



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Tarea 2. Implementación de un token-ring

Unidad de aprendizaje: Desarrollo de Sistemas Distribuidos

Grupo: 4CV11

Alumno:
Sanchez Mendez Edmundo Josue

Profesor:
Pineda Guerrero Carlos

7 de septiembre de 2021

Índice

1. Introducción	2
2. Desarrollo	2
2.1. Compilación del código	3
2.2. Ejecución del programa	4
3. Conclusiones	6

1. Introducción

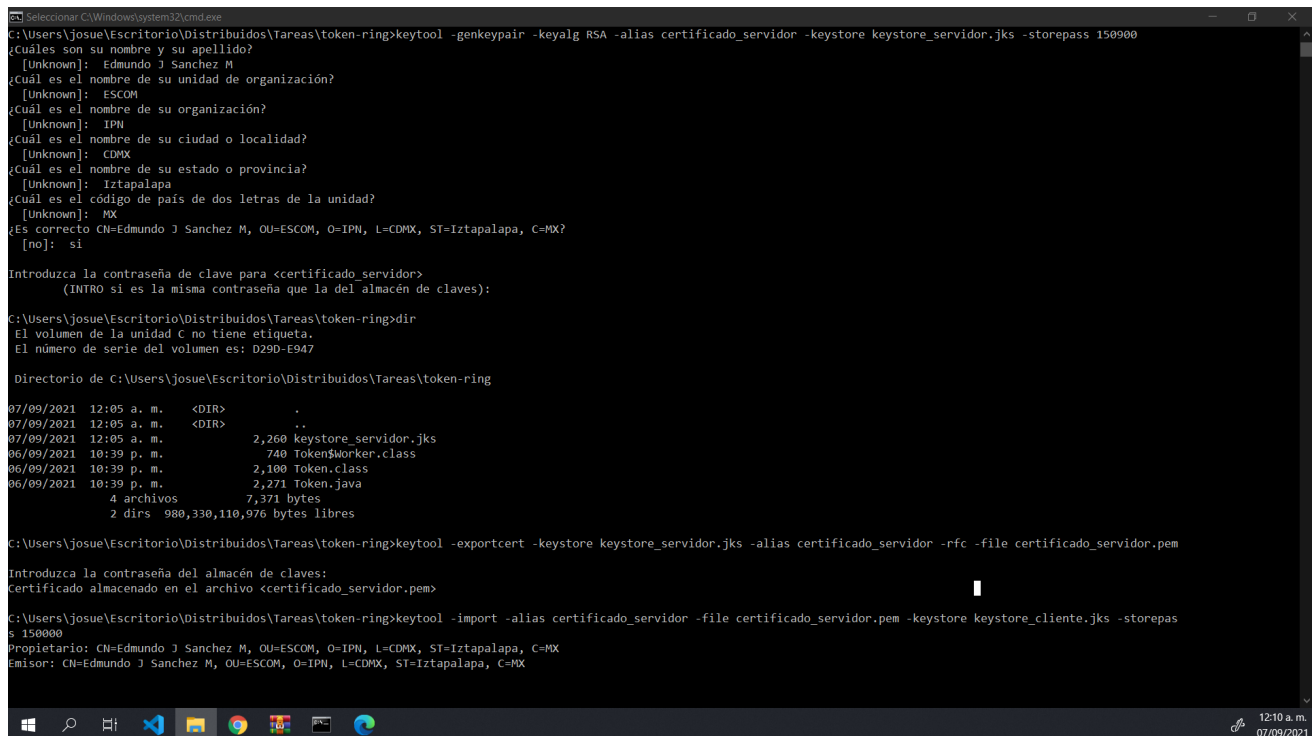
Mediante el desarrollo de un sistema distribuido el cual tendrá una topología lógica de anillo en donde interceden 4 nodos y la comunicación sera por medio de sockets seguros, sera transmitido un token el cual pasara por toda la topología e ira aumentando de valor en 1, el token será un número entero de 64 bits.

El funcionamiento del programa es el siguiente:

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.

2. Desarrollo

Una vez programado el sistema distribuido con base en los algoritmos establecidos en la tarea se procedió a implementar los sockets seguros, para lo cual fue necesario la creación de los keystore tanto para el cliente como para el servidor, la creación del keystore para el servidor la podemos ver en la figura 1 y la creación del keystore para el cliente la podemos ver en la figura 2.



```
C:\Users\Josue\Escritorio\Distribuidos\Tareas\token-ring>keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 150900
¿Cuáles son su nombre y su apellido?
[Unknown]: Edmundo J Sanchez M
¿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]: Iztapalapa
¿Cuál es el código de país de dos letras de la unidad?
[Unknown]: MX
¿Es correcto CN=Edmundo J Sanchez M, OU=ESCOM, O=IPN, L=CDMX, ST=Iztapalapa, C=MX?
[no]: si

Introduzca la contraseña de clave para <certificado_servidor>
(INTRO si es la misma contraseña que la del almacén de claves):

C:\Users\Josue\Escritorio\Distribuidos\Tareas\token-ring>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: D29D-E947

Directorio de C:\Users\Josue\Escritorio\Distribuidos\Tareas\token-ring
07/09/2021 12:05 a. m. <DIR> .
07/09/2021 12:05 a. m. <DIR> ..
07/09/2021 12:05 a. m. 2,268 keystore_servidor.jks
06/09/2021 10:39 p. m. 740 Token$Monger.class
06/09/2021 10:39 p. m. 2,100 Token.class
06/09/2021 10:39 p. m. 2,271 Token.java
4 archivos 7,371 bytes
2 dirs 980,330,110,976 bytes libres

C:\Users\Josue\Escritorio\Distribuidos\Tareas\token-ring>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>

C:\Users\Josue\Escritorio\Distribuidos\Tareas\token-ring>keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass 150900
Propietario: CN=Edmundo J Sanchez M, OU=ESCOM, O=IPN, L=CDMX, ST=Iztapalapa, C=MX
Emisor: CN=Edmundo J Sanchez M, OU=ESCOM, O=IPN, L=CDMX, ST=Iztapalapa, C=MX
```

Figura 1: Creación del keystore para el servidor.

```

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass s 150000
Propietario: CN=Edmundo J Sanchez M, OU=ESCOM, O=IPN, L=CDMX, ST=Iztapalapa, C=MX
Emisor: CN=Edmundo J Sanchez M, OU=ESCOM, O=IPN, L=CDMX, ST=Iztapalapa, C=MX
Número de serie: 22adfedf
Válido desde: Tue Sep 07 00:05:38 CDT 2021 hasta: Sun Dec 05 23:05:38 CST 2021
Huellas digitales del Certificado:
MD5: F7:D0:02:28:51:FF:B0:61:89:F2:7D:CB:42:50:C5:AC
SHA1: 40:61:AE:08:CC:38:39:A3:06:20:3E:11:26:B9:C3:21:22:64:EA:B0
SHA256: 9C:BA:02:7B:C5:98:2E:41:9E:79:13:D0:40:17:10:64:33:1E:BA:F1:6A:3F:A0:FC:DC:01:33:AE:84:43:38:28
Nombre del Algoritmo de Firma: SHA256withRSA
Versión: 3

Extensiones:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 60 69 18 24 84 C1 50 AC 1D 5D A7 1F F7 B6 26 7F 11...P...>...<...
0010: 3C 19 1D 01
]
]

¿Confiar en este certificado? [no]: si
Se ha agregado el certificado al almacén de claves

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: D29D-E947

Directorio de C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring
07/09/2021 12:08 a. m. <DIR> .
07/09/2021 12:08 a. m. <DIR> ..
07/09/2021 12:06 a. m. 1,286 certificado_servidor.pem
07/09/2021 12:08 a. m. 970 keystore_cliente.jks
07/09/2021 12:05 a. m. 2,260 keystore_servidor.jks
06/09/2021 10:39 p. m. 740 Token$worker.class
06/09/2021 10:39 p. m. 2,100 Token.class
06/09/2021 10:39 p. m. 2,271 Token.java
6 archivos 9,627 bytes
2 dirs 980,330,967,040 bytes libres

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>

```

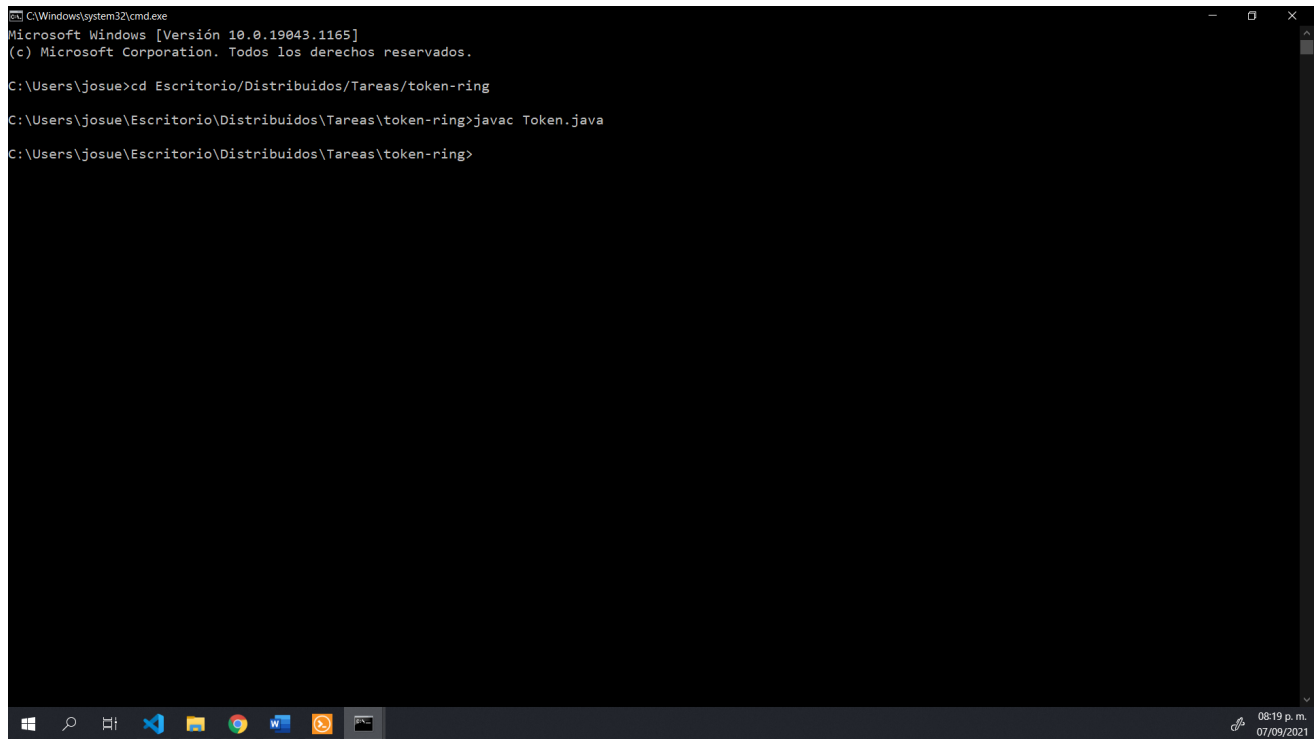
Figura 2: Creación del keystore para el cliente.

Una vez creado los keystore se procede a implementar lo sockets, empezamos a colocar propiedades del sistema los cuales contienen los archivos con las keystore del cliente y del servidor y las contraseñas correspondientes, todo lo anterior en las líneas 40-43, después en la línea 49 creamos la variable cliente de tipo `SSLConnectionFactory` y asignar lo siguiente (`SSLConnectionFactory.getDefault()`); para después remplazar el paso 5.1.1 y asignar a la variable conexión la siguiente línea de código cliente.`createSocket(ip, 20000 + (nodo + 1) % 4)`; lo cual nos permite crear un socket seguro para el cliente, después para nuestro servidor al inicio del algoritmo 1 justo después crear el bloque try declaramos la variable socket_ factory de tipo `SSLServerConnectionFactory` y asignamos el valor siguiente (`SSLServerConnectionFactory.getDefault()`); para después remplazar el punto 1.2 del algoritmo 1 y signar a la variable servidor lo siguiente `socket_ factory.createServerSocket(20000 + nodo)`;

Para finalizar se debe si o si cerrar la variable salida, entrada y conexión para que no se tenga error alguno, esto se tendrá que hacer justo después del ciclo del paso 8 del algoritmo 2.

2.1. Compilación del código

En la figura 3 podemos ver como la compilación de nuestro programa `Token.java` se hace de manera exitosa y sin ningún error alguno, esto desde el Símbolo del sistema (CMD) de Windows 10.



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

C:\Users\josue>cd Escritorio/Distribuidos/Tareas/token-ring
C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>javac Token.java
C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>
```

Figura 3: Compilación del código por medio de CMD.

2.2. Ejecución del programa

Una vez compilado se ejecuta nuestro programa para lo cual necesitamos abrir 4 consolas y ejecutar los nodos 0, 1, 2, 3 para cumplir con el sistema distribuido, en la figura 4 podemos ver el inicio de estas 4 consolas al ejecutar el programa:

```

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 0 localhost
nodo = 0 token = 5
nodo = 0 token = 9
nodo = 0 token = 13
nodo = 0 token = 17
nodo = 0 token = 21
nodo = 0 token = 25
nodo = 0 token = 29
nodo = 0 token = 33
nodo = 0 token = 37
nodo = 0 token = 41
nodo = 0 token = 45
nodo = 0 token = 49
nodo = 0 token = 53
nodo = 0 token = 57
nodo = 0 token = 61
nodo = 0 token = 65
nodo = 0 token = 69
nodo = 0 token = 73
nodo = 0 token = 77

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 1 localhost
nodo = 1 token = 2
nodo = 1 token = 6
nodo = 1 token = 10
nodo = 1 token = 14
nodo = 1 token = 18
nodo = 1 token = 22
nodo = 1 token = 26
nodo = 1 token = 30
nodo = 1 token = 34
nodo = 1 token = 38
nodo = 1 token = 42
nodo = 1 token = 46
nodo = 1 token = 50
nodo = 1 token = 54
nodo = 1 token = 58
nodo = 1 token = 62
nodo = 1 token = 66
nodo = 1 token = 70
nodo = 1 token = 74

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 2 localhost
nodo = 2 token = 3
nodo = 2 token = 7
nodo = 2 token = 11
nodo = 2 token = 15
nodo = 2 token = 19
nodo = 2 token = 23
nodo = 2 token = 27
nodo = 2 token = 31
nodo = 2 token = 35
nodo = 2 token = 39
nodo = 2 token = 43
nodo = 2 token = 47
nodo = 2 token = 51
nodo = 2 token = 55
nodo = 2 token = 59
nodo = 2 token = 63
nodo = 2 token = 67
nodo = 2 token = 71
nodo = 2 token = 75

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 3 localhost
nodo = 3 token = 4
nodo = 3 token = 8
nodo = 3 token = 12
nodo = 3 token = 16
nodo = 3 token = 20
nodo = 3 token = 24
nodo = 3 token = 28
nodo = 3 token = 32
nodo = 3 token = 36
nodo = 3 token = 40
nodo = 3 token = 44
nodo = 3 token = 48
nodo = 3 token = 52
nodo = 3 token = 56
nodo = 3 token = 60
nodo = 3 token = 64
nodo = 3 token = 68
nodo = 3 token = 72
nodo = 3 token = 76

```

Figura 4: Consolas justo después de ejecutar el programa.

Al finalizar se nos muestra lo que podemos ver en las figuras 5 y 6, se decidió tomar una screenshot mas para visualizar de una mejor manera el resultado de los nodos 2 y 3:

```

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 0 localhost
nodo = 0 token = 929
nodo = 0 token = 933
nodo = 0 token = 937
nodo = 0 token = 941
nodo = 0 token = 945
nodo = 0 token = 949
nodo = 0 token = 953
nodo = 0 token = 957
nodo = 0 token = 961
nodo = 0 token = 965
nodo = 0 token = 969
nodo = 0 token = 973
nodo = 0 token = 977
nodo = 0 token = 981
nodo = 0 token = 985
nodo = 0 token = 989
nodo = 0 token = 993
nodo = 0 token = 997
nodo = 0 token = 1001

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 1 localhost
nodo = 1 token = 942
nodo = 1 token = 946
nodo = 1 token = 950
nodo = 1 token = 954
nodo = 1 token = 958
nodo = 1 token = 962
nodo = 1 token = 966
nodo = 1 token = 970
nodo = 1 token = 974
nodo = 1 token = 978
nodo = 1 token = 982
nodo = 1 token = 986
nodo = 1 token = 990
nodo = 1 token = 994
nodo = 1 token = 998
Exception in thread "main" java.io.EOFException
    at java.io.DataInputStream.readFully(Unknown Source)
    at java.io.DataInputStream.readLong(Unknown Source)
    at Token.main(Token.java:72)

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 2 localhost
nodo = 2 token = 987
nodo = 2 token = 991
nodo = 2 token = 995
nodo = 2 token = 999
Exception in thread "main" javax.net.ssl.SSLException: java.net.SocketException: Connection reset
    at sun.security.ssl.Alert.createSSLException(Unknown Source)
    at sun.security.ssl.TransportContext.fatal(Unknown Source)
    at sun.security.ssl.TransportContext.fatal(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.handleException(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
    at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
    at java.io.DataInputStream.readFully(Unknown Source)
    at java.io.DataInputStream.readLong(Unknown Source)
    at Token.main(Token.java:72)
    Suppressed: java.net.SocketException: Connection reset by peer: socket write error
        at java.net.SocketOutputStream.socketWrite0(Native Method)
        at java.net.SocketOutputStream.socketWrite(Unknown Source)
        at java.net.SocketOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.handleException(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
        at java.io.DataInputStream.readFully(Unknown Source)
        at java.io.DataInputStream.readLong(Unknown Source)
        at Token.main(Token.java:72)
    Suppressed: java.net.SocketException: Connection reset by peer: socket write error
        at java.net.SocketOutputStream.socketWrite0(Native Method)
        at java.net.SocketOutputStream.socketWrite(Unknown Source)
        at java.net.SocketOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.handleException(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
        at java.io.DataInputStream.readFully(Unknown Source)
        at java.io.DataInputStream.readLong(Unknown Source)
        at Token.main(Token.java:72)

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>java Token 3 localhost
nodo = 3 token = 988
nodo = 3 token = 992
nodo = 3 token = 996
nodo = 3 token = 1000
Exception in thread "main" javax.net.ssl.SSLException: java.net.SocketException: Connection reset
    at sun.security.ssl.Alert.createSSLException(Unknown Source)
    at sun.security.ssl.TransportContext.fatal(Unknown Source)
    at sun.security.ssl.TransportContext.fatal(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.handleException(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
    at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
    at java.io.DataInputStream.readFully(Unknown Source)
    at java.io.DataInputStream.readLong(Unknown Source)
    at Token.main(Token.java:72)
    Suppressed: java.net.SocketException: Connection reset by peer: socket write error
        at java.net.SocketOutputStream.socketWrite0(Native Method)
        at java.net.SocketOutputStream.socketWrite(Unknown Source)
        at java.net.SocketOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.handleException(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
        at java.io.DataInputStream.readFully(Unknown Source)
        at java.io.DataInputStream.readLong(Unknown Source)
        at Token.main(Token.java:72)
    Suppressed: java.net.SocketException: Connection reset by peer: socket write error
        at java.net.SocketOutputStream.socketWrite0(Native Method)
        at java.net.SocketOutputStream.socketWrite(Unknown Source)
        at java.net.SocketOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppOutputStream.write(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.handleException(Unknown Source)
        at sun.security.ssl.SSLSocketImpl.access$400(Unknown Source)
        at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
        at java.io.DataInputStream.readFully(Unknown Source)
        at java.io.DataInputStream.readLong(Unknown Source)
        at Token.main(Token.java:72)

```

Figura 5: Consolas un poco antes del fin de lo ejecución del programa.

```

C:\Windows\system32\cmd.exe
nodo = 0 token = 929
nodo = 0 token = 933
nodo = 0 token = 937
nodo = 0 token = 941
nodo = 0 token = 945
nodo = 0 token = 949
nodo = 0 token = 953
nodo = 0 token = 957
nodo = 0 token = 961
nodo = 0 token = 965
nodo = 0 token = 969
nodo = 0 token = 973
nodo = 0 token = 977
nodo = 0 token = 981
nodo = 0 token = 985
nodo = 0 token = 989
nodo = 0 token = 993
nodo = 0 token = 997
nodo = 0 token = 1001

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>

C:\Windows\system32\cmd.exe
nodo = 1 token = 942
nodo = 1 token = 946
nodo = 1 token = 950
nodo = 1 token = 954
nodo = 1 token = 958
nodo = 1 token = 962
nodo = 1 token = 966
nodo = 1 token = 970
nodo = 1 token = 974
nodo = 1 token = 978
nodo = 1 token = 982
nodo = 1 token = 986
nodo = 1 token = 990
nodo = 1 token = 994
nodo = 1 token = 998
Exception in thread "main" java.io.EOFException
    at java.io.DataInputStream.readFully(Unknown Source)
    at java.io.DataInputStream.readLong(Unknown Source)
    at Token.main(Token.java:72)

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>

C:\Windows\system32\cmd.exe
at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
at java.io.DataInputStream.readFully(Unknown Source)
at java.io.DataInputStream.readLong(Unknown Source)
at Token.main(Token.java:72)
Suppressed: java.net.SocketException: Connection reset by peer: socket write error
    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(Unknown Source)
    at java.net.SocketOutputStream.write(Unknown Source)
    at sun.security.ssl.SSLSocketOutputRecord.encodeAlert(Unknown Source)
    ... 9 more
Caused by: java.net.SocketException: Connection reset
    at java.net.SocketInputStream.read(Unknown Source)
    at java.net.SocketInputStream.read(Unknown Source)
    at sun.security.ssl.SSLSocketInputRecord.read(Unknown Source)
    at sun.security.ssl.SSLSocketInputRecord.readHeader(Unknown Source)
    at sun.security.ssl.SSLSocketInputRecord.bytesInCompletePacket(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.readApplicationRecord(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.access$300(Unknown Source)
    ... 4 more

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>

C:\Windows\system32\cmd.exe
at sun.security.ssl.SSLSocketImpl$AppInputStream.read(Unknown Source)
at java.io.DataInputStream.readFully(Unknown Source)
at java.io.DataInputStream.readLong(Unknown Source)
at Token.main(Token.java:72)
Suppressed: java.net.SocketException: Connection reset by peer: socket write error
    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(Unknown Source)
    at java.net.SocketOutputStream.write(Unknown Source)
    at sun.security.ssl.SSLSocketOutputRecord.encodeAlert(Unknown Source)
    ... 9 more
Caused by: java.net.SocketException: Connection reset
    at java.net.SocketInputStream.read(Unknown Source)
    at java.net.SocketInputStream.read(Unknown Source)
    at sun.security.ssl.SSLSocketInputRecord.read(Unknown Source)
    at sun.security.ssl.SSLSocketInputRecord.readHeader(Unknown Source)
    at sun.security.ssl.SSLSocketInputRecord.bytesInCompletePacket(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.readApplicationRecord(Unknown Source)
    at sun.security.ssl.SSLSocketImpl.access$300(Unknown Source)
    ... 4 more

C:\Users\josue\Escritorio\Distribuidos\Tareas\token-ring>
  
```

Figura 6: Consolas al final de ejecución de nuestro programa.

Cuando el token se encuentre en el nodo 0 con el valor 1001 el programa terminara pero en los demás nodos se nos mostrara los errores mostrados en la figura 6, esto debido a que el nodo 0 se cierra de forma abrupta.

3. Conclusiones

El desarrollo del código de la práctica fue muy sencillo, sin embargo, añadir los sockets seguros fue algo complicado ya que en un inicio no cerraba las variables de entrada, salida y conexión algo que es sin duda alguna necesario ya que si no lo hacemos nos genera error de handshake y ese error solo se soluciona cerrando las variables anteriormente mencionadas. Mencionar que los nodos se quedaban en espera, hasta que se ejecutaran los 4 nodos totales. Los 4 nodos muestran, como salida, el número de nodo y el valor del token, mencionar que los nodos de número impar el valor del token mostrado en pantalla serán impares también y por otro lado los nodos pares mostraran valores de token pares. Al finalizar el nodo 0, el resto de nodos terminan de forma abrupta.

Finalmente se anexan en el zip el archivo Token.java y los .jks en donde están almacenados los keystore para el cliente y el servidor.