



Instituto Politécnico Nacional



Escuela Superior de Cómputo

TOKEN RING

TAREA 2

Materia:

Desarrollo De Sistemas Distribuidos

Grupo:

4CV14

Profesor:

Pineda Guerrero Carlos

Alumno:

Castro Cruces Jorge Eduardo

Boleta:

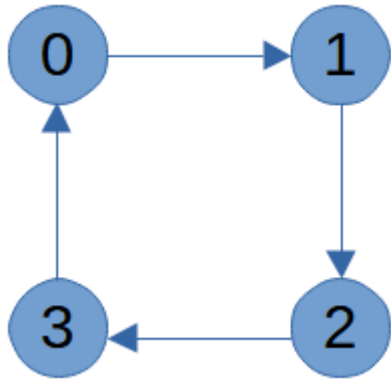
2015080213

Fecha:

Viernes, 9 de septiembre de 2021

1. Desarrollo del programa

Desarrollar un programa en Java, el cual implementará un token que se enviará de un nodo a otro nodo mediante **sockets seguros**, en una topología lógica de anillo. El anillo constará de cuatro nodos:



El token será un número entero de 64 bits.

1. Al principio el nodo 0 enviará el token al nodo 1.
2. El nodo 1 recibirá el token y lo enviará al nodo 2.
3. El nodo 2 recibirá el token y lo enviará al nodo 3.
4. El nodo 3 recibirá el token y lo enviará al nodo 0.
5. El nodo 0 recibirá el token y lo enviará al nodo 1.
6. Ir al paso 2.

Cuando un nodo reciba el token lo incrementará en 1 y desplegará el valor del token.

Cuando la cuenta en el nodo 0 llegue a 1000, el nodo 0 deberá terminar su ejecución.

Vamos a probar el programa en una sola computadora utilizando cuatro ventanas de comandos de Windows o cuatro terminales de Linux o MacOS, cada ventana ejecutará una instancia del programa.

Se deberá completar el siguiente programa (notar que este programa es un cliente y un servidor):

```
class Token
{
    static DataInputStream entrada;
```

```
static DataOutputStream salida;  
static boolean inicio = true;  
static String ip;  
static int nodo;
```

```
static long token;
```

```
static class Worker extends Thread
```

```
{  
  
    public void run()  
  
    {  
  
        Algoritmo 1  
    }  
}
```

```
public static void main(String[] args) throws Exception
```

```
{  
  
    if (args.length != 2)  
  
    {  
  
        System.err.println("Se debe pasar como parametros el numero del nodo y la IP del siguiente nodo en el  
anillo");  
  
        System.exit(1);  
    }  
}
```

```
nodo = Integer.valueOf(args[0]);  
ip = args[1];
```

```
Algoritmo 2
```

```
}  
  
}
```

Se deberá implementar los siguientes algoritmos:

Algoritmo 1

1. En un bloque try:
 - 1.1 Declarar la variable **servidor** de tipo ServerSocket,
 - 1.2. Asignar a la variable **servidor** el objeto: new ServerSocket(20000+nodo)
 - 1.3. Declarar la variable **conexion** de tipo Socket.
 - 1.4. Asignar a la variable **conexion** el objeto servidor.accept().
 - 1.5. Asignar a la variable **entrada** el objeto new DataInputStream(conexion.getInputStream()).
2. En el bloque catch desplegar el mensaje de la excepción.

Algoritmo 2

1. Declarar la variable **w** de tipo Worker.
2. Asignar a la variable **w** el objeto new Worker().
3. Invocar el método w.start().
4. Declarar la variable **conexion** de tipo Socket. Asignar null a la variable **conexion**.
5. En un ciclo:
 - 5.1 En un bloque try:
 - 5.1.1 Asignar a la variable **conexion** el objeto Socket(ip,20000+(nodo+1)%4).
 - 5.1.2 Ejecutar break para salir del ciclo.
 - 5.2 En el bloque catch:
 - 5.2.1 Invocar el método Thread.sleep(500).
6. Asignar a la variable **salida** el objeto new DataOutputStream(conexion.getOutputStream()).
7. Invocar el método w.join().
8. En un ciclo:
 - 8.1 Si la variable **nodo** es cero:
 - 8.1.1 Si la variable **inicio** es true:

- 8.1.1.1 Asignar false a la variable **inicio**.
 - 8.1.1.2 Asignar 1 a la variable **token**.
 - 8.1.2 De otra manera:
 - 8.1.2.1 Asignar a la variable **token** el resultado del método `entrada.readLong()`.
 - 8.1.2.2 Incrementar la variable **token**.
 - 8.1.2.3 Desplegar las variables **nodo** y **token**.
- 8.2 De otra manera:
 - 8.2.1 Asignar a la variable **token** el resultado del método `entrada.readLong()`.
 - 8.2.2 Incrementar la variable **token**.
 - 8.2.3 Desplegar las variables **nodo** y **token**.
- 8.3 Si la variable **nodo** es cero y la variable **token** es mayor o igual a 1000:
 - 8.3.1 Salir del ciclo.
- 8.4 Invocar el método `salida.writeLong(token)`.

Notar que cada nodo abre un puerto diferente debido a que los cuatro nodos ejecutan en la misma computadora.

Para ejecutar el programa en cada nodo se debe pasar como parámetros el número del nodo y la IP del siguiente nodo en el anillo (en este caso localhost).

Se deberá subir a la plataforma un archivo PDF que incluya: portada, captura de pantalla de la compilación del programa, captura de pantalla de la ejecución del programa en cada ventana (capturar la pantalla cuando termina el programa), captura de pantalla correspondiente a la creación de los repositorios (keystore) del cliente y el servidor, y conclusiones.

También se deberá subir a la plataforma el código fuente del programa desarrollado (como texto no como imagen).

De acuerdo a los "Lineamientos del curso" las capturas de pantalla deberán ser completas y legibles. El único formato de compresión admitido es ZIP.

Valor de la tarea: 30% (2.1 puntos de la primera evaluación parcial)

2. Pruebas de escritorio

Primero se crearon los certificados correspondientes, suando los comandos vistos en clase:

```
C:\WINDOWS\system32\cmd.exe
C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 1234567
¿Cuáles son su nombre y su apellido?
[Unknown]: Eduardo Castro
¿Cuál es el nombre de su unidad de organización?
[Unknown]: ESCOM
¿Cuál es el nombre de su organización?
[Unknown]: IPN
¿Cuál es el nombre de su ciudad o localidad?
[Unknown]: CDMX
¿Cuál es el nombre de su estado o provincia?
[Unknown]: CDMX
¿Cuál es el código de país de dos letras de la unidad?
[Unknown]: MX
¿Es correcto CN=Eduardo Castro, OU=ESCOM, O=IPN, L=CDMX, ST=CDMX, C=MX?
[no]: si

Introduzca la contraseña de clave para <certificado_servidor>:
(IntRO si es la misma contraseña que la del almacén de claves):
Volver a escribir la contraseña nueva:

Warning:
El almacén de claves JKS utiliza un formato privativo. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore keystore_servidor.jks -destkeystore keystore_servidor.jks -deststoretype pkcs12".

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor -rfc -file certificado_servidor.pem
Introduzca la contraseña del almacén de claves:
Certificado almacenado en el archivo <certificado_servidor.pem>

Warning:
El almacén de claves JKS utiliza un formato privativo. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore keystore_servidor.jks -destkeystore keystore_servidor.jks -deststoretype pkcs12".

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass 1234567
Propietario: CN=Eduardo Castro, OU=ESCOM, O=IPN, L=CDMX, ST=CDMX, C=MX
Emisor: CN=Eduardo Castro, OU=ESCOM, O=IPN, L=CDMX, ST=CDMX, C=MX
Número de serie: 7b646180
Válido desde: Thu Sep 09 14:18:32 CDT 2021 hasta: Wed Dec 08 13:18:32 CST 2021
Huellas digitales del certificado:
    SHA1: 6A:31:12:EE:73:BA:BA:0C:1B:7A:3B:D1:0D:7B:A5:84:8B:E8:3A:71
    SHA256: FC:00:E2:49:E3:63:71:4F:B:62:18:2E:7F:D6:89:33:19:89:B4:E0:D5:73:81:A2:09:B5:2D:27:E4:99:56:86:73
Nombre del algoritmo de firma: SHA256withRSA
Algoritmo de clave pública de asunto: Clave RSA de 2048 bits
Version: 3

Extensiones:
```

Compilación del programa en la terminal 0:

```
C:\WINDOWS\system32\cmd.exe
C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>javac TokenRing.java

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: F264-AD96

Directorio de C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2
09/09/2021 01:41 a. m.      <DIR>      .
09/09/2021 01:41 a. m.      <DIR>      ..
01/08/2021 07:36 p. m.      99,711 Reporte_Tareal.docx
09/09/2021 01:41 a. m.         768 TokenRing$worker.class
09/09/2021 01:41 a. m.       2,277 TokenRing.class
09/09/2021 01:39 a. m.       3,525 TokenRing.java
                4 archivos      106,281 bytes
                2 dirs      520,532,447,232 bytes libres

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>java TokenRing$worker.class
Error: Could not find or load main class TokenRing$worker.class
Caused by: java.lang.ClassNotFoundException: TokenRing$worker.class

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>java TokenRing
#Nodo + nextIP

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>java TokenRing 0 192.168.1.103
Nodo: 0
Token: 4
Nodo: 0
Token: 8
Nodo: 0
Token: 12
Nodo: 0
Token: 16
Nodo: 0
Token: 20
Nodo: 0
Token: 24
Nodo: 0
Token: 28
Nodo: 0
Token: 32
Nodo: 0
Token: 36
Nodo: 0
Token: 40
Nodo: 0
Token: 44
Nodo: 0
Token: 48
Nodo: 0
```

Ejecución del programa en la terminal 0 (servidor):

```
C:\WINDOWS\system32\cmd.exe
C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>java -Djavax.net.ssl.keyStore=keystore_servidor.jks -Djavax.net.ssl.keyStorePassword=1234567 TokenR
ing 0 192.168.1.103
Nodo: 0
Token: 4
Nodo: 0
Token: 8
Nodo: 0
Token: 12
Nodo: 0
Token: 16
Nodo: 0
Token: 20
Nodo: 0
Token: 24
Nodo: 0
Token: 28
Nodo: 0
Token: 32
Nodo: 0
Token: 36
Nodo: 0
Token: 40
Nodo: 0
Token: 44
Nodo: 0
Token: 48
Nodo: 0
Token: 52
Nodo: 0
Token: 56
Nodo: 0
Token: 60
Nodo: 0
Token: 64
Nodo: 0
Token: 68
Nodo: 0
Token: 72
Nodo: 0
Token: 76
Nodo: 0
Token: 80
Nodo: 0
Token: 84
Nodo: 0
Token: 88
Nodo: 0
Token: 92
Nodo: 0
```

```
C:\WINDOWS\system32\cmd.exe
Token: 908
Nodo: 0
Token: 912
Nodo: 0
Token: 916
Nodo: 0
Token: 920
Nodo: 0
Token: 924
Nodo: 0
Token: 928
Nodo: 0
Token: 932
Nodo: 0
Token: 936
Nodo: 0
Token: 940
Nodo: 0
Token: 944
Nodo: 0
Token: 948
Nodo: 0
Token: 952
Nodo: 0
Token: 956
Nodo: 0
Token: 960
Nodo: 0
Token: 964
Nodo: 0
Token: 968
Nodo: 0
Token: 972
Nodo: 0
Token: 976
Nodo: 0
Token: 980
Nodo: 0
Token: 984
Nodo: 0
Token: 988
Nodo: 0
Token: 992
Nodo: 0
Token: 996
Nodo: 0
Token: 1000
C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>
```


Ejecución del programa en el terminal 1 (cliente):

```
C:\WINDOWS\system32\cmd.exe
C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>java -Djavax.net.ssl.trustStore=keystore_cliente.jks -Djavax.net.ssl.trustStorePassword=1234567 Tok
enRing 1 192.168.1.103
Nodo: 1
Token: 1
Nodo: 1
Token: 5
Nodo: 1
Token: 9
Nodo: 1
Token: 13
Nodo: 1
Token: 17
Nodo: 1
Token: 21
Nodo: 1
Token: 25
Nodo: 1
Token: 29
Nodo: 1
Token: 33
Nodo: 1
Token: 37
Nodo: 1
Token: 41
Nodo: 1
Token: 45
Nodo: 1
Token: 49
Nodo: 1
Token: 53
Nodo: 1
Token: 57
Nodo: 1
Token: 61
Nodo: 1
Token: 65
Nodo: 1
Token: 69
Nodo: 1
Token: 73
Nodo: 1
Token: 77
Nodo: 1
Token: 81
Nodo: 1
Token: 85
Nodo: 1
Token: 89
Nodo: 1
```

```
C:\WINDOWS\system32\cmd.exe
Nodo: 1
Token: 925
Nodo: 1
Token: 929
Nodo: 1
Token: 933
Nodo: 1
Token: 937
Nodo: 1
Token: 941
Nodo: 1
Token: 945
Nodo: 1
Token: 949
Nodo: 1
Token: 953
Nodo: 1
Token: 957
Nodo: 1
Token: 961
Nodo: 1
Token: 965
Nodo: 1
Token: 969
Nodo: 1
Token: 973
Nodo: 1
Token: 977
Nodo: 1
Token: 981
Nodo: 1
Token: 985
Nodo: 1
Token: 989
Nodo: 1
Token: 993
Nodo: 1
Token: 997
Exception in thread "main" java.net.SocketException: Connection reset
    at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:323)
    at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)
    at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:803)
    at java.base/java.net.Socket$SocketInputStream.read(Socket.java:976)
    at java.base/java.io.DataInputStream.readFully(DataInputStream.java:201)
    at java.base/java.io.DataInputStream.readLong(DataInputStream.java:422)
    at TokenRing.main(TokenRing.java:93)
C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>
```

Ejecución del programa en terminal 2 (cliente):

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\georg>cd C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>java -Djavax.net.ssl.trustStore=keystore_cliente.jks -Djavax.net.ssl.trustStorePassword=1234567 Tok
enRing 2 192.168.1.103
Nodo: 2
Token: 2
Nodo: 2
Token: 6
Nodo: 2
Token: 10
Nodo: 2
Token: 14
Nodo: 2
Token: 18
Nodo: 2
Token: 22
Nodo: 2
Token: 26
Nodo: 2
Token: 30
Nodo: 2
Token: 34
Nodo: 2
Token: 38
Nodo: 2
Token: 42
Nodo: 2
Token: 46
Nodo: 2
Token: 50
Nodo: 2
Token: 54
Nodo: 2
Token: 58
Nodo: 2
Token: 62
Nodo: 2
Token: 66
Nodo: 2
Token: 70
Nodo: 2
Token: 74
Nodo: 2
Token: 78
Nodo: 2
Token: 82
Nodo: 2
```

```
C:\WINDOWS\system32\cmd.exe
Token: 922
Nodo: 2
Token: 926
Nodo: 2
Token: 930
Nodo: 2
Token: 934
Nodo: 2
Token: 938
Nodo: 2
Token: 942
Nodo: 2
Token: 946
Nodo: 2
Token: 950
Nodo: 2
Token: 954
Nodo: 2
Token: 958
Nodo: 2
Token: 962
Nodo: 2
Token: 966
Nodo: 2
Token: 970
Nodo: 2
Token: 974
Nodo: 2
Token: 978
Nodo: 2
Token: 982
Nodo: 2
Token: 986
Nodo: 2
Token: 990
Nodo: 2
Token: 994
Nodo: 2
Token: 998
Exception in thread "main" java.net.SocketException: Connection reset
    at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:323)
    at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)
    at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:803)
    at java.base/java.net.Socket$SocketInputStream.read(Socket.java:976)
    at java.base/java.io.DataInputStream.readFully(DataInputStream.java:201)
    at java.base/java.io.DataInputStream.readLong(DataInputStream.java:422)
    at TokenRing.main(TokenRing.java:93)

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>
```

Ejecución del programa en terminal 3 (cliente):

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\georg>cd C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>java -Djavax.net.ssl.trustStore=keystore_cliente.jks -Djavax.net.ssl.trustStorePassword=1234567 Tok
enRing 3 192.168.1.103
Nodo: 3
Token: 3
Nodo: 3
Token: 7
Nodo: 3
Token: 11
Nodo: 3
Token: 15
Nodo: 3
Token: 19
Nodo: 3
Token: 23
Nodo: 3
Token: 27
Nodo: 3
Token: 31
Nodo: 3
Token: 35
Nodo: 3
Token: 39
Nodo: 3
Token: 43
Nodo: 3
Token: 47
Nodo: 3
Token: 51
Nodo: 3
Token: 55
Nodo: 3
Token: 59
Nodo: 3
Token: 63
Nodo: 3
Token: 67
Nodo: 3
Token: 71
Nodo: 3
Token: 75
Nodo: 3
Token: 79
Nodo: 3
Token: 83
Nodo: 3
```

```
C:\WINDOWS\system32\cmd.exe
Nodo: 3
Token: 923
Nodo: 3
Token: 927
Nodo: 3
Token: 931
Nodo: 3
Token: 935
Nodo: 3
Token: 939
Nodo: 3
Token: 943
Nodo: 3
Token: 947
Nodo: 3
Token: 951
Nodo: 3
Token: 955
Nodo: 3
Token: 959
Nodo: 3
Token: 963
Nodo: 3
Token: 967
Nodo: 3
Token: 971
Nodo: 3
Token: 975
Nodo: 3
Token: 979
Nodo: 3
Token: 983
Nodo: 3
Token: 987
Nodo: 3
Token: 991
Nodo: 3
Token: 995
Nodo: 3
Token: 999
Exception in thread "main" java.net.SocketException: Connection reset
    at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:323)
    at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)
    at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:803)
    at java.base/java.net.Socket$SocketInputStream.read(Socket.java:976)
    at java.base/java.io.DataInputStream.readFully(DataInputStream.java:201)
    at java.base/java.io.DataInputStream.readLong(DataInputStream.java:422)
    at TokenRing.main(TokenRing.java:93)

C:\Users\georg\Desktop\ESCOM\8vo Semestre\Distribuidos\Tareas\Tarea2>
```

3. Conclusiones

Puesto que es un sistema distribuido, no importa el orden en que se ejecuten los nodos, siempre y cuando se ejecuten los 3 y no se repita ninguno para que funcione correctamente.

El problema llega cuando ejecutamos más de una vez algún cliente, porque el programa puede ejecutarse en cualquier nodo, simplemente se queda a la escucha de la ejecución del siguiente nodo iniciando el token cuando es la primera vez, se incrementa y lo manda al siguiente, este volverá a incrementar y lo manda al siguiente nodo, así se repite infinitamente en la topología de anillo. No funciona si no están conectadas entre sí, y cuando se ejecuta espera a que se conecte el siguiente nodo y no se sigue de largo, e iniciará, hasta que el último nodo se conecte al primero.