168.1.10	TCP	58	5000 → 55712 [PSH, ACK] Seq=1 Ack=7 Win=65280 Len=4
48	24.589013	192.168.1.8	192.168.1.10	TCP	54	5000 → 55712 [FIN, ACK] Seq=5 Ack=7 Win=65280 Len=0
49	24.597297	192.168.1.10	192.168.1.8	TCP	54	55712 → 5000 [FIN, ACK] Seq=7 Ack=5 Win=65280 Len=0
50	24.597297	192.168.1.10	192.168.1.8	TCP	54	55712 → 5000 [ACK] Seq=8 Ack=6 Win=65280 Len=0
51	25.614212	192.168.1.10	192.168.1.8	TCP	54	[TCP Retransmission] 55712 → 5000 [FIN, ACK] Seq=7 Ack=6 Win=65280 Len=0
52	25.614335	192.168.1.8	192.168.1.10	TCP	54	[TCP ZeroWindow] 5000 → 55712 [ACK] Seq=6 Ack=8 Win=0 Len=0

> Frame 52: Packet, 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface
 > Ethernet II, Src: Intel_b5:ed:11 (dc:46:28:b5:ed:11), Dst: AzureWaveTec_6e:c1:49 (90:05:0d:6e:c1:49)
 > Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.10
 > Transmission Control Protocol, Src Port: 5000, Dst Port: 55712, Seq: 6, Ack: 8, Len: 0

calculadora_remota.pcapng Paquetes: 148 - Displayed: 10 (6.8%) - Perdido: 0 (0.0%) Perfil: Default

9. Enlace del repositorio:

Link: <https://github.com/Tatiana2412/CalculadoraTCP.git>