

**Universidad Nacional
de Costa Rica**

Escuela de Informática

Sistemas Distribuidos

Proyecto 1 BINGO

Profesor: Armando Arce
Estudiante: Sergio Villegas

Ciclo II

Año: 2017

Introducción

El siguiente documento muestra las diferentes etapas que se llevaron a cabo para la realización del juego bingo a cartón lleno. Explica la descripción del problema, así como los diferentes componentes que forman el sistema de bingo. Además se podrán encontrar algunos pantallazos del sistema en ejecución

Descripción Del Problema

Ver anexo 1

Definición de Estructuras de Datos

Las estructuras utilizadas en los diferentes procesos son muy sencillas. Se trata de arreglos utilizados para el manejo de los cartones así como el de los números que han salido ya que estos no deben repetirse. La siguiente tabla muestra las estructuras de datos utilizadas en el proceso Jugador.lua

Estructura	Nombre en código	Notas
Arreglo de arreglos	miCarton	<ul style="list-style-type: none">• Se generan 5 arreglos que representan las columnas.• Se agregan al arreglo miCarton y se le aplica la traspuesta
Arreglo	<ul style="list-style-type: none">• column1• column2• column3• column4• column5	<ul style="list-style-type: none">• Se llenan con números aleatorios• Rango de numeros dependen de la columna• se agregan a miCarton para crear el carton final

La siguiente tabla muestra las estructuras utilizadas en el proceso emisor.lua

Estructura	Nombre en código	Notas
Arreglo	numerosCantados	<ul style="list-style-type: none">se llenan con números aleatorios del 1 al 75 para poder validar si el número ya salió o no en el bingo

En el proceso difusor no se utiliza ningún tipo de arreglo.

Componentes Principales del Sistema

El sistema se compone de 3 partes, el emisor, difusor y jugador. El emisor se encarga de escoger las bolitas y cantar los números, el difusor se encarga de distribuir la información del emisor a los jugadores conectados. lo jugadores solo reciben los números para marcar sus cartones.

Partes del Jugador.

La siguiente tabla muestra las partes principales del proceso jugador:

Parte	Linea	Explicación
socket suscriptor	9-10	Se declara este socket para hacer posible la comunicación con el servidor. socket de tipo SUB
socket socketGane	13-14	Se declara el socket para comunicar si el jugador ha ganado. Sirve para enviar la confirmación al difusor
generateBoard	25-40	Genera un cartón totalmente con números aleatorios del 1 al 75 según las reglas del bingo tradicional
imprimeTablero	90-99	recorre el arreglo miCarton y lo imprime en forma de tablero en consola

marcarCarton	102-114	recibe un número y verifica si existe en el arreglo miCarton. Si existe lo marca(pone un 0 en el campo)
cartonLleno	116-134	verifica si el el carton jugado está lleno. retorna true si todos los números del arreglo miCarton son 0(cero)
script de ejecución	143-172	recibe los numeros desde el difusor, y los procesa para verificar los cartones y marcarlos.

Partes del Emisor.

La siguiente tabla muestra las partes principales del proceso emisor:

Parte	Línea	Explicación
socket publisher	7-8	tipo pub. Envía los números a los suscriptores
cantarNumerno	15-32	Genera un número aleatorio entre 1 y 75 y los guarda en el arreglo númerosCantados
esNumeroCantado	34-46	Verifica si el número cantado existe en el arreglo numerosCantados
script de ejecución	51-74	genera los numeros y los envía al difusor por medio del socket

Partes del Jugador.

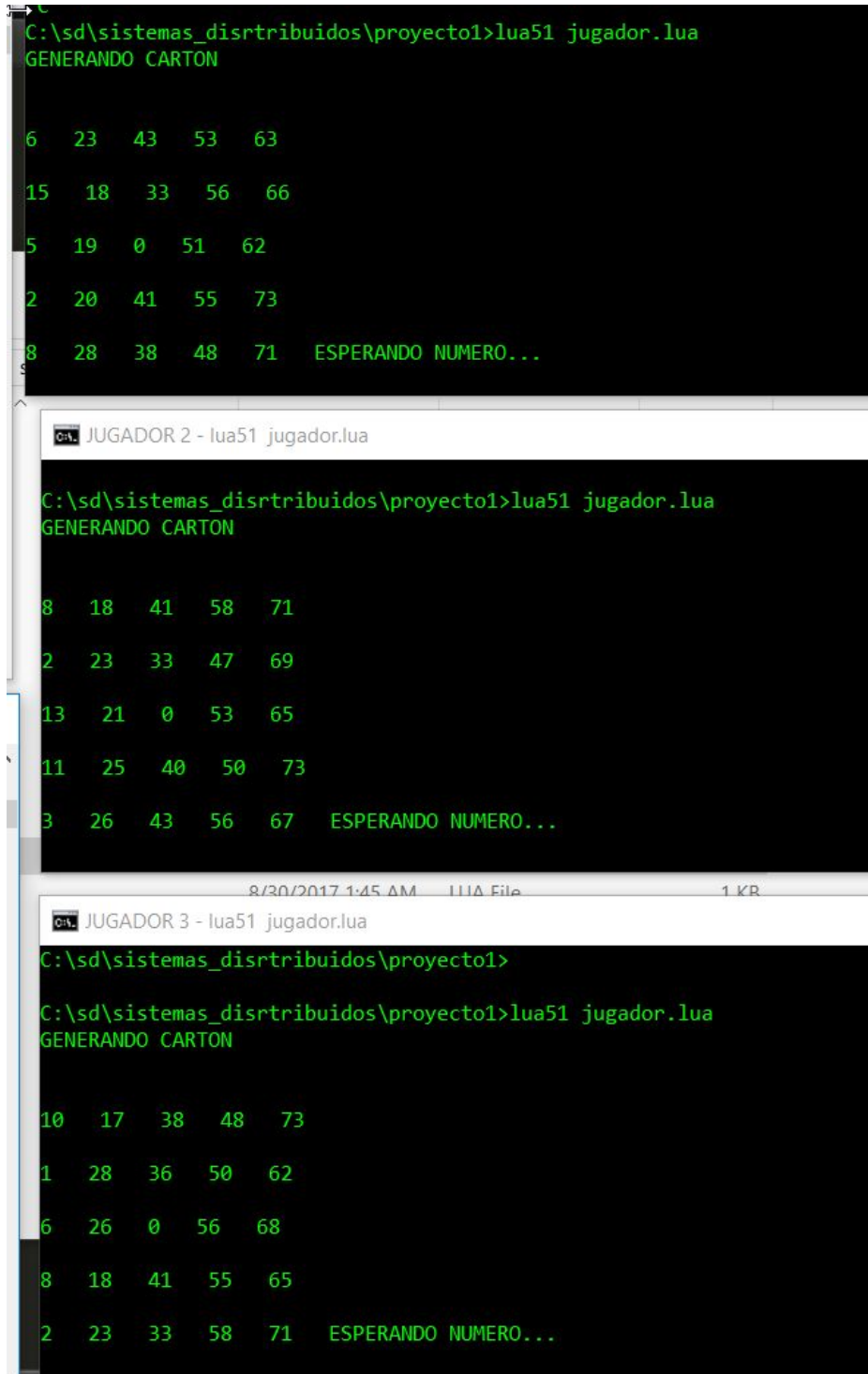
La siguiente tabla muestra las partes principales del proceso jugador:

Parte	Línea	Explicación
socket frontend	7-8	para suscribirse al proceso emisor. Tipo SUB
socket backend	10-11	tipo PUB, para enviar los números a los jugadores
socket socketGano	14-15	recibe alerta de gane de alguno de los jugadores.
script de ejecución	23-51	recibe mensajes del emisor y los distribuye a los jugadores. también recibe alerta de algún jugador cuando este canta bingo.

Pruebas de Ejecución

Las siguientes imágenes muestran el sistema en ejecución

1. Estado inicial de los cartones:



The image displays three separate terminal windows, each showing the execution of a Lua script named 'jugador.lua'. Each window represents a different player (Jugador 1, 2, and 3) and shows the initial state of their cards after the command 'GENERANDO CARTON' has been executed. The cards are displayed as a 5x5 grid of numbers. The prompt 'ESPERANDO NUMERO...' indicates that the program is waiting for user input.

JUGADOR 1 - lua51 jugador.lua

```
C:\sd\sistemas_disrtribuidos\proyecto1>lua51 jugador.lua
GENERANDO CARTON

6   23   43   53   63
15  18   33   56   66
5   19   0    51   62
2   20   41   55   73
8   28   38   48   71  ESPERANDO NUMERO...
```

JUGADOR 2 - lua51 jugador.lua

```
C:\sd\sistemas_disrtribuidos\proyecto1>lua51 jugador.lua
GENERANDO CARTON

8   18   41   58   71
2   23   33   47   69
13  21   0    53   65
11  25   40   50   73
3   26   43   56   67  ESPERANDO NUMERO...
```

JUGADOR 3 - lua51 jugador.lua

```
C:\sd\sistemas_disrtribuidos\proyecto1>
C:\sd\sistemas_disrtribuidos\proyecto1>lua51 jugador.lua
GENERANDO CARTON

10  17   38   48   73
1   28   36   50   62
6   26   0    56   68
8   18   41   55   65
2   23   33   58   71  ESPERANDO NUMERO...
```

a.

2. Bingo en Ejecución

EMISOR - lua51 emisor.lua

```
C:\sd\sistemas_disrtibuidos\proyecto1>lua51 emisor.lua
2 -- SENDING: 70
3 -- SENDING: 39
4 -- SENDING: 49
5 -- SENDING: 52
6 -- SENDING: 45
7 -- SENDING: 8
8 -- SENDING: 40
```

JUGADOR 1 - lua51 jugador.lua

```
EL NUMERO EN JUEGO ES: 40
3 17 41 60 0
11 23 37 46 75
5 30 0 0 61
15 22 32 59 64
2 26 0 55 67
```

JUGADOR 2 - lua51 jugador.lua

```
EL NUMERO EN JUEGO ES: 40
5 23 0 60 0
11 30 41 46 75
15 22 0 0 61
2 26 37 59 64
0 24 0 55 67
```

JUGADOR 3 - lua51 jugador.lua

```
EL NUMERO EN JUEGO ES: 40
2 26 0 55 67
0 22 31 60 73
15 24 0 46 75
7 17 44 0 62
11 30 0 0 64
```

DIFUSOR - lua51 difusor.lua

```
C:\sd\sistemas_disrtibuidos\proyecto1>lua51 difusor.lua
PROXY GETS: 70
PROXY SENDS: 70
UUPROXY
PROXY GETS: 39
PROXY SENDS: 39
UUPROXY
PROXY GETS: 49
PROXY SENDS: 49
UUPROXY
PROXY GETS: 52
PROXY SENDS: 52
UUPROXY
PROXY GETS: 45
PROXY SENDS: 45
UUPROXY
PROXY GETS: 8
PROXY SENDS: 8
UUPROXY
PROXY GETS: 40
PROXY SENDS: 40
UUPROXY
```

a.

3. Estado Final de los cartones

```
C:\> JUGADOR 1

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 38 0 0

OTRO CARTON GANO
*_*_*_*_*_*_*_*_*_*JUEGO TERMINADO*_*_*_*_*_*_*_*_*_*
C:\sd\sistemas_disrtribuidos\proyecto1>
```

```
C:\> JUGADOR 2

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0 BINGO

C:\sd\sistemas_disrtribuidos\proyecto1>
```

8/30/2017 1:45 AM LIA File

```
C:\> JUGADOR 3

0 0 38 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

OTRO CARTON GANO
*_*_*_*_*_*_*_*_*_*JUEGO TERMINADO*_*_*_*_*_*_*_*_*_*
C:\sd\sistemas_disrtribuidos\proyecto1>
```

Conclusiones

- Se concluye que se necesita de un difusor que nos comple la funcion de proxy en un sistema de publicación suscripción
- Los sockets de tipo PUB solo funcionan para publicar, no pueden recibir mensajes
- Los sockets de tipo SUB solo funcionan para recibir mensajes, no se pueden utilizar para enviarlos.