Name: Adarsh

Roll No: 106119001

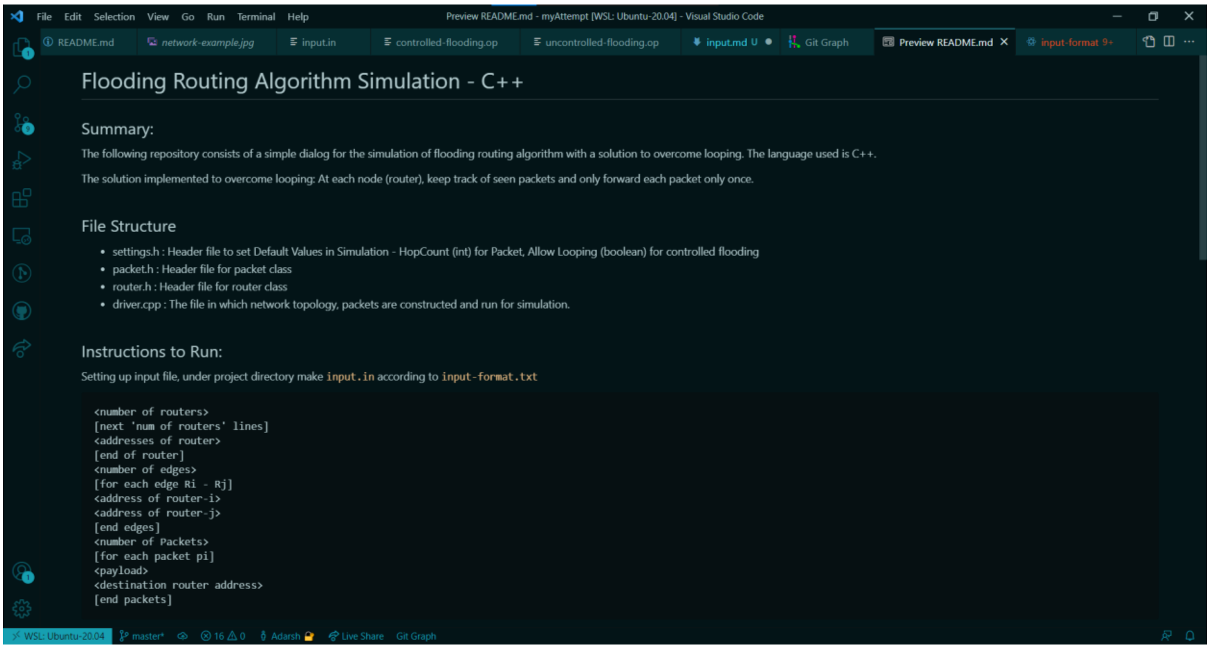
Networks Assignment – 2: Flooding Routing

The entire project has been uploaded to a Github repository, which can be accessed with the following link: https://github.com/9401adarsh/shiny-octo-spork

README.md:

Link: https://github.com/9401adarsh/shiny-octo-spork/blob/6b09e85a3083ba1802e1c2e825aa833fd2d17c31/README.md#L21

Screenshot of README.md:



Flooding Routing Algorithm Simulation - C++

Summary:

The following repository consists of a simple dialog for the simulation of flooding routing algorithm with a solution to overcome looping. The language used is C++.

The solution implemented to overcome looping: At each node (router), keep track of seen packets and only forward each packet only once.

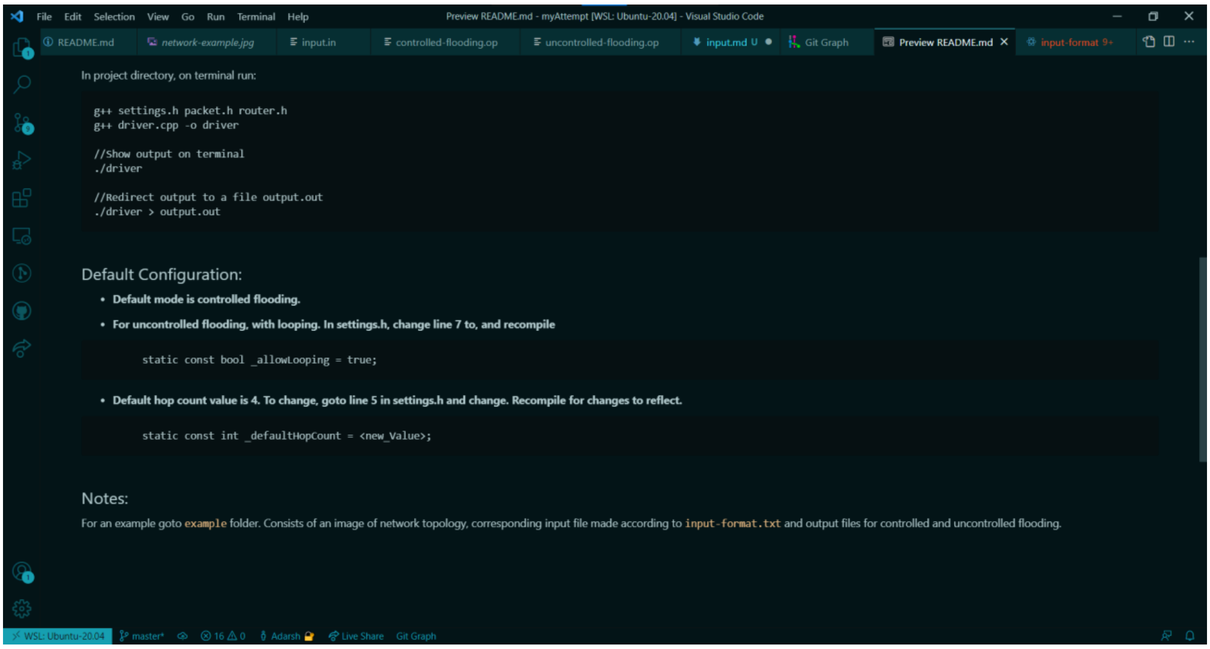
File Structure

- settings.h : Header file to set Default Values in Simulation - HopCount (int) for Packet, Allow Looping (boolean) for controlled flooding
- packet.h : Header file for packet class
- router.h : Header file for router class
- driver.cpp : The file in which network topology, packets are constructed and run for simulation.

Instructions to Run:

Setting up input file, under project directory make `input.in` according to `input-format.txt`

```
<number of routers>
[next 'num of routers' lines]
<addresses of router>
[end of router]
<number of edges>
[for each edge Ri - Rj]
<address of router-i>
<address of router-j>
[end edges]
<number of Packets>
[for each packet pi]
<payload>
<destination router address>
[end packets]
```



In project directory, on terminal run:

```
g++ settings.h packet.h router.h
g++ driver.cpp -o driver

//Show output on terminal
./driver

//Redirect output to a file output.out
./driver > output.out
```

Default Configuration:

- **Default mode is controlled flooding.**
- **For uncontrolled flooding, with looping. In settings.h, change line 7 to, and recompile**

```
        static const bool _allowLooping = true;
```

- **Default hop count value is 4. To change, goto line 5 in settings.h and change. Recompile for changes to reflect.**

```
        static const int _defaultHopCount = <new_Value>;
```

Notes:

For an example goto `example` folder. Consists of an image of network topology, corresponding input file made according to `input-format.txt` and output files for controlled and uncontrolled flooding.

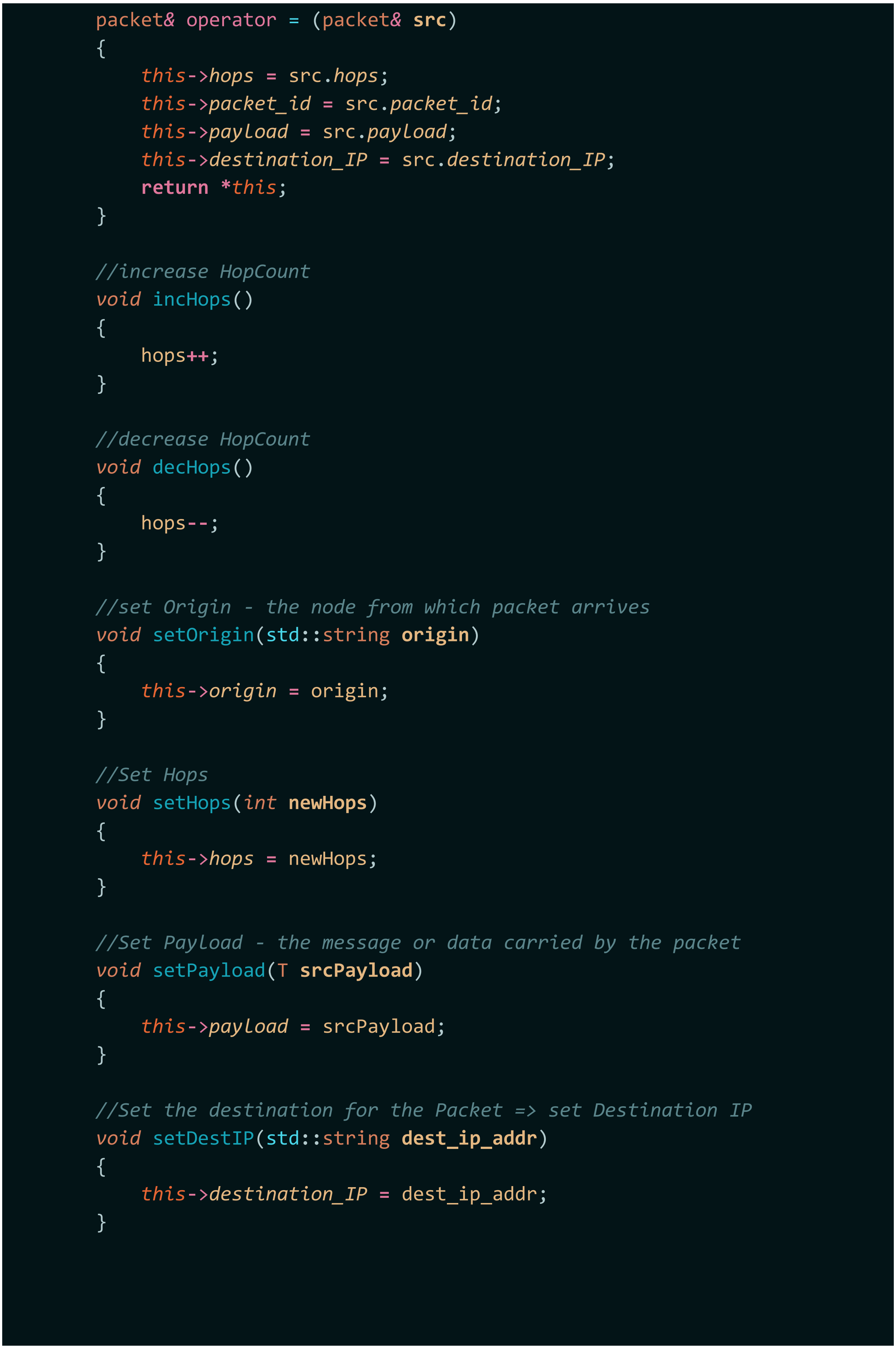
Code Files:

settings.h

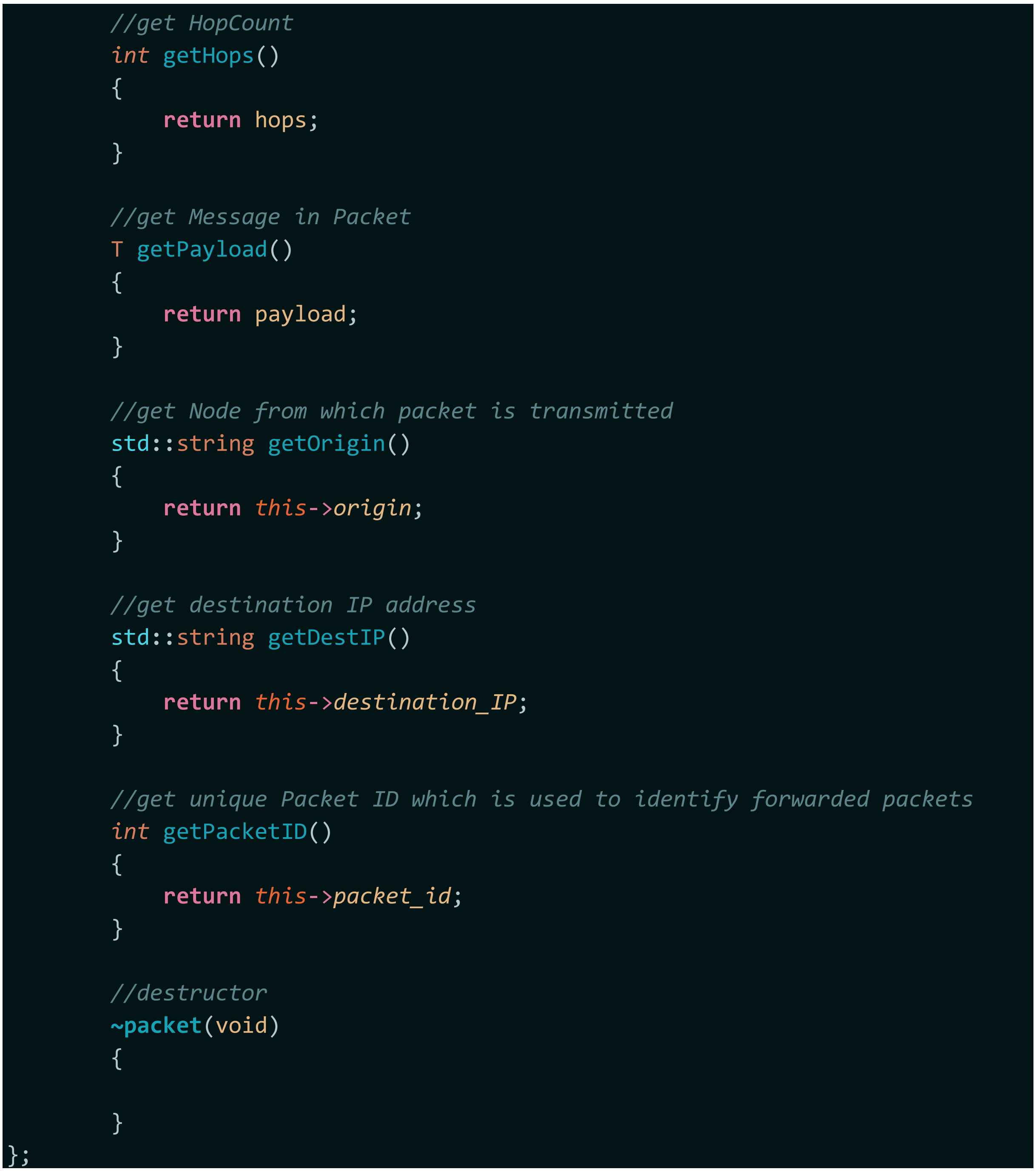
```
//creating a singleton class to determine default configuration for
//simulation.
class Settings{
    public:
        //default hopCount values for packet Object
        static const int _defaultHopCount = 4;
        //boolean to allow looping or not in flooding algorithm
        static const bool _allowLooping = false;
};
```

packet.h

```
#include "settings.h"
#include <string>
int packetIDseq = 0;

//class defining a packet
template <class T>
class packet{
    private:
        int packet_id; //packet_id : unique ID used to identify packets on a
network
        int hops; //hops: hop Counter for the packets. takes the default value
as per the value in the Settings singleton
        T payload; //payload: message in the packet
        Settings settings;
        std::string origin; //origin: stores the IP Address of router or host
from packet originates
        std::string destination_IP; //destination_IP: stores the IP address of
the dest. router for the packet.

    public:
        //contructors
        packet(void)
        {
            this->hops = settings._defaultHopCount;
            packet_id = packetIDseq++;
        }
        packet(T &payLoad, std::string dest_ip_addr)
        {
            this->hops = settings._defaultHopCount;
            packet_id = packetIDseq++;
            payload = payLoad;
            destination_IP = dest_ip_addr;
        }
```



```
    packet& operator = (packet& src)
    {
        this->hops = src.hops;
        this->packet_id = src.packet_id;
        this->payload = src.payload;
        this->destination_IP = src.destination_IP;
        return *this;
    }

    //increase HopCount
    void incHops()
    {
        hops++;
    }

    //decrease HopCount
    void decHops()
    {
        hops--;
    }

    //set Origin - the node from which packet arrives
    void setOrigin(std::string origin)
    {
        this->origin = origin;
    }

    //Set Hops
    void setHops(int newHops)
    {
        this->hops = newHops;
    }

    //Set Payload - the message or data carried by the packet
    void setPayload(T srcPayload)
    {
        this->payload = srcPayload;
    }

    //Set the destination for the Packet => set Destination IP
    void setDestIP(std::string dest_ip_addr)
    {
        this->destination_IP = dest_ip_addr;
    }
```

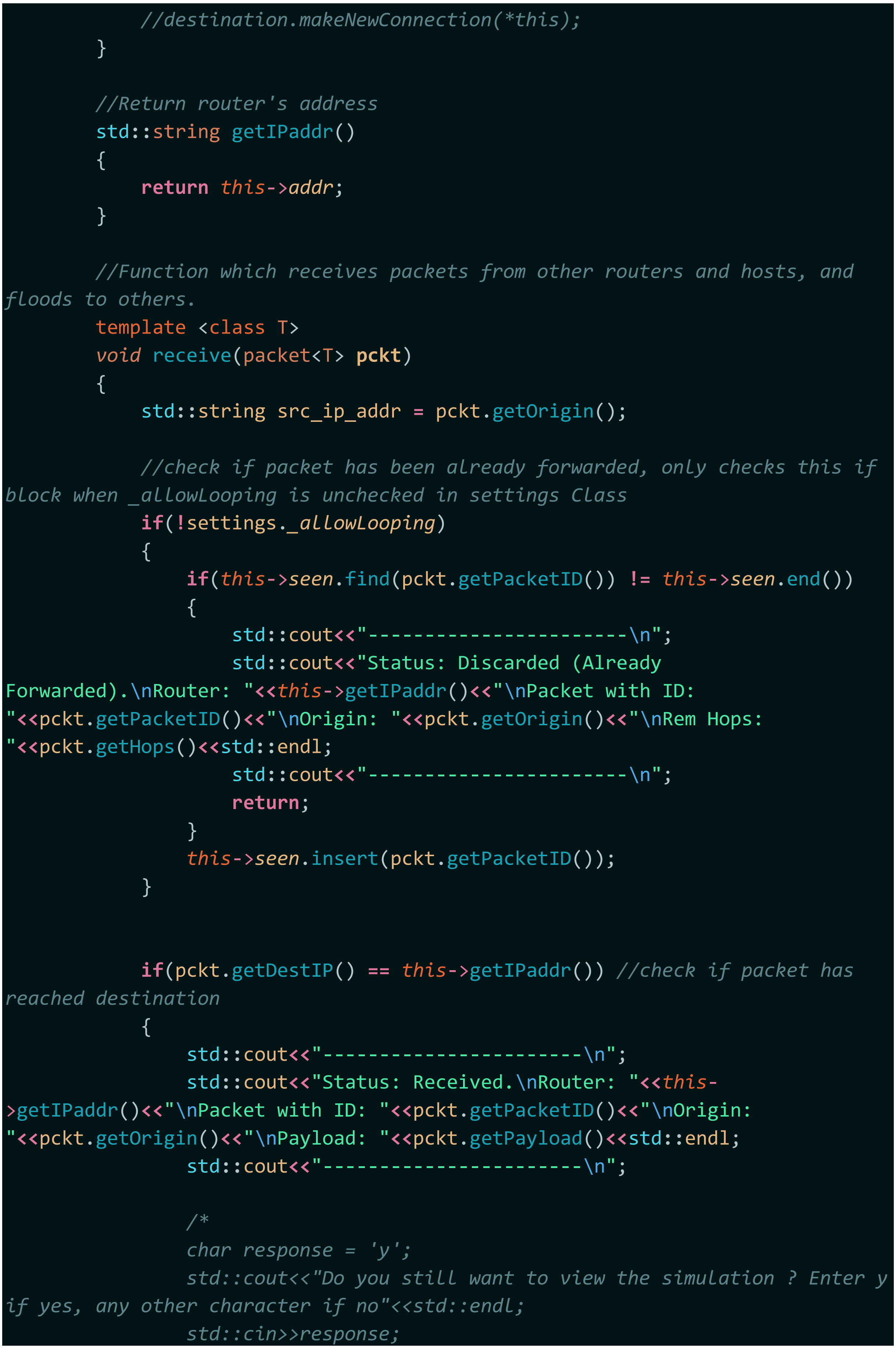


```
        //get HopCount
        int getHops()
        {
            return hops;
        }

        //get Message in Packet
        T getPayload()
        {
            return payload;
        }

        //get Node from which packet is transmitted
        std::string getOrigin()
        {
            return this->origin;
        }

        //get destination IP address
        std::string getDestIP()
        {
            return this->destination_IP;
        }

        //get unique Packet ID which is used to identify forwarded packets
        int getPacketID()
        {
            return this->packet_id;
        }

        //destructor
        ~packet(void)
        {

        }
};
```

P.T.O

router.h:

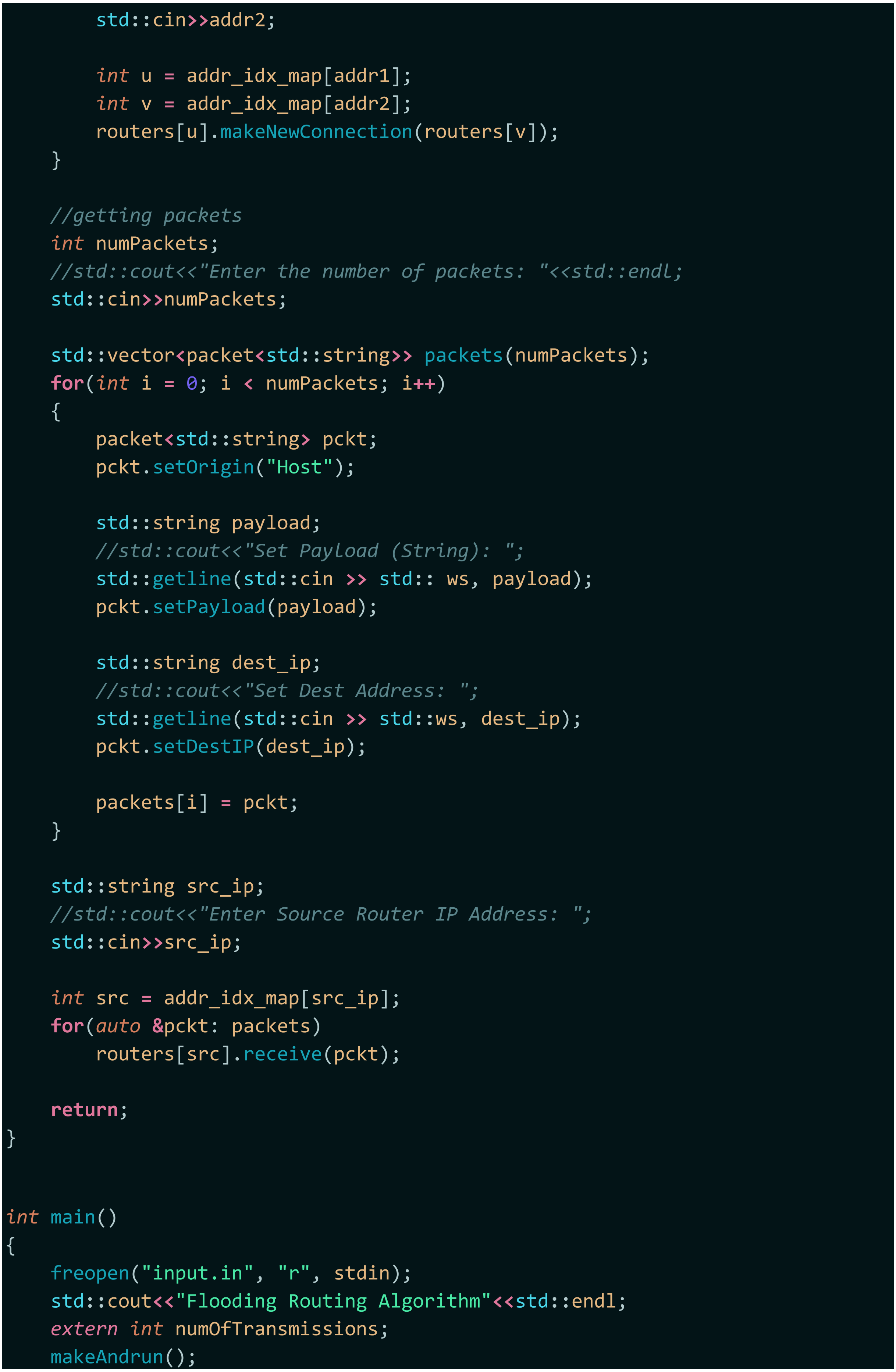
```
#include<stdlib.h>
#include<string>
#include<iostream>
#include<list>
#include<unordered_set>
#include "packet.h"

int numOfTransmissions = 0; //keeps track of total number of packets flooded
(includes the original packet)

//class defining router => the nodes in our topology
class router{

    private:
        std::string addr; //addr: stores addr by which router is identified
        std::unordered_set<router*> adjRouters; //set of routers adjacent to
this router
        std::unordered_set<int> seen; //set of seen packet_ids by the router,
used for identifying previously forwarded packets
        static Settings settings; //static object of the singleton Settings
class for access

    public:
        //Constructors and Copy Assignment
        router(void)
        {

        }
        router(std::string router_addr)
        {
            this->addr = router_addr;
        }
        router& operator = (router& src){
            this->addr = src.addr;
            this->adjRouters = std::unordered_set< router*
>(src.adjRouters.begin(), src.adjRouters.end());
            return *this;
        }

        //Function to put destination in this router's adjRouters and vice
versa
        void makeNewConnection(router &destination)
        {
            this->adjRouters.insert(&destination);
            destination.adjRouters.insert(this);
            //std::cout<<"Added "<<destination.getIPaddr()<<" to "<<this-
>getIPaddr()<<"'s adj List"<<std::endl;
```



```
            //destination.makeNewConnection(*this);
        }

        //Return router's address
        std::string getIPaddr()
        {
            return this->addr;
        }

        //Function which receives packets from other routers and hosts, and
floods to others.
        template <class T>
        void receive(packet<T> pckt)
        {
            std::string src_ip_addr = pckt.getOrigin();

            //check if packet has been already forwarded, only checks this if
block when _allowLooping is unchecked in settings Class
            if(!settings._allowLooping)
            {
                if(this->seen.find(pckt.getPacketID()) != this->seen.end())
                {
                    std::cout<<"-----------------------\n";
                    std::cout<<"Status: Discarded (Already
Forwarded).\nRouter: "<<this->getIPaddr()<<"\nPacket with ID:
"<<pckt.getPacketID()<<"\nOrigin: "<<pckt.getOrigin()<<"\nRem Hops:
"<<pckt.getHops()<<std::endl;
                    std::cout<<"-----------------------\n";
                    return;
                }
                this->seen.insert(pckt.getPacketID());
            }


            if(pckt.getDestIP() == this->getIPaddr()) //check if packet has
reached destination
            {
                std::cout<<"-----------------------\n";
                std::cout<<"Status: Received.\nRouter: "<<this-
>getIPaddr()<<"\nPacket with ID: "<<pckt.getPacketID()<<"\nOrigin:
"<<pckt.getOrigin()<<"\nPayload: "<<pckt.getPayload()<<std::endl;
                std::cout<<"-----------------------\n";

                /*
                char response = 'y';
                std::cout<<"Do you still want to view the simulation ? Enter y
if yes, any other character if no"<<std::endl;
                std::cin>>response;
                if(response != 'y')
                    exit(EXIT_SUCCESS);

                */
                std::cout<<"-----------------------"<<std::endl;
                std::cout<<"Resuming Simulation....\n"<<std::endl;
            }
            else if(pckt.getHops() == 0) //check if packet has zero hopCount
            {
                std::cout<<"-----------------------\n";
                std::cout<<"Status: Discarded (Hop Count = 0).\nRouter: "<<this->getIPaddr()<<"\nPacket with ID: "<<pckt.getPacketID()<<"\nOrigin: "<<pckt.getOrigin()<<std::endl;
                std::cout<<"-----------------------\n";
            }
            else //flooding to all adjacent routers except the origin node for this packet
            {
                std::cout<<"-----------------------\n";
                std::cout<<"Status: In Transit.\nRouter: "<<this->getIPaddr()<<"\nPacket with ID: "<<pckt.getPacketID()<<"\nOrigin: "<<pckt.getOrigin()<<"\nRem Hops: "<<pckt.getHops()<<std::endl;
                std::cout<<"-----------------------\n";

                //sending clone packets to all adjacent Routers except the packet's origin node
                for(auto adjRouter: this->adjRouters)
                    if(adjRouter->getIPaddr() != src_ip_addr)
                        this->send(pckt, adjRouter); //sends packet to adjRouter
            }

            return;
        }

        template <class T>
        void send(packet<T> pckt, router *destination) //function to send packets
        {
            numOfTransmissions++;
            pckt.decHops(); //decreasing hops for the cloned packet to be sent
            pckt.setOrigin(this->getIPaddr()); //rechanging origin for newly cloned packet to current router
            //std::cout<<"The origin for this cloned packet is "<<pckt.getOrigin()<<std::endl;
            destination->receive(pckt); //calling receive for destination
router, for it to receive and flood to its neighbours.
        }

        ~router()
        {

        }
};
```

driver.cpp

```
#include<bits/stdc++.h>
#include "router.h"

//function to define router topology, make packets, and simulate
void makeAndrun()
{
    std::vector<router> routers;
    int numRouters;

    //std::cout<<"Enter the number of routers: "<<std::endl;
    std::cin>>numRouters;
    std::unordered_map<std::string, int> addr_idx_map;

    //getting IP addresses of all routers and pushing them onto a vector of
routers
    for(int i = 0; i < numRouters; i++)
    {
        std::string ip_addr;
        //std::cout<<"Enter IP Address for Router-"<<i<<": ";
        std::cin>>ip_addr;
        addr_idx_map[ip_addr] = i;
        routers.push_back(router(ip_addr));
    }

    //getting all edges and setting up connections
    int numEdges;
    //std::cout<<"Enter the number of edges/connections in the topology:
"<<std::endl;
    std::cin>>numEdges;

    while(numEdges--)
    {
        std::string addr1, addr2;
        //std::cout<<"To make an edge, enter the IP addresses of the 2 routers
on 2 separate lines: "<<std::endl;
        std::cin>>addr1;
```



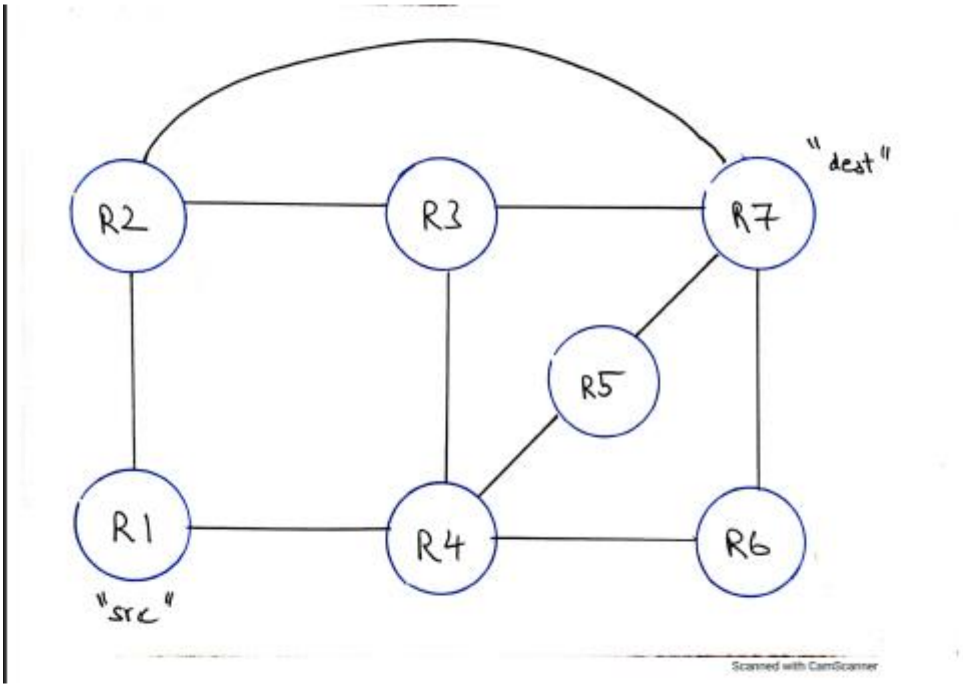
```
        std::cin>>addr2;

        int u = addr_idx_map[addr1];
        int v = addr_idx_map[addr2];
        routers[u].makeNewConnection(routers[v]);
    }

    //getting packets
    int numPackets;
    //std::cout<<"Enter the number of packets: "<<std::endl;
    std::cin>>numPackets;

    std::vector<packet<std::string>> packets(numPackets);
    for(int i = 0; i < numPackets; i++)
    {
        packet<std::string> pckt;
        pckt.setOrigin("Host");

        std::string payload;
        //std::cout<<"Set Payload (String): ";
        std::getline(std::cin >> std:: ws, payload);
        pckt.setPayload(payload);

        std::string dest_ip;
        //std::cout<<"Set Dest Address: ";
        std::getline(std::cin >> std::ws, dest_ip);
        pckt.setDestIP(dest_ip);

        packets[i] = pckt;
    }

    std::string src_ip;
    //std::cout<<"Enter Source Router IP Address: ";
    std::cin>>src_ip;

    int src = addr_idx_map[src_ip];
    for(auto &pckt: packets)
        routers[src].receive(pckt);

    return;
}


int main()
{
    freopen("input.in", "r", stdin);
    std::cout<<"Flooding Routing Algorithm"<<std::endl;
    extern int numOfTransmissions;
    makeAndrun();
    std::cout<<std::endl;
    std::cout<<"Simulation is Over!! "<<std::endl;
    std::cout<<"The number of transmissions is
"<<numOfTransmissions<<std::endl;

    return 1;
}
```

Running the Program:

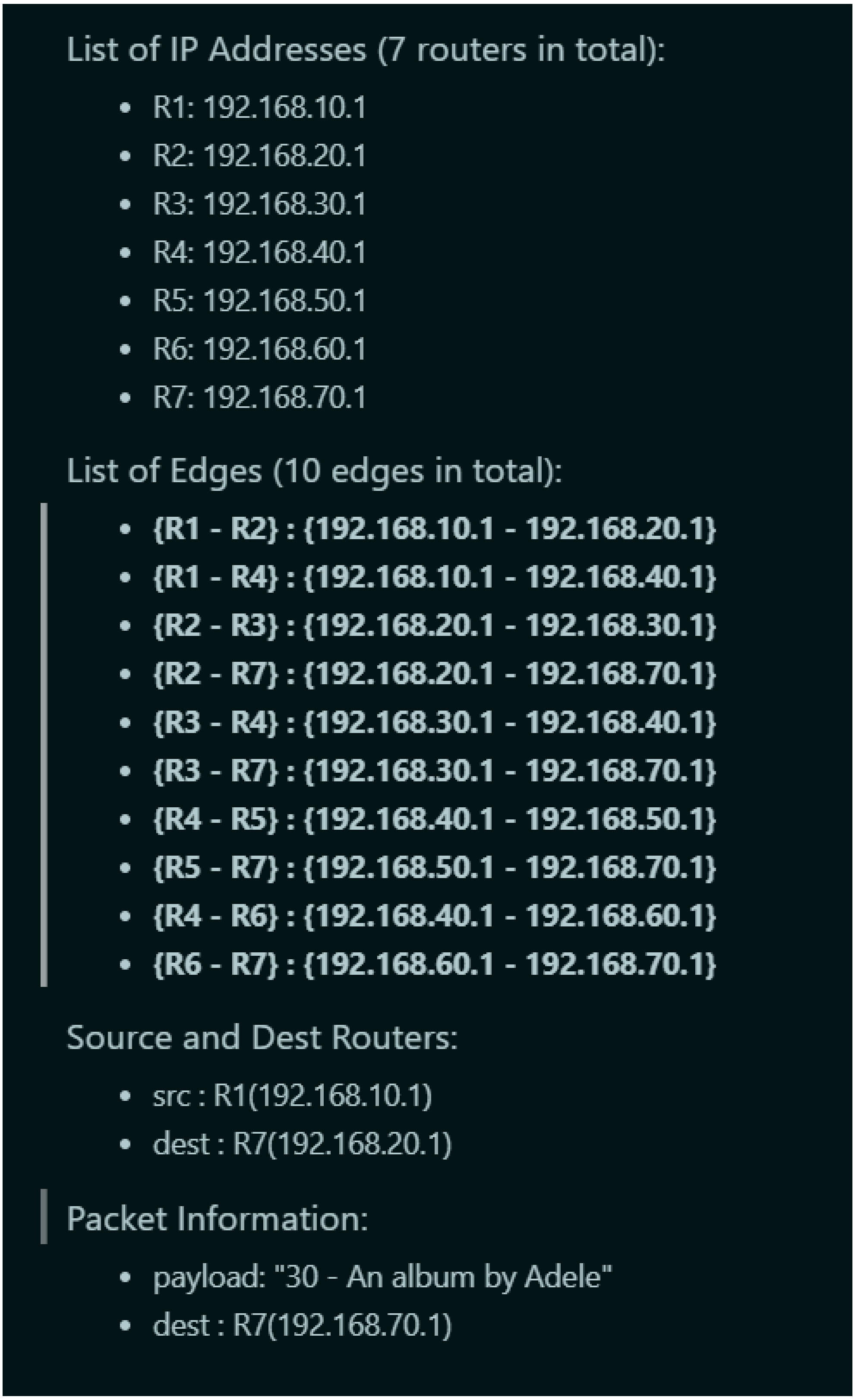
Follow the instructions given in the README.md

Output:

The network that is going to be simulated can be listed as follows:



List of Input Parameters Used:



List of IP Addresses (7 routers in total):

- R1: 192.168.10.1
- R2: 192.168.20.1
- R3: 192.168.30.1
- R4: 192.168.40.1
- R5: 192.168.50.1
- R6: 192.168.60.1
- R7: 192.168.70.1

List of Edges (10 edges in total):

> - **{R1 - R2} : {192.168.10.1 - 192.168.20.1}**
> - **{R1 - R4} : {192.168.10.1 - 192.168.40.1}**
> - **{R2 - R3} : {192.168.20.1 - 192.168.30.1}**
> - **{R2 - R7} : {192.168.20.1 - 192.168.70.1}**
> - **{R3 - R4} : {192.168.30.1 - 192.168.40.1}**
> - **{R3 - R7} : {192.168.30.1 - 192.168.70.1}**
> - **{R4 - R5} : {192.168.40.1 - 192.168.50.1}**
> - **{R5 - R7} : {192.168.50.1 - 192.168.70.1}**
> - **{R4 - R6} : {192.168.40.1 - 192.168.60.1}**
> - **{R6 - R7} : {192.168.60.1 - 192.168.70.1}**

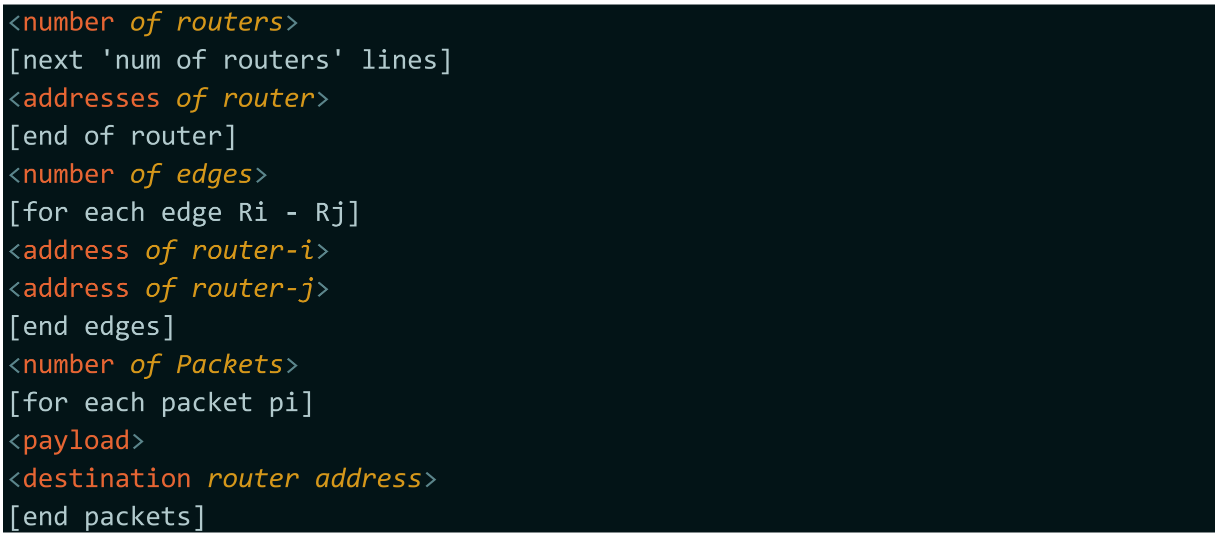
Source and Dest Routers:

- src : R1(192.168.10.1)
- dest : R7(192.168.20.1)

> Packet Information:

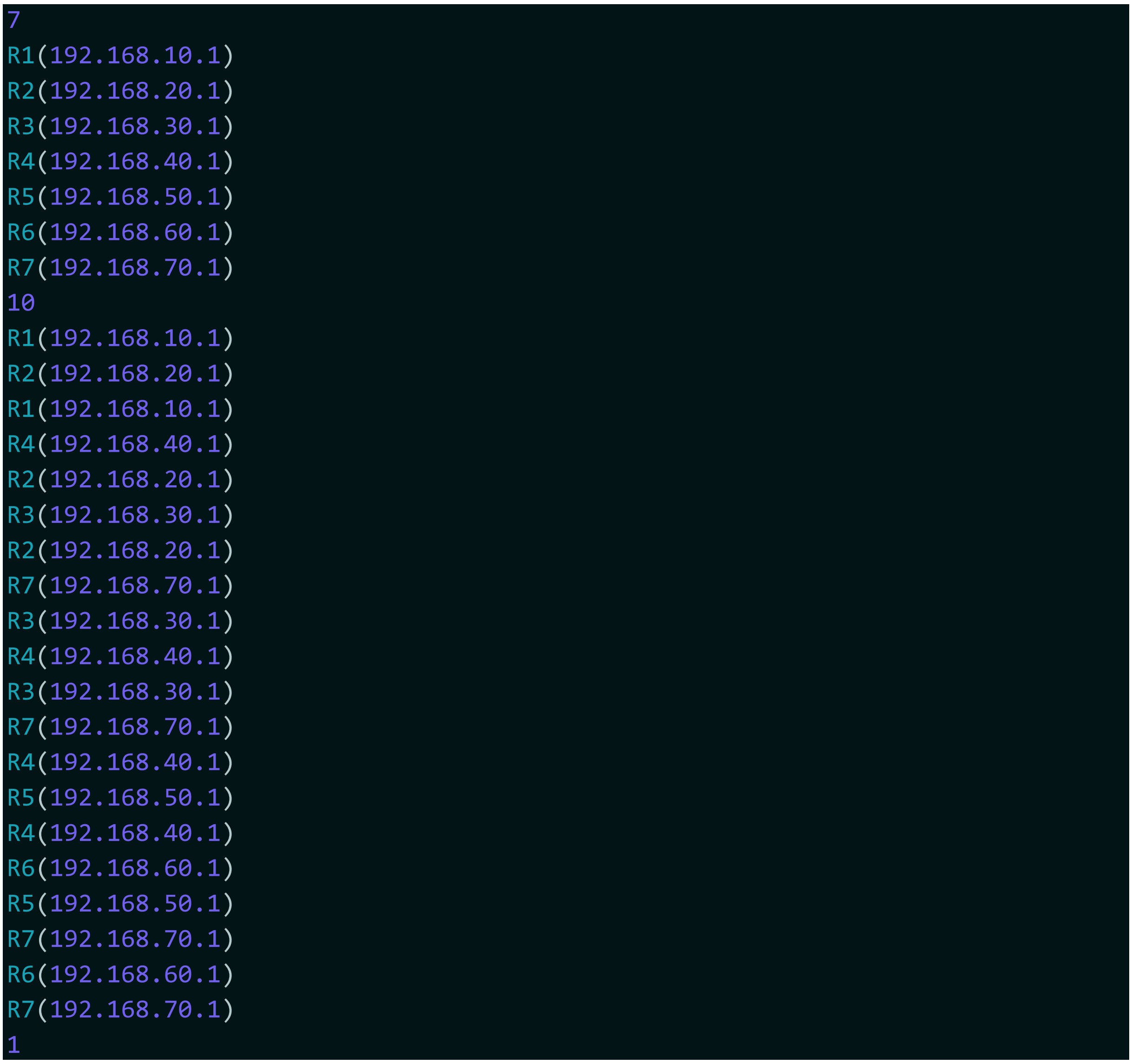
- payload: "30 - An album by Adele"
- dest : R7(192.168.70.1)

Input File Format:



```
<number of routers>
[next 'num of routers' lines]
<addresses of router>
[end of router]
<number of edges>
[for each edge Ri - Rj]
<address of router-i>
<address of router-j>
[end edges]
<number of Packets>
[for each packet pi]
<payload>
<destination router address>
[end packets]
```

Input File for above Network Topology: (File name: input.in)



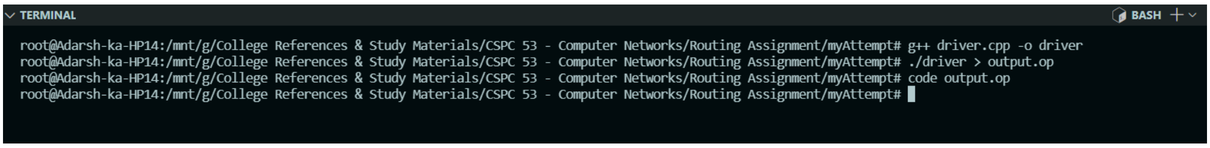
```
7
R1(192.168.10.1)
R2(192.168.20.1)
R3(192.168.30.1)
R4(192.168.40.1)
R5(192.168.50.1)
R6(192.168.60.1)
R7(192.168.70.1)
10
R1(192.168.10.1)
R2(192.168.20.1)
R1(192.168.10.1)
R4(192.168.40.1)
R2(192.168.20.1)
R3(192.168.30.1)
R2(192.168.20.1)
R7(192.168.70.1)
R3(192.168.30.1)
R4(192.168.40.1)
R3(192.168.30.1)
R7(192.168.70.1)
R4(192.168.40.1)
R5(192.168.50.1)
R4(192.168.40.1)
R6(192.168.60.1)
R5(192.168.50.1)
R7(192.168.70.1)
R6(192.168.60.1)
R7(192.168.70.1)
1
```

```
30 - An Album by Adele
R7(192.168.70.1)
R1(192.168.10.1)
```

Settings: Default Hop Count is 4.

Case – 1: When there is no controlled flooding, i.e, when looping is allowed.

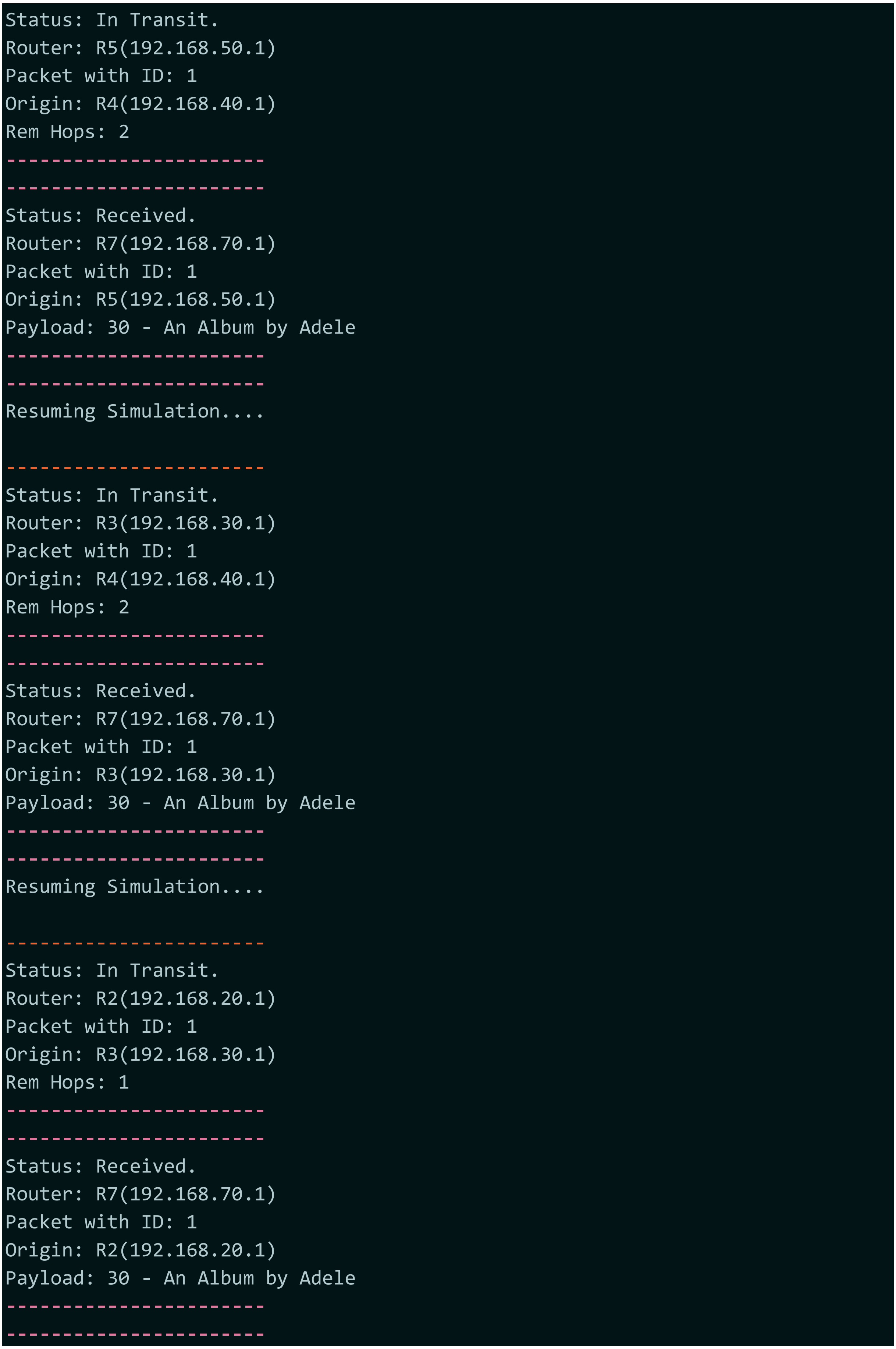
On terminal:



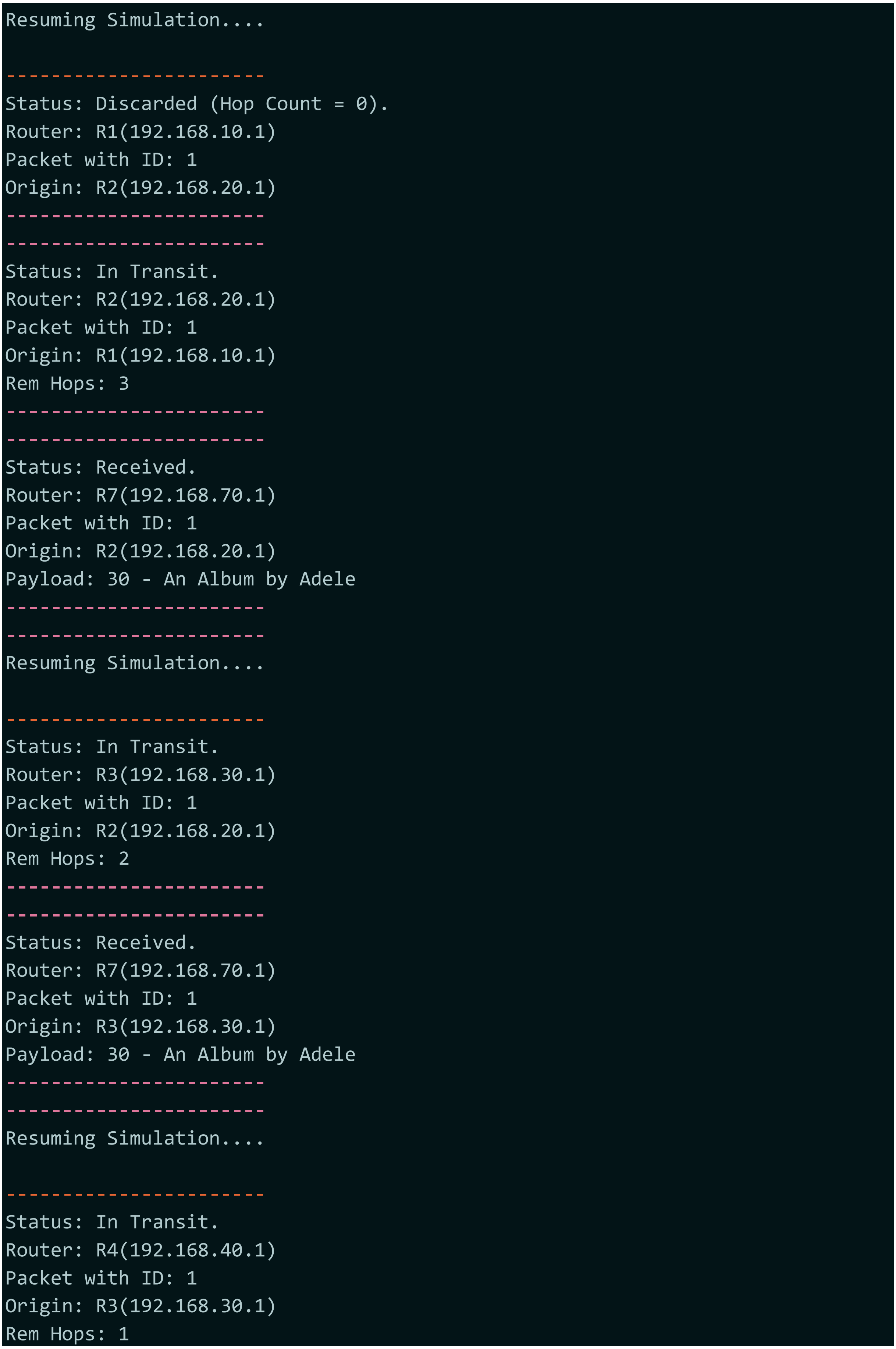
```
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt# g++ driver.cpp -o driver
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt# ./driver > output.op
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt# code output.op
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt#
```

output.op

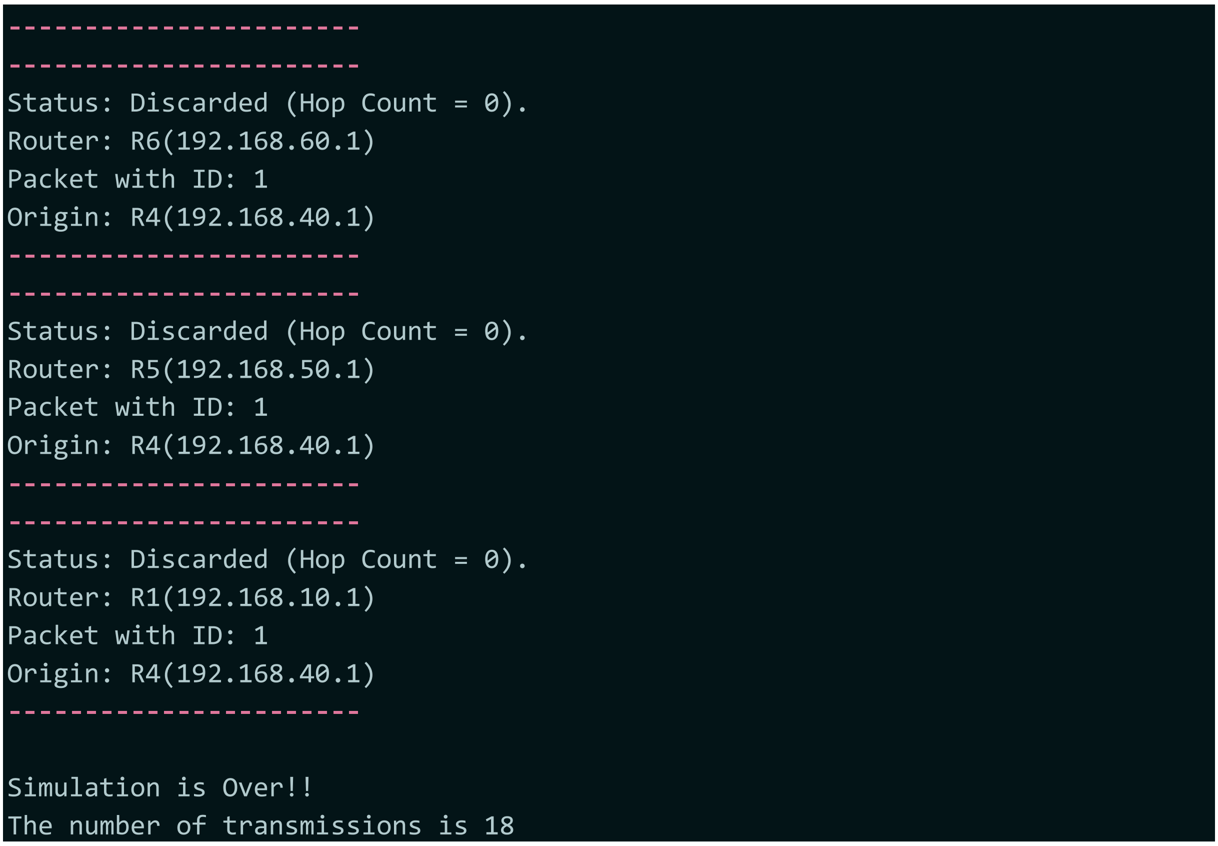
```
Flooding Routing Algorithm
-----------------------
Status: In Transit.
Router: R1(192.168.10.1)
Packet with ID: 1
Origin: 
Rem Hops: 4
-----------------------
-----------------------
Status: In Transit.
Router: R4(192.168.40.1)
Packet with ID: 1
Origin: R1(192.168.10.1)
Rem Hops: 3
-----------------------
-----------------------
Status: In Transit.
Router: R6(192.168.60.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
Rem Hops: 2
-----------------------
-----------------------
Status: Received.
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R6(192.168.60.1)
Payload: 30 - An Album by Adele
-----------------------
-----------------------
Resuming Simulation....

-----------------------
```



```
Status: In Transit.
Router: R5(192.168.50.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
Rem Hops: 2
-----------------------
-----------------------
Status: Received.
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R5(192.168.50.1)
Payload: 30 - An Album by Adele
-----------------------
-----------------------
Resuming Simulation....

-----------------------
Status: In Transit.
Router: R3(192.168.30.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
Rem Hops: 2
-----------------------
-----------------------
Status: Received.
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R3(192.168.30.1)
Payload: 30 - An Album by Adele
-----------------------
-----------------------
Resuming Simulation....

-----------------------
Status: In Transit.
Router: R2(192.168.20.1)
Packet with ID: 1
Origin: R3(192.168.30.1)
Rem Hops: 1
-----------------------
-----------------------
Status: Received.
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R2(192.168.20.1)
Payload: 30 - An Album by Adele
-----------------------
-----------------------
```



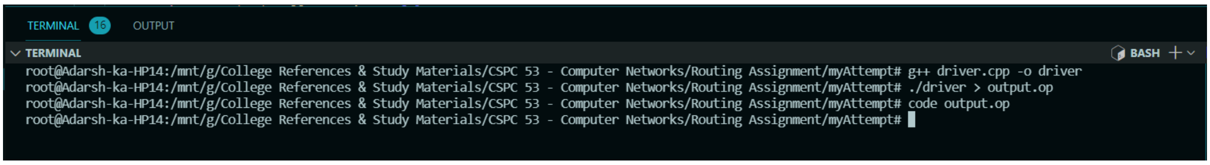
```
Resuming Simulation....

-----------------------
Status: Discarded (Hop Count = 0).
Router: R1(192.168.10.1)
Packet with ID: 1
Origin: R2(192.168.20.1)
-----------------------
-----------------------
Status: In Transit.
Router: R2(192.168.20.1)
Packet with ID: 1
Origin: R1(192.168.10.1)
Rem Hops: 3
-----------------------
-----------------------
Status: Received.
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R2(192.168.20.1)
Payload: 30 - An Album by Adele
-----------------------
-----------------------
Resuming Simulation....

-----------------------
Status: In Transit.
Router: R3(192.168.30.1)
Packet with ID: 1
Origin: R2(192.168.20.1)
Rem Hops: 2
-----------------------
-----------------------
Status: Received.
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R3(192.168.30.1)
Payload: 30 - An Album by Adele
-----------------------
-----------------------
Resuming Simulation....

-----------------------
Status: In Transit.
Router: R4(192.168.40.1)
Packet with ID: 1
Origin: R3(192.168.30.1)
Rem Hops: 1
```



```
-----------------------
-----------------------
Status: Discarded (Hop Count = 0).
Router: R6(192.168.60.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
-----------------------
-----------------------
Status: Discarded (Hop Count = 0).
Router: R5(192.168.50.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
-----------------------
-----------------------
Status: Discarded (Hop Count = 0).
Router: R1(192.168.10.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
-----------------------

Simulation is Over!!
The number of transmissions is 18
```

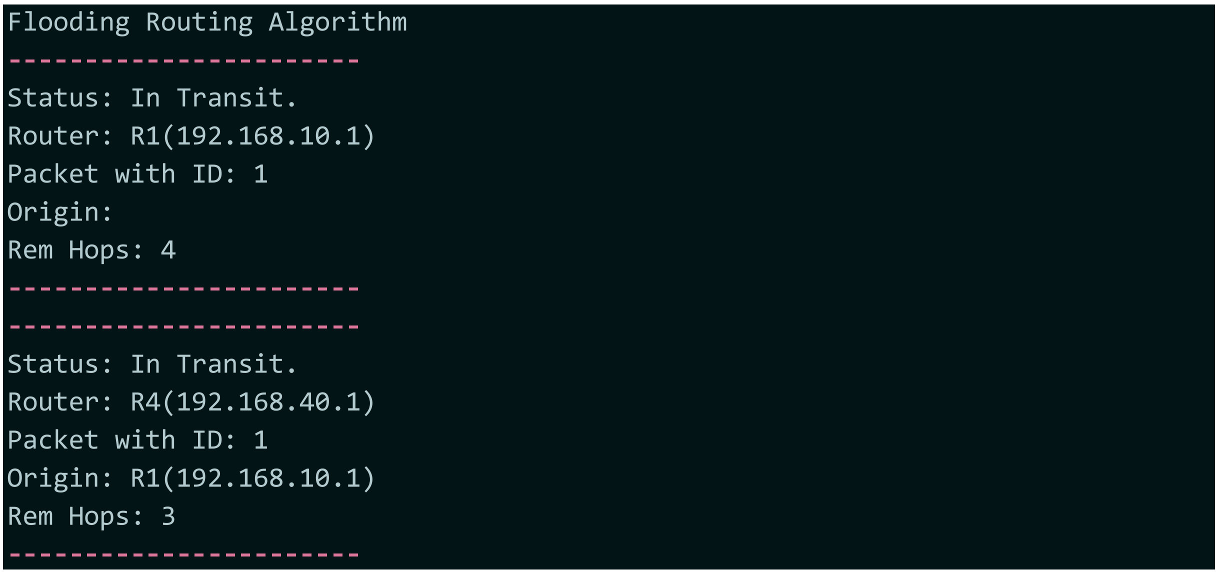
Case – 2: When there is controlled flooding, every packet is forwarded only once.

On terminal:

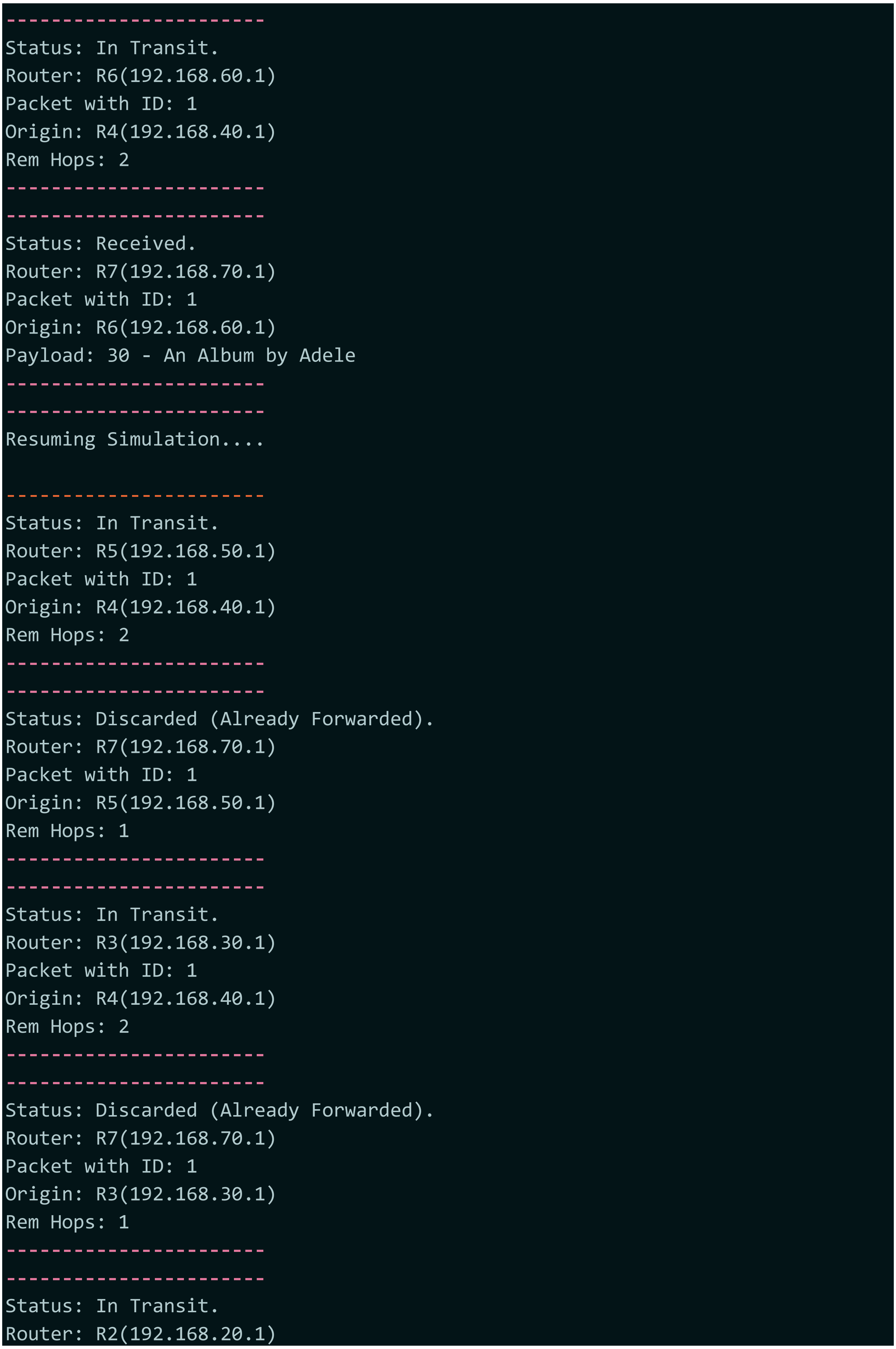


```
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt# g++ driver.cpp -o driver
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt# ./driver > output.op
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt# code output.op
root@Adarsh-ka-HP14:/mnt/g/College References & Study Materials/CSPC 53 - Computer Networks/Routing Assignment/myAttempt# 
```

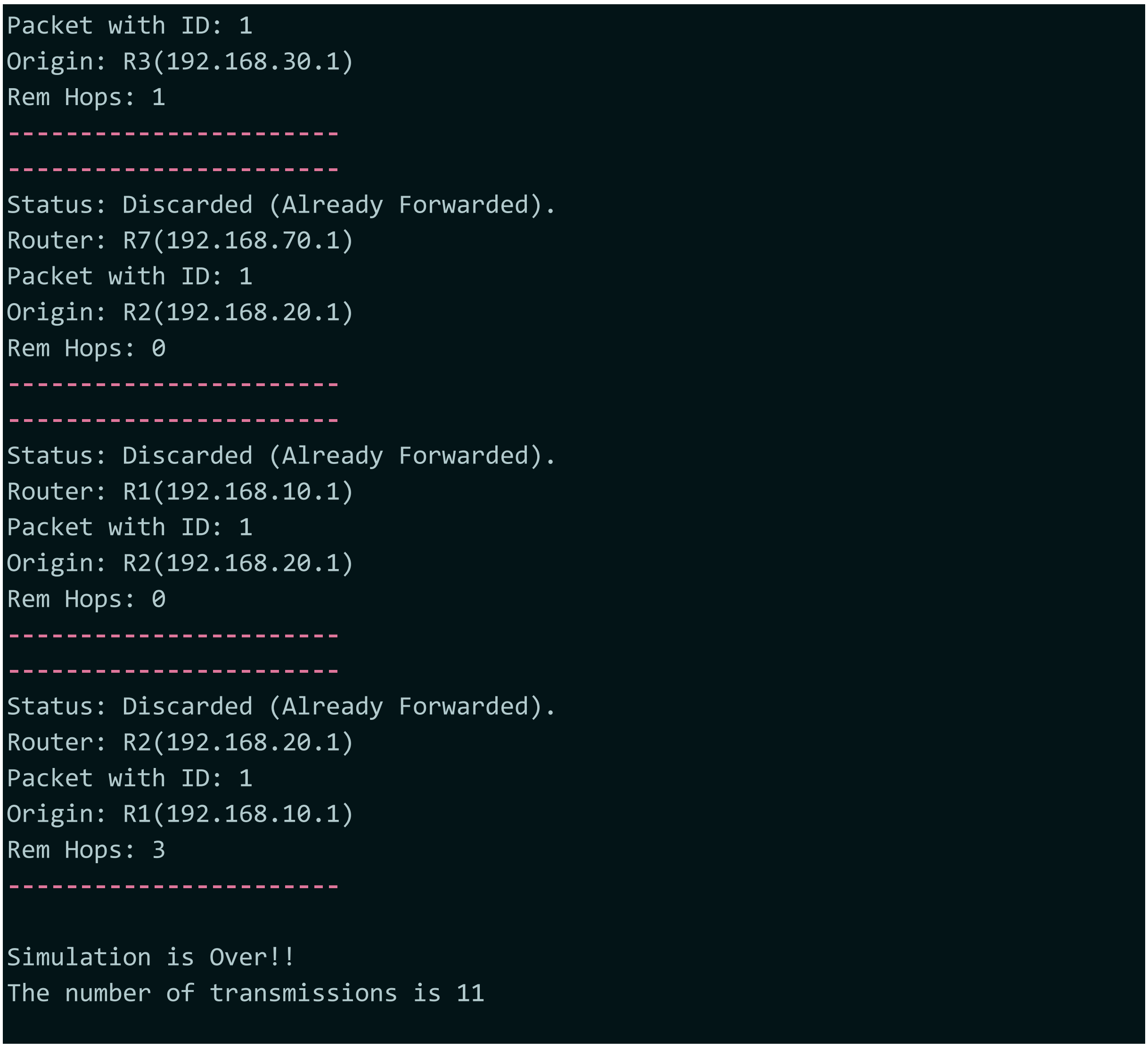
output.op:



```
Flooding Routing Algorithm
-----------------------
Status: In Transit.
Router: R1(192.168.10.1)
Packet with ID: 1
Origin:
Rem Hops: 4
-----------------------
-----------------------
Status: In Transit.
Router: R4(192.168.40.1)
Packet with ID: 1
Origin: R1(192.168.10.1)
Rem Hops: 3
-----------------------
```



```
-----------------------
Status: In Transit.
Router: R6(192.168.60.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
Rem Hops: 2
-----------------------
-----------------------
Status: Received.
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R6(192.168.60.1)
Payload: 30 - An Album by Adele
-----------------------
-----------------------
Resuming Simulation....

-----------------------
Status: In Transit.
Router: R5(192.168.50.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
Rem Hops: 2
-----------------------
-----------------------
Status: Discarded (Already Forwarded).
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R5(192.168.50.1)
Rem Hops: 1
-----------------------
-----------------------
Status: In Transit.
Router: R3(192.168.30.1)
Packet with ID: 1
Origin: R4(192.168.40.1)
Rem Hops: 2
-----------------------
-----------------------
Status: Discarded (Already Forwarded).
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R3(192.168.30.1)
Rem Hops: 1
-----------------------
-----------------------
Status: In Transit.
Router: R2(192.168.20.1)
```



```
Packet with ID: 1
Origin: R3(192.168.30.1)
Rem Hops: 1
-----------------------
-----------------------
Status: Discarded (Already Forwarded).
Router: R7(192.168.70.1)
Packet with ID: 1
Origin: R2(192.168.20.1)
Rem Hops: 0
-----------------------
-----------------------
Status: Discarded (Already Forwarded).
Router: R1(192.168.10.1)
Packet with ID: 1
Origin: R2(192.168.20.1)
Rem Hops: 0
-----------------------
-----------------------
Status: Discarded (Already Forwarded).
Router: R2(192.168.20.1)
Packet with ID: 1
Origin: R1(192.168.10.1)
Rem Hops: 3
-----------------------

Simulation is Over!!
The number of transmissions is 11
```

We can see that the number of cloned packets with controlled flooding has reduced from the initial case.