



MULTICAST CONFIABLE BÁSICO



GRUPO: 4CM3

EQUIPO 3

INTEGRANTES:

- MAYA ROCHA LUIS EMMANUEL
- OSUNA BANDA ITZEL ARELY
- MIRANDA MOJICA ERICK
- BARBOSA PEÑA XAVIER MARISTIN

De la misma forma que los mensajes UDP unicast no son confiables, tampoco lo son los mensajes UDP multicast. En esta práctica agregaremos un modelo muy básico de confiabilidad a los mensajes multicast.

Ejercicio 1:

A la clase SocketMulticast agregue el método:

int enviaConfiable(PaqueteDatagrama & p, unsigned char ttl, int num_receptores);

El cual se encarga de hacer el envío confiable de un mensaje multicast. Este método es similar al método envia(). Se sugiere enviar una estructura que contenga los datos y el identificador de mensaje que se explica en el libro de Tanenbaum. Este método debe devolver -1 en el caso de que el mensaje no les haya llegado a todos los receptores.

Implementación de método enviaConfiable:

```
int SocketMulticast::reliableSend(DatagramPacket & p, unsigned char ttl, int num_receptores){
    int receptores_recibidos = 0;
    send(p, ttl);

    int ack[1];
    DatagramPacket paqAcuse = DatagramPacket((char*)ack, sizeof(int));

    while(receptores_recibidos < num_receptores){
        if(receiveTimeout(paqAcuse, 3, 0) < 0){
            unbind();
            std::cout << "Recibo de acuse no recibido"<< std::endl;
            return -1;
        }
        receptores_recibidos++;
        std::cout << "Acuse #" << receptores_recibidos << " de " << paqAcuse.getAddress() << std::endl;
    }
    return receptores_recibidos;
}
```

Agregue también el método:

int recibeConfiable(PaqueteDatagrama & p);

Que se debe ejecutar para recibir mensajes multicast que implementan un protocolo confiable.

Implementación de método recibeConfiable:

```
int SocketMulticast::reliableReceive(DatagramPacket & p, unsigned char ttl){
    int ans = receive(p);
    char res = 1;
    DatagramPacket dp(&res, 1, p.getAddress(), p.getPort());
    send(dp, ttl);
    return ans;
}
```

Ejercicio 2:

Para probar el funcionamiento de su programa vamos a simular una nano base de datos que solo almacena la cuenta en pesos de un cliente en la variable entera nbd, y cuyo valor inicial es cero. Por seguridad, esta nano base de datos se encuentra replicada en tres servidores ubicados en tres computadoras independientes. En una cuarta computadora vamos a ejecutar un cliente que recibe en la línea de comandos un entero n, y va a ejecutar ese número de depósitos de una cantidad aleatoria de pesos comprendida entre \$1 y \$9 sobre la cuenta.

Implementación del servidor:

```
#include <bits/stdc++.h>
using namespace std;

int main(int argc, char* argv[]) {
    try {
        int cont = 0;
        int deposit;
        SocketMulticast ms(atoi(argv[2]));
        ms.joinGroup(argv[1]);
        DatagramPacket packet((char*)&deposit, sizeof(int));
        //ms.receiveTimeout(packet, 3, 0);
        while (true){
            ms.reliableReceive(packet, atoi(argv[3]));
            cont += deposit;
            cout << "Message from " << packet.getAddress() << ':' << packet.getPort() << endl;
            cout << "Recibimos " << deposit << ", Saldo actual " << cont << endl;
        }
    } catch (string err) {
        cout << err << endl;
    }
}
```

Implementación del cliente:

```
#include <bits/stdc++.h>
using namespace std;

/* Generamos semilla */
std::mt19937_64 seed(std::chrono::steady_clock::now().time_since_epoch().count());

int random(int min, int max){
    return std::uniform_int_distribution<int>(min, max)(seed);
}

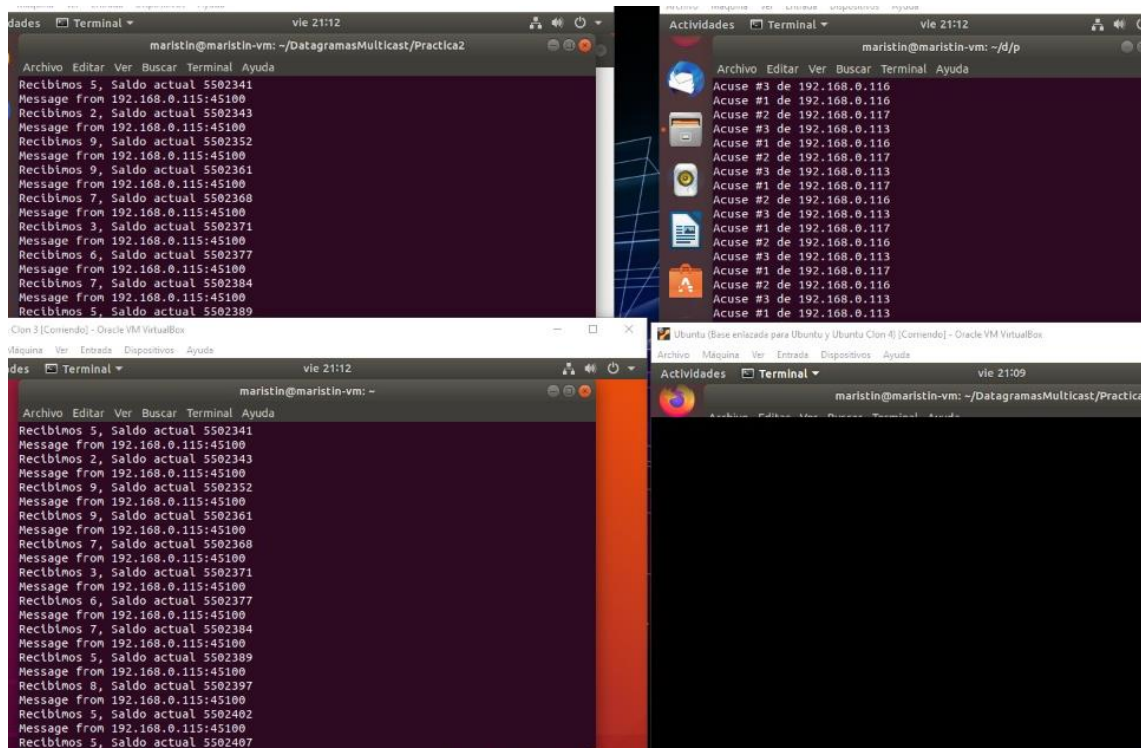
int main(int argc, char* argv[]){
    int n = atoi(argv[4]), deposit;
    DatagramPacket packet((char*) &deposit, sizeof(int), argv[1], atoi(argv[2]));
    SocketMulticast ms;
    for(int i = 0; i < n; i++){
        deposit = random(1, 9);
        try{
            ms.reliableSend(packet, atoi(argv[3]), 3);
        } catch (string err){
            cout << err << endl;
        }
    }
}
```

Para que la base de datos en los tres servidores se mantenga consistente en el monto, es necesario que los tres clientes realicen los depósitos mediante envíos multicast confiables hacia las tres bases de datos.

Pruébelo varias veces y con números n grandes para comprobar que la base de datos se mantiene consistente en los tres servidores.

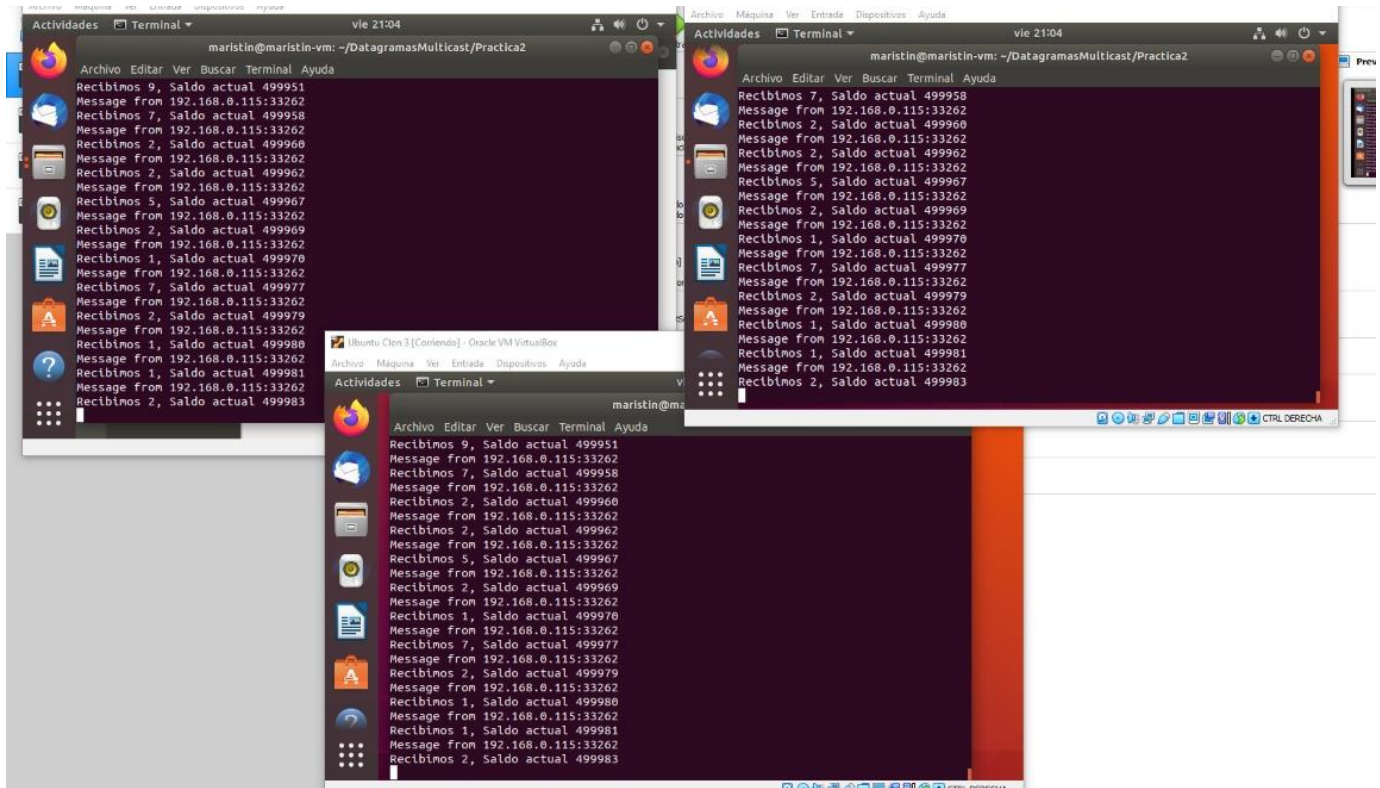
Pruebas:

Probamos con $n = 99\,999$, en ambas pruebas podemos observar la consistencia en el monto en cada servidor.



The image displays four terminal windows from a virtual machine named 'maristin@maristin-vm'. The top-left window shows a series of 'Recibimos' (Received) messages with 'Saldo actual' (Current balance) values ranging from 5502341 to 5502389. The top-right window shows a series of 'Acuse' (Acknowledgment) messages with values ranging from 192.168.0.116 to 192.168.0.113. The bottom-left window shows a series of 'Recibimos' messages with 'Saldo actual' values ranging from 5502341 to 5502407. The bottom-right window shows a series of 'Acuse' messages with values ranging from 192.168.0.116 to 192.168.0.113. The terminal windows are arranged in a 2x2 grid, with the top-left and bottom-left windows showing the 'Recibimos' messages and the top-right and bottom-right windows showing the 'Acuse' messages.

Probamos con $n = 999\ 999$



The image displays three terminal windows from a virtual machine named 'maristin-vm'. Each window shows a log of network messages and their corresponding 'Saldo actual' (current balance) values. The logs are as follows:

Top Left Terminal:

```
Recibimos 9, Saldo actual 499951
Message from 192.168.0.115:33262
Recibimos 7, Saldo actual 499958
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499960
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499962
Message from 192.168.0.115:33262
Recibimos 5, Saldo actual 499967
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499969
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499970
Message from 192.168.0.115:33262
Recibimos 7, Saldo actual 499977
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499979
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499980
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499981
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499983
```

Top Right Terminal:

```
Recibimos 7, Saldo actual 499958
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499960
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499962
Message from 192.168.0.115:33262
Recibimos 5, Saldo actual 499967
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499969
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499970
Message from 192.168.0.115:33262
Recibimos 7, Saldo actual 499977
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499979
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499980
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499981
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499983
```

Bottom Terminal:

```
Recibimos 9, Saldo actual 499951
Message from 192.168.0.115:33262
Recibimos 7, Saldo actual 499958
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499960
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499962
Message from 192.168.0.115:33262
Recibimos 5, Saldo actual 499967
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499969
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499970
Message from 192.168.0.115:33262
Recibimos 7, Saldo actual 499977
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499979
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499980
Message from 192.168.0.115:33262
Recibimos 1, Saldo actual 499981
Message from 192.168.0.115:33262
Recibimos 2, Saldo actual 499983
```