



Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y
Tecnologías Avanzadas

Alumnos:

Olivares Piña Usiel Alonso

López López Arturo

Santana Islas Gerardo Leonardo

Docente Evaluador:

Mata Rivera Miguel Felix

Unidad de Aprendizaje:

Sistemas Distribuidos

Practica 1

12 de Febrero 2022

UPIITA---IPN---TELEMÁTICA Práctica 1

Sistema distribuidos

Introducción al Manejo de Sockets

Objetivos de la practica:

- Conocer como construir un Socket en Java, bajo la arquitectura CLIENTE-SERVIDOR y el enfoque de programación en red.
- Comprender el concepto de arquitectura.
- Comprender el concepto de transparencia en la comunicación.
- Conocer el manejo básico de Github.

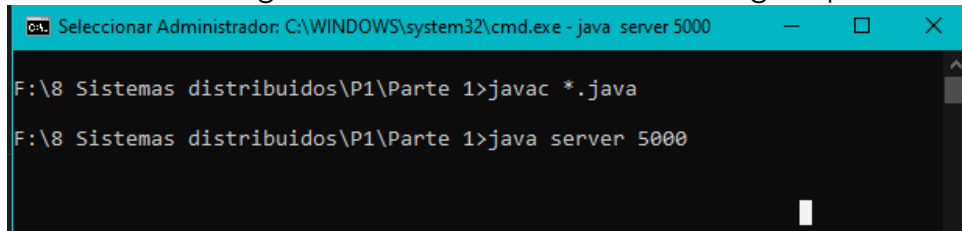
Requisitos para la realizacion de la practica:

- Un ambiente de red (alámbrica o inalámbrica).
- Tener instalado y configurado java, un IDE (e.g. Netbeans o Eclipse).
- Privilegios de administrador para configuración de la red, y de las directivas de seguridad del sistema operativo[permisos de root en Linux]).
- Descarga el código (los archivos java) desde GITHUB en: <https://github.com/migfel/practicass-SD-2022.git>.

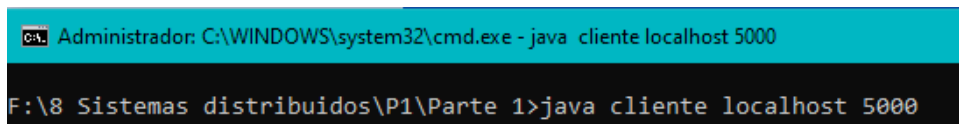
I.- Comunicación con Sockets localmente

A la hora de compilar el codigo se tuvieron unos cuantos problemas debido a las dependencias de C y Java, debido a que el compilador no se ejecutaba de manera correcta, sin embargo a la hora de reinstalarlos se llevo a proceder

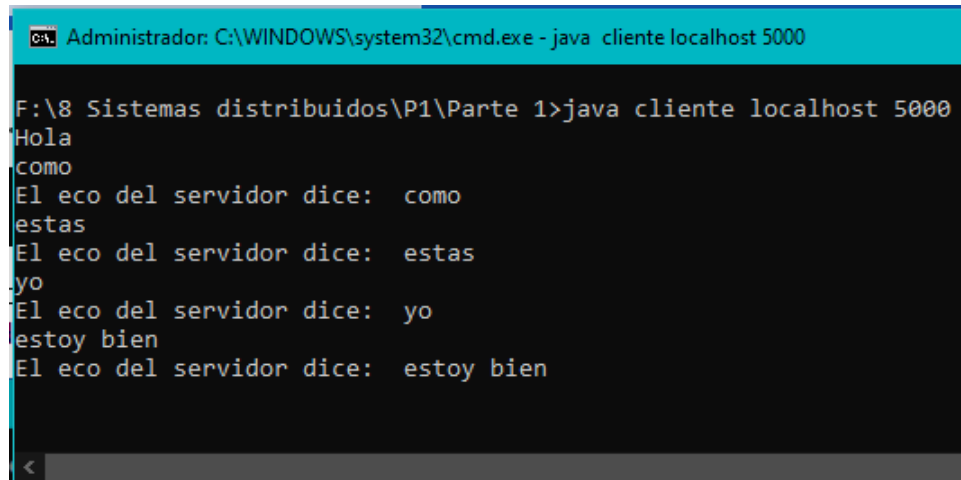
sin problemas.



```
C:\WINDOWS\system32\cmd.exe - java server 5000
F:\8 Sistemas distribuidos\P1\Parte 1>javac *.java
F:\8 Sistemas distribuidos\P1\Parte 1>java server 5000
```



```
Administrador: C:\WINDOWS\system32\cmd.exe - java cliente localhost 5000
F:\8 Sistemas distribuidos\P1\Parte 1>java cliente localhost 5000
```



```
C:\> Administrador: C:\WINDOWS\system32\cmd.exe - java cliente localhost 5000

F:\8 Sistemas distribuidos\P1\Parte 1>java cliente localhost 5000
Hola
como
El eco del servidor dice:  como
estas
El eco del servidor dice:  estas
yo
El eco del servidor dice:  yo
estoy bien
El eco del servidor dice:  estoy bien
```

Preguntas por responder en la presente practica

¿Que necesitamos para que pueda comunicarse el programa Servidor (codificado en java) con un cliente (codificado en C) y viceversa?

Para la conexión entre el programa servidor (codificado en java) y un cliente (codificado en C) se requiere de un middleware que permita enviar los datos a distintas plataformas de programación con el fin de poder acarrear los bytes de los de datos sin el riesgo que estos se pierdan debido al manejo de distintas cantidades de bytes, dando el ejemplo en donde un valor char enviado por programa codificado en java hacia un programa codificado en C mediante un socket tendrá el valor de 2 bytes, sin embargo, el programa codificado en C solo permite leer un char con el valor de un byte por lo que el middleware permite acarrear el byte sin perder la fluidez de la comunicación.

2.- Comunicación con Sockets Remotamente

Con el fin de realizar esta practica se tuvieron que realizar 2 pasos siendo que llegaron a ser problematicos debido a que existia un error en el firewall ademas del adaptador , sin embargo a traves de varias paginas , logramos arreglar el problema.

Referencias

Latecnologiaatualcance. 2022. *Como arreglar el Trafico de Entrada Bloqueado en Hamachi.* [online] Available at: <<https://latecnologiaatualcance.com/como-arreglar-el-trafico-de-entrada-bloqueado-en-hamachi/>> [Accessed 16 February 2022].

Túnel: (Tunnel:) Aceptar (OK)

Resultados locales:

Configuración del adaptador:

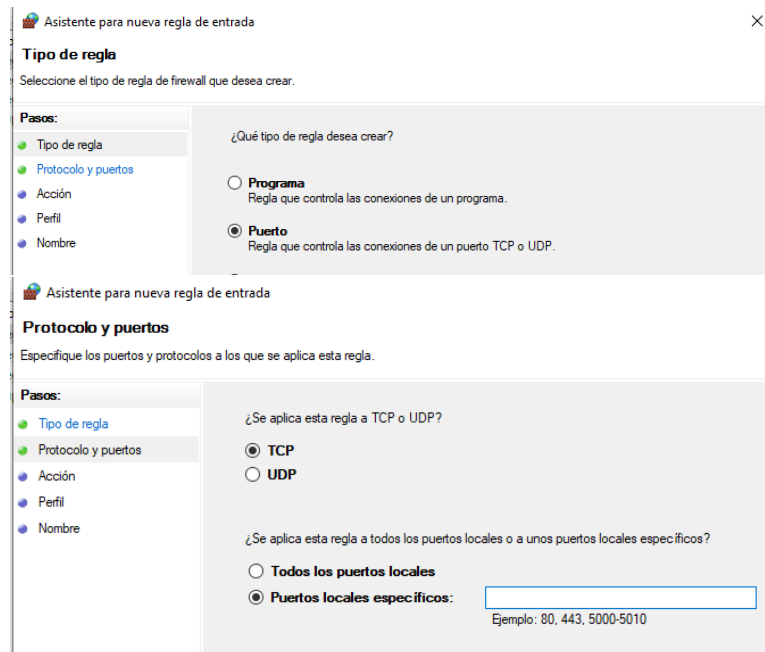
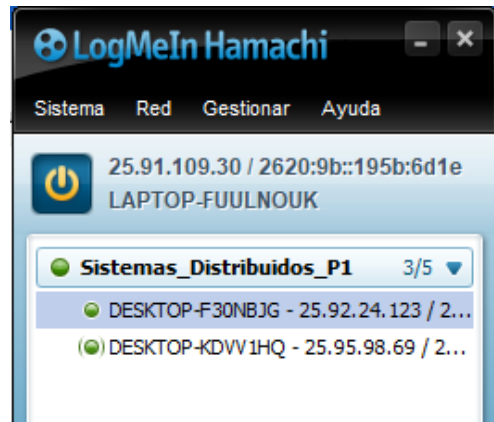
No se puede obtener la configuración del adaptador (Cannot get adapter config)

Prueba del tráfico: Aceptar (OK)

Resultados del interlocutor: [265-261-360]

Configuración del adaptador: Aceptar (OK)

Prueba del tráfico: Aceptar (OK)



Perfil

Especifique los perfiles en los que se va a aplicar esta regla.

Pasos:

- Tipo de regla
- Protocolo y puertos
- Acción
- Perfil
- Nombre

¿Cuándo se aplica esta regla?

☐ Dominio

Se aplica cuando un equipo está conectado a su dominio corporativo.

☐ Privado

Se aplica cuando un equipo está conectado a una ubicación de red privada, como una red doméstica o del lugar de trabajo.

☒ Público

Se aplica cuando un equipo está conectado a una ubicación de redes públicas.

```
C:\Users\Usiel Alonso\Desktop\Upiita\8 Semestre\Sistemas Distribuidos\practica 1>javac server.java
C:\Users\Usiel Alonso\Desktop\Upiita\8 Semestre\Sistemas Distribuidos\practica 1>java server 5000
Socket Conctado:
```

```
administrador C:\Windows\system32\cmd.exe - java cliente 25.91.109.30 5000
F:\8 Sistemas distribuidos\P1\Parte 1>java cliente 25.91.109.30 5000
Hola
Hola usiel
El eco del servidor dice: Hola usiel
como estas
El eco del servidor dice: como estas
me llamo Arturo
El eco del servidor dice: me llamo Arturo
```

¿Cómo se llama esta característica/funcionalidad en un sistema distribuido? Las características que se observan en este proceso se tratan de la concurrencia y de la transparencia para este sistema.

¿Qué es lo que permite que esta característica ocurra? Gracias al programa hamachi que nos permite traducir la dirección IP a una IP privada dentro de este mismo programa, lo que permite la conexión, debido a esto se permite la comunicación entre las maquinas, así ocurre la transparencia de este sistema.

La concurrencia ocurre al momento que el evento que implica enviar el mensaje o recibirlo, es un proceso independiente del estado de los ordenadores, ya que si los componentes fallan se tiene una tolerancia al ser eventos independientes.

3.- Comunicación con Sockets Remotamente

1.- Codifique dos programas usando sockets, en el enfoque cliente-servidor, que permita el intercambio de mensajes de texto.

- El programa Servidor (debe ser codificado en java)
- Mientras que el cliente (debe ser codificado en C)
- Funcionamiento: Cuando se conecten entre sí, el cliente enviará una cadena de texto cualquiera, por ejemplo, Hola y el Servidor debe responder con algún otro mensaje, por ejemplo, Hola que tal.

```

alonso@alonso-VirtualBox: ~/Documentos/Equipo/Server J Client C$ ./Client
Mensaje a Enviar Hola
Enviando Al servidor.....
El servidor envia: Que onda

alonso@alonso-VirtualBox:~/Documentos/Equipo/Server J Client C$

alonso@alonso-VirtualBox:~/Documentos/Equipo/Server J Client C$ ls
bash      DatoSocket.class  Servidor_Eco.java Socket_Cliente.h
Cliente   DatoSocket.java   Socket.c        Socket.h
Cliente.c Servidor_Eco.class Socket_Cliente.c

alonso@alonso-VirtualBox:~/Documentos/Equipo/Server J Client C$ java S
ervidor_Eco
En espera...

alonso@alonso-VirtualBox:~/Documentos/Equipo/Server J Client C$

```

2.- Codifique dos programas usando sockets, en el enfoque cliente-servidor, que permita que se envíen números enteros entre si

- El programa Servidor (debe ser codificado en C)
- El programa cliente (debe ser codificado en Java)
- Funcionamiento: Cuando se conecten entre sí, el cliente enviará un entero y el servidor lo incrementara en uno. Ejemplo, el cliente envía un 5 y el servidor contestará con un 6, el programa terminará cuando el cliente escriba un cero.

```

alonso@alonso-VirtualBox:~/Documentos/Equipo/ServerCClienteJ$ java Soc
ket_Cliente
Cliente
conectado
Inserta un valor de 1 - 9 a enviar
Si es 0 saldra del programa
1
Mensaje de Cliente Java Enviado 1 -- 1
Mensaje de Cliente Java Recibido 1 -- 2
2
Mensaje de Cliente Java Enviado 1 -- 2
Mensaje de Cliente Java Recibido 1 -- 3
3
Mensaje de Cliente Java Enviado 1 -- 3
Mensaje de Cliente Java Recibido 1 -- 4
4
Mensaje de Cliente Java Enviado 1 -- 4
Mensaje de Cliente Java Recibido 1 -- 5
5
Mensaje de Cliente Java Enviado 1 -- 5
Mensaje de Cliente Java Recibido 1 -- 6
6
Mensaje de Cliente Java Enviado 1 -- 6
Mensaje de Cliente Java Recibido 1 -- 7
7
Mensaje de Cliente Java Enviado 1 -- 7
Mensaje de Cliente Java Recibido 1 -- 8
8
Mensaje de Cliente Java Enviado 1 -- 8
Mensaje de Cliente Java Recibido 1 -- 9
0
Mensaje de Cliente Java Enviado 1 -- 0
Mensaje de Cliente Java Recibido 1 -- 1
alonso@alonso-VirtualBox:~/Documentos/Equipo/ServerCClienteJ$

Socket_Servidor.c:108:12: warning: unused variable 'Longitud_Cliente'
[-Wunused-variable]
108 | socklen_t Longitud_Cliente;
    |
Socket_Servidor.c:107:18: warning: unused variable 'Cliente' [-Wunused
-variable]
107 | struct sockaddr Cliente;
    |
Servidor.c:14:1: warning: return type defaults to 'int' [-Wimplicit-in
t]
14 | main ()
    |
alonso@alonso-VirtualBox:~/Documentos/Equipo/ServerCClienteJ$ ./Server
El Servidor recibio: 1
El Servidor Enviara: 2
El Servidor recibio: 2
El Servidor Enviara: 3
El Servidor recibio: 3
El Servidor Enviara: 4
El Servidor recibio: 4
El Servidor Enviara: 5
El Servidor recibio: 5
El Servidor Enviara: 6
El Servidor recibio: 6
El Servidor Enviara: 7
El Servidor recibio: 7
El Servidor Enviara: 8
El Servidor recibio: 8
El Servidor Enviara: 9
El Servidor recibio: 0
El Servidor Enviara: 1
El Servidor recibio: Hola
alonso@alonso-VirtualBox:~/Documentos/Equipo/ServerCClienteJ$

```

Con estos 2 últimos programas tuvimos varios inconvenientes , debido a que entre todos tuvimos problemas con los compiladores, ademas de poder ejecutar los archivos C y Java por lo cual tambien debido a la anterioridad con la que trabajamos con estas extensiones se tuvo problemas a la hora de la sintaxis en los archivos y su compilacion/ejecucion.

JC Mouse. (31 AGOSTO 2016). Cliente/Servidor en java y c#. 19 de febrero 2022, de jc-Mouse.net
Sitio web: <https://www.jc-mouse.net/proyectos/clienteservidor-en-java-y-c>

Anonymous. (4 Feb 2007). Socket entre C y java. 19 de febrero 2022, de chuidiang Sitio web:
http://www.chuidiang.org/java/sockets/cpp_java/cpp_java.php

Anonymous. (agosto 2018). Programación cliente servidor. 19 de febrero 2022, de xdoc Sitio web:
<https://xdoc.mx/preview/programacion-cliente-servidor-5ec1a135c2f94>