Automatic Repeat Request

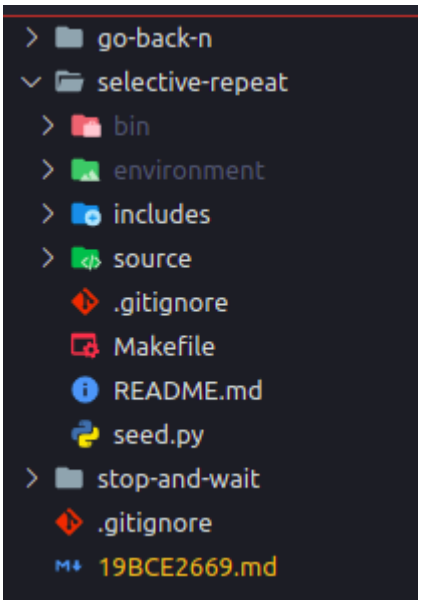
Submitted by:

- Yash Kumar Verma
- 19BCE2669

For this assignment, I have written a full scale network simulator from scratch in C++.

Directory Structure

The zip file shared has three folders, named `go-back-n`, `selective-repeat` and `stop-and-wait` for the three types of AQR algorithms.



Each folder has makefile attached, so that the project can be compiled using a single command. The project logic is written in C++, and a seeding script to generate dummy packets for demonstration is written in python.

How to run

To run the project, enter any one of the directory and run the following command:

```
make simulate
cd environment
python3 seed.py
```

```
→  selective-repeat git:(master) ✗ make simulate && cd environment && python3 seed.py && cd ..
#
#
# removing old builds
#
#
rm -rf bin/*
rm -rf environment
#
#
# building directory tree
#
#
mkdir environment
mkdir environment/sender
mkdir environment/receiver
mkdir environment/channel
#
#
# building packages
#
#
make all
make[1]: Entering directory '/home/yash/Desktop/files/works/projects/college-netcom/arq-simulators/selective-repea
t'
g++ --std=c++17 -c source/receiver.cpp -o bin/receiver.o
g++ --std=c++17 -c includes/rainbow/rainbow.cpp -o bin/rainbow.o
g++ --std=c++17 -c includes/simulator/simulator.cpp -o bin/simulator.o
g++ --std=c++17 -c includes/log/log.cpp -o bin/log.o
```

Go-Back-N

Sender Code

```cpp
/**
 *@author: YashKumarVerma
 *@desc: main entry point of application
 *@time: Tuesday 30-March-2021 01:18:27
 **/

#include<iostream>
#include<vector>

#include<ctime>
#include<cstdio>

#include"./../includes/rainbow/rainbow.h"
#include"./../includes/simulator/simulator.h"
#include"./../includes/log/log.h"

using namespace std;

/** to facilitate creating problems in transmission based on frequency **/
bool problemCreator(float errorRate){
    int number = rand()%(11);
    return number < errorRate;
}

/** main function **/
int main(){
    srand(time(0));
```

```
    /** set the fraction of error rate. This is applied on all operations
**/
    int error_rate = 3;
    int total_packets = 10;
    float simulator_frequency = 1.0;

    /** set window size **/
    int n = 4;

    /** set max timeout limit **/
    int max_timeout_range = 10;
    int timeout = 0;

    /** all configurations end here, now application codebase **/
    Simulator simulator(total_packets, "sender");
    simulator.setClockFrequency(simulator_frequency);
    simulator.setErrorRate(error_rate);

    // anchor contains how many items have been successfully transmitted
    int anchor = 1;

    // lower and upper to keep track of sliding window
    int lower = 1;
    int upper = lower + n;


    // till we get callback for each packet
    while(simulator.senderTransmissionNotComplete()){

        while(lower <= upper && lower <= total_packets){

            /** this simulates packet loss **/
            if(problemCreator(simulator.errorRate)){
                log::sender_error("Error sending packet " +
to_string(lower));
                lower++;
            }
            simulator.sendDataPacket(lower++);
        }

        //    the first element of sliding window is verified
        while(simulator.acceptAcknowledgementIfExist(anchor)){
            log::sender_info("Received acknowledgement for #" +
to_string(anchor));
            upper++;
            anchor++;
            timeout = 0;
        }

        if(anchor == total_packets){
            break;
        }
```

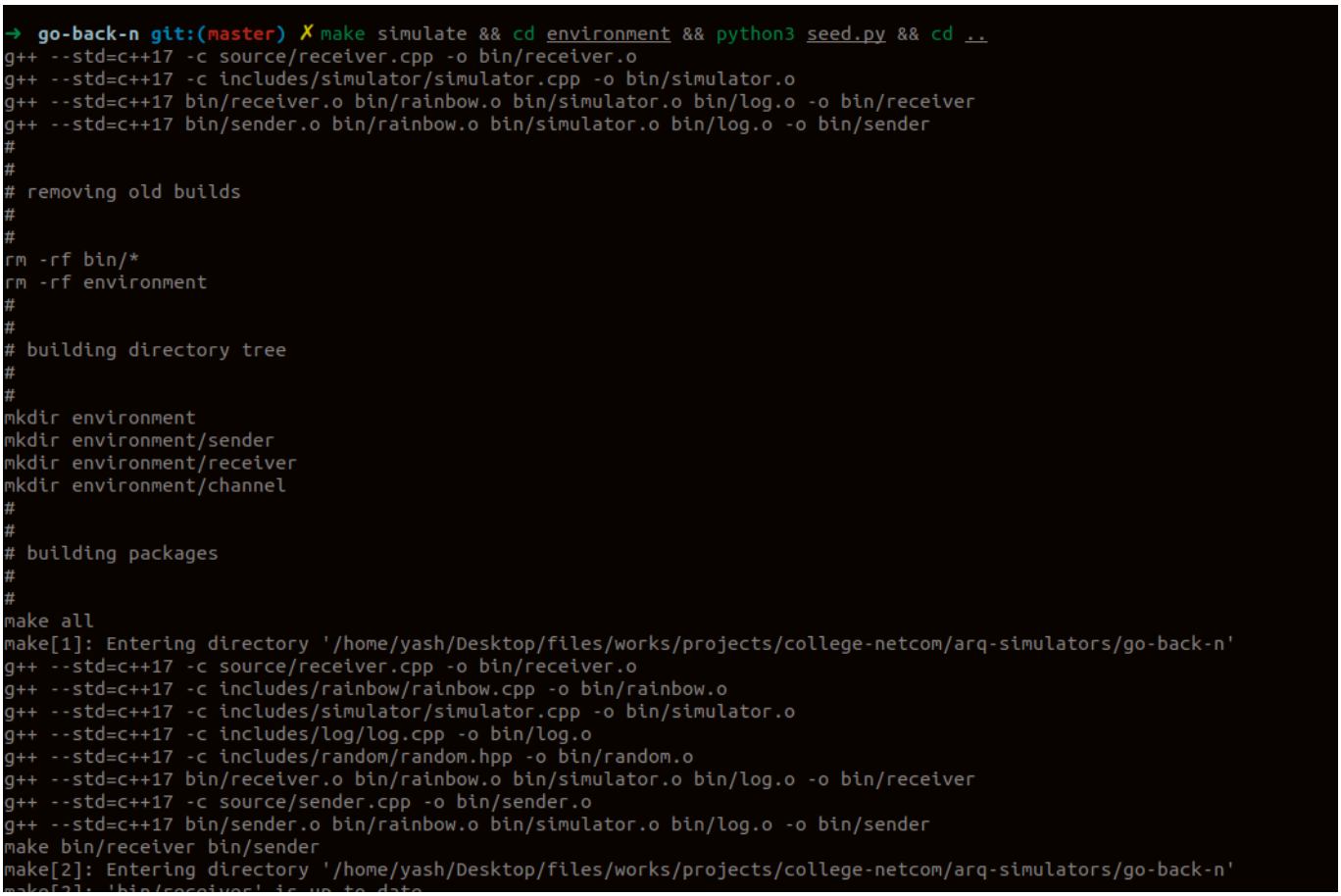
```
        if(timeout < max_timeout_range){
            timeout++;
            log::sender_info("Waiting for acknowledgement for packet #" +
to_string(anchor) + ", timeout = " + to_string(timeout));
        }else{
            // trigger request burst again
            lower = anchor;
            upper = lower + n;
            if(upper > total_packets){
                upper = total_packets;
            }
            timeout = 0;

            for(int i=anchor; i<upper; i++){
                simulator.abandonPacket(i);
            }
            log::sender_error("Request timed out, sending packets from " +
to_string(lower) + " to " + to_string(upper));
        }

        // tick the process cycle
        simulator.tick();
    }
    return 0;
}
```

Receiver Code

```
/**
 *@author: YashKumarVerma
 *@desc: main entry point of application
 *@time: Tuesday 30-March-2021 01:18:27
 **/

#include<iostream>
#include<vector>
#include"./../includes/rainbow/rainbow.h"
#include"./../includes/simulator/simulator.h"
#include"./../includes/log/log.h"

using namespace std;


/** to facilitate creating problems in transmission based on frequency **/
bool problemCreator(float errorRate){
    int number = rand()%(11);
    return number < errorRate;
}


int main(){
    srand(time(0));
```

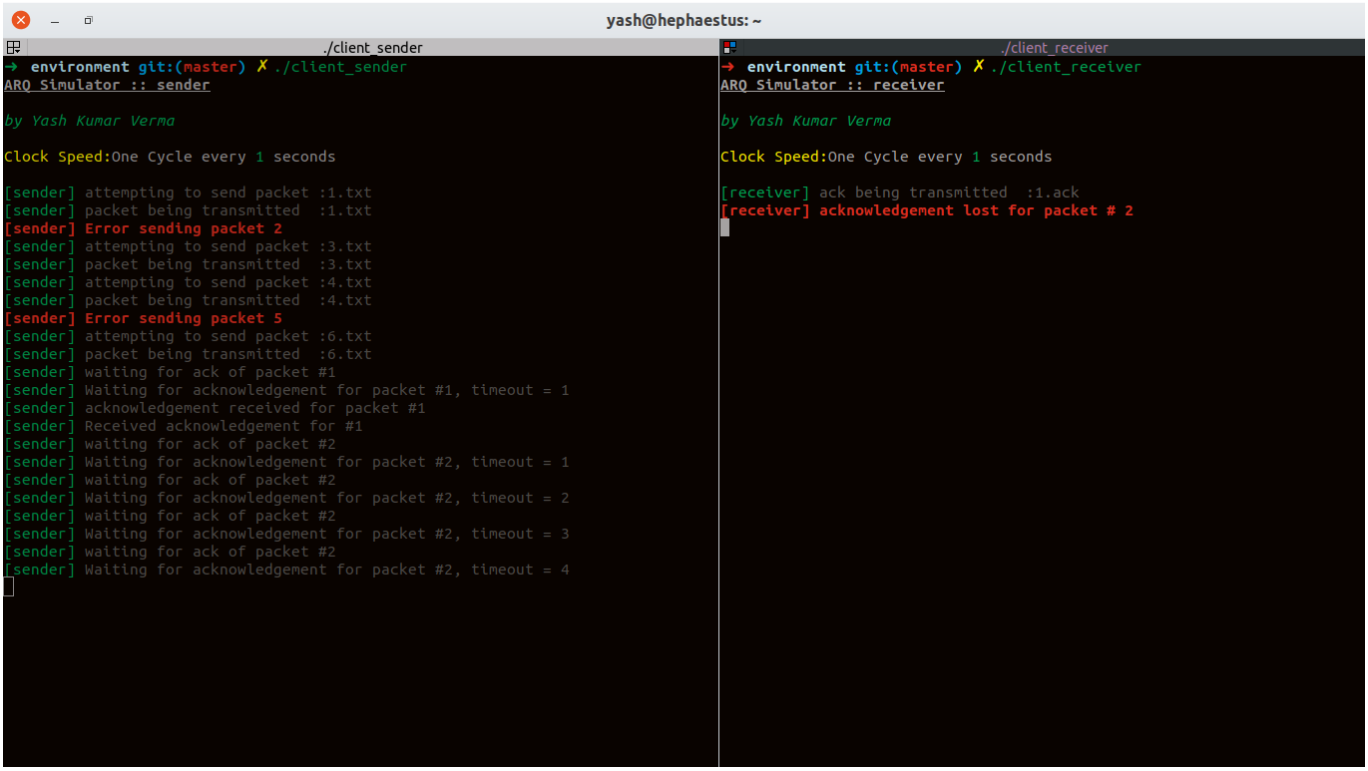
```
    /** simulator configurations **/
    int error_rate = 3;
    int total_packets = 10;
    float simulator_frequency = 1.0;


    Simulator simulator(total_packets, "receiver");
    simulator.setClockFrequency(simulator_frequency);
    simulator.setErrorRate(error_rate);

    int counter = 1;
    while(simulator.receiverTransmissionNotComplete() && counter !=
total_packets){

        /** simulate acknowledgement loss **/
        if(problemCreator(simulator.errorRate)){
            log::receiver_error("acknowledgement lost for packet # " +
to_string(counter));
        }

        // check if there is any packet received
        if(simulator.sendAcknowledgementForPacketIfExist(counter)){
            // expect next item now
            counter++;
        }

        simulator.tick();
    }
    return 0;
}
```

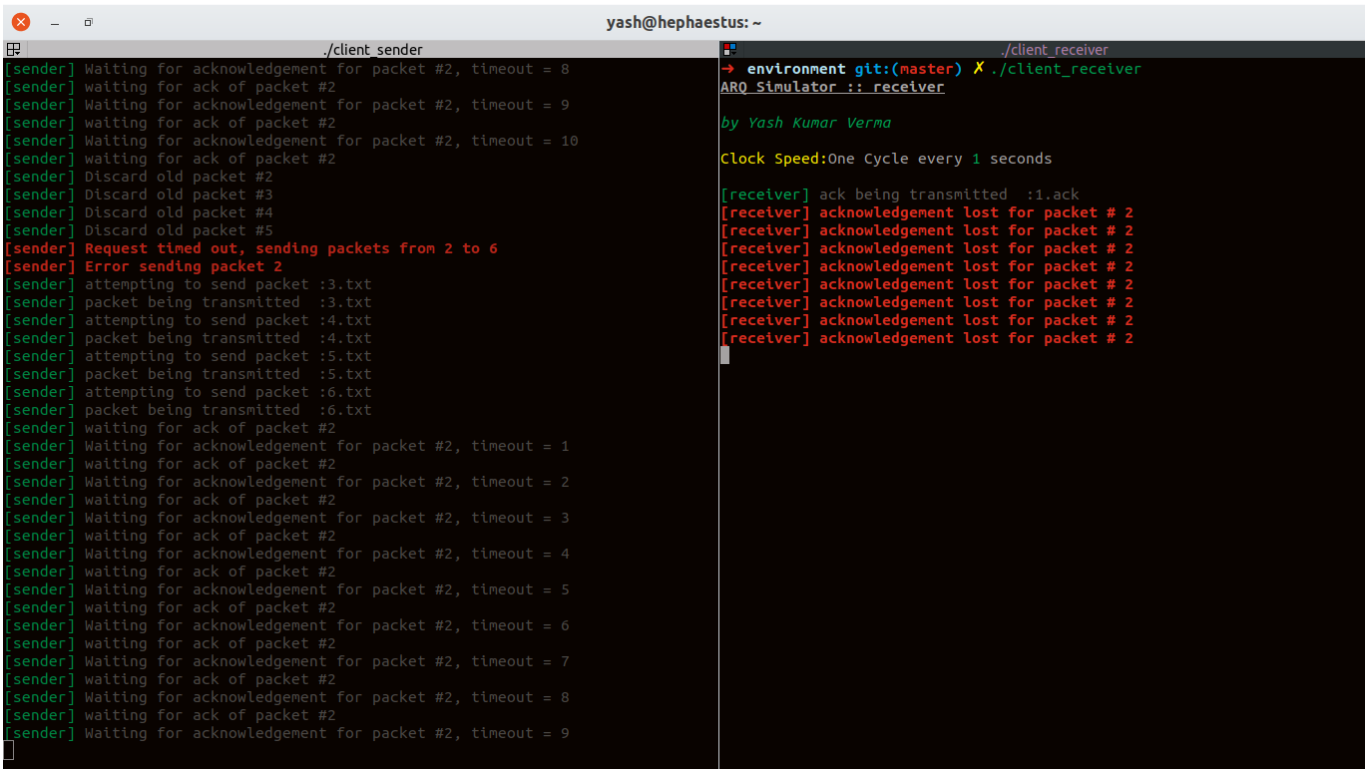
Installation

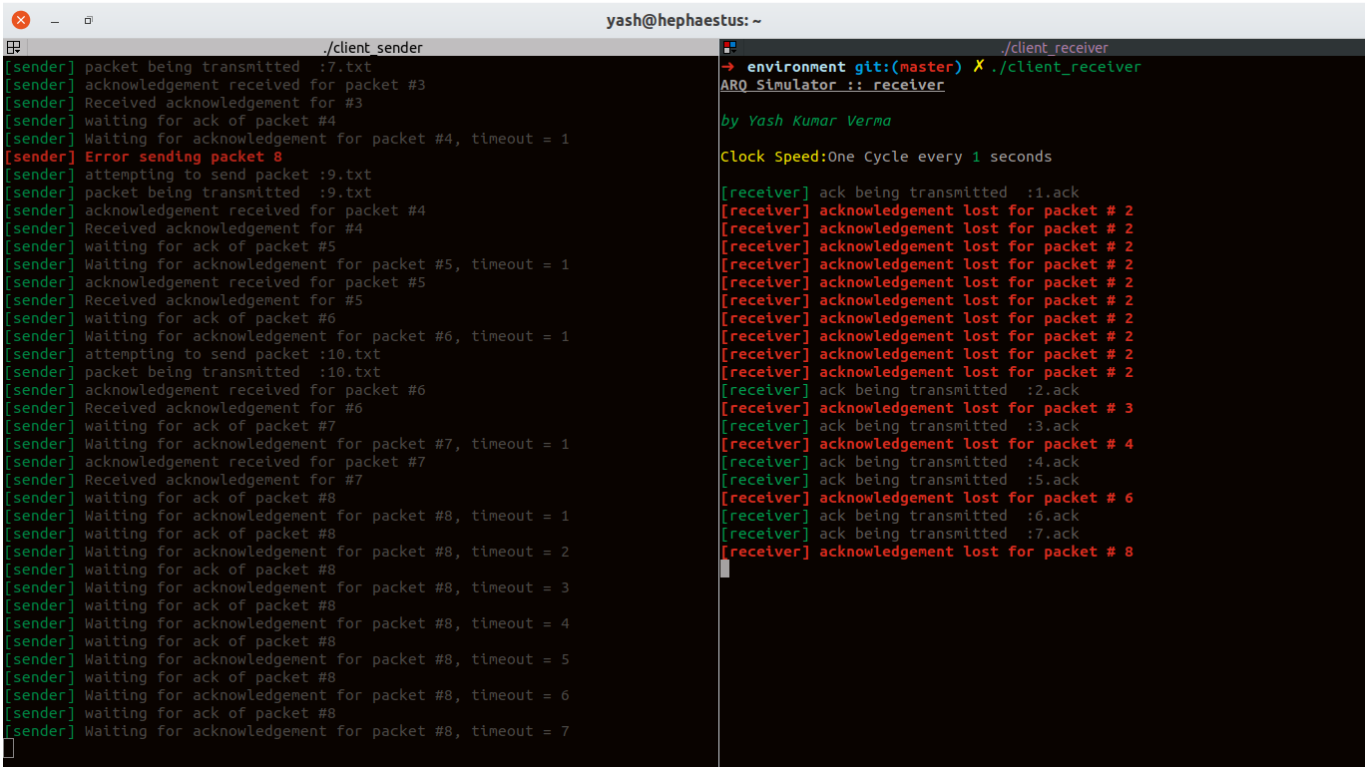


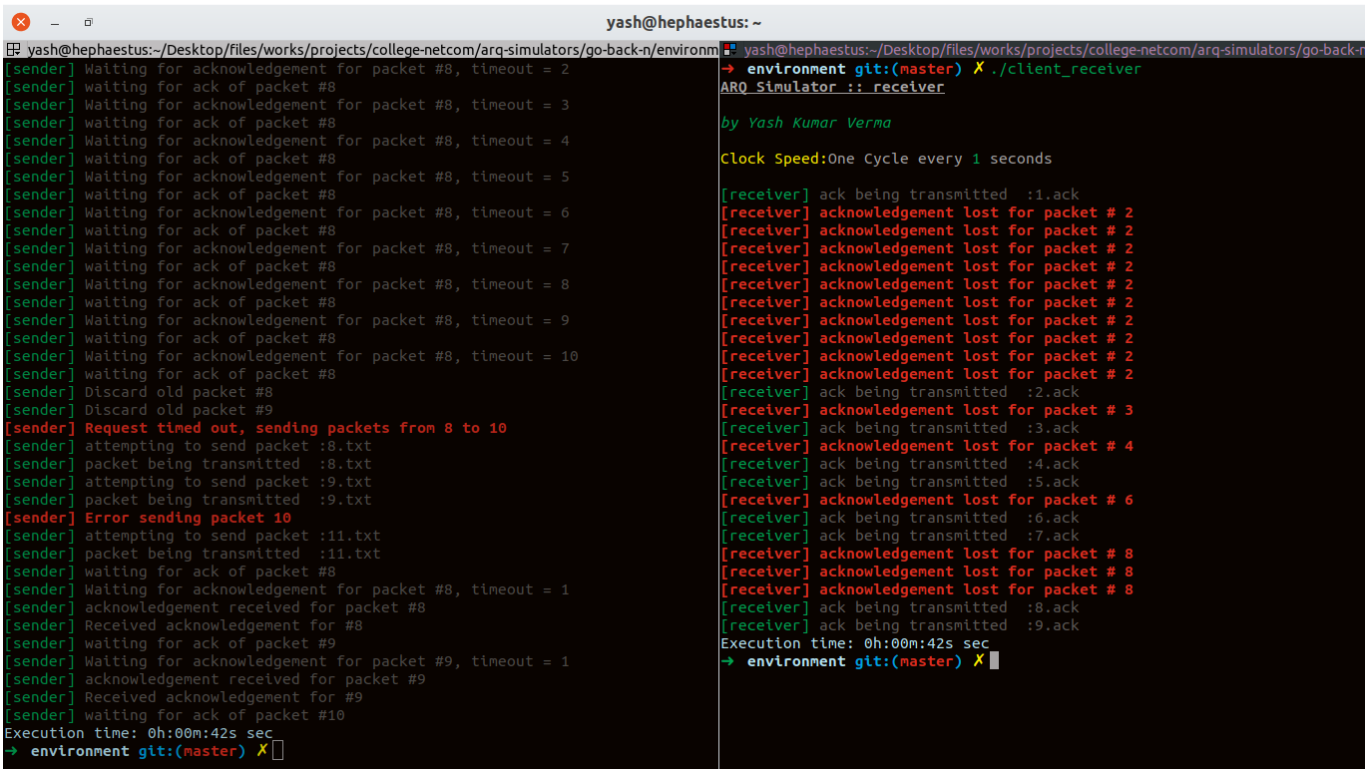
```
→ go-back-n git:(master) ✗ make simulate && cd environment && python3 seed.py && cd ..
g++ --std=c++17 -c source/receiver.cpp -o bin/receiver.o
g++ --std=c++17 -c includes/simulator/simulator.cpp -o bin/simulator.o
g++ --std=c++17 bin/receiver.o bin/rainbow.o bin/simulator.o bin/log.o -o bin/receiver
g++ --std=c++17 bin/sender.o bin/rainbow.o bin/simulator.o bin/log.o -o bin/sender
#
#
# removing old builds
#
#
rm -rf bin/*
rm -rf environment
#
#
# building directory tree
#
#
mkdir environment
mkdir environment/sender
mkdir environment/receiver
mkdir environment/channel
#
#
# building packages
#
#
make all
make[1]: Entering directory '/home/yash/Desktop/files/works/projects/college-netcom/arq-simulators/go-back-n'
g++ --std=c++17 -c source/receiver.cpp -o bin/receiver.o
g++ --std=c++17 -c includes/rainbow/rainbow.cpp -o bin/rainbow.o
g++ --std=c++17 -c includes/simulator/simulator.cpp -o bin/simulator.o
g++ --std=c++17 -c includes/log/log.cpp -o bin/log.o
g++ --std=c++17 -c includes/random/random.hpp -o bin/random.o
g++ --std=c++17 bin/receiver.o bin/rainbow.o bin/simulator.o bin/log.o -o bin/receiver
g++ --std=c++17 -c source/sender.cpp -o bin/sender.o
g++ --std=c++17 bin/sender.o bin/rainbow.o bin/simulator.o bin/log.o -o bin/sender
make bin/receiver bin/sender
make[2]: Entering directory '/home/yash/Desktop/files/works/projects/college-netcom/arq-simulators/go-back-n'
```

Output The error rate is defined by code, as a fraction.


yash@hephaestus: ~
./client_sender
→ environment git:(master) ✗ ./client_sender
ARQ Simulator :: sender
by Yash Kumar Verma
Clock Speed:One Cycle every 1 seconds
[sender] attempting to send packet :1.txt
[sender] packet being transmitted :1.txt
[sender] Error sending packet 2
[sender] attempting to send packet :3.txt
[sender] packet being transmitted :3.txt
[sender] attempting to send packet :4.txt
[sender] packet being transmitted :4.txt
[sender] Error sending packet 5
[sender] attempting to send packet :6.txt
[sender] packet being transmitted :6.txt
[sender] waiting for ack of packet #1
[sender] Waiting for acknowledgement for packet #1, timeout = 1
[sender] acknowledgement received for packet #1
[sender] Received acknowledgement for #1
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 1
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 2
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 3
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 4
./client_receiver
→ environment git:(master) ✗ ./client_receiver
ARQ Simulator :: receiver
by Yash Kumar Verma
Clock Speed:One Cycle every 1 seconds
[receiver] ack being transmitted :1.ack
[receiver] acknowledgement lost for packet # 2


yash@hephaestus: ~
./client_sender
[sender] Waiting for acknowledgement for packet #2, timeout = 8
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 9
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 10
[sender] waiting for ack of packet #2
[sender] Discard old packet #2
[sender] Discard old packet #3
[sender] Discard old packet #4
[sender] Discard old packet #5
[sender] Request timed out, sending packets from 2 to 6
[sender] Error sending packet 2
[sender] attempting to send packet :3.txt
[sender] packet being transmitted :3.txt
[sender] attempting to send packet :4.txt
[sender] packet being transmitted :4.txt
[sender] attempting to send packet :5.txt
[sender] packet being transmitted :5.txt
[sender] attempting to send packet :6.txt
[sender] packet being transmitted :6.txt
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 1
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 2
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 3
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 4
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 5
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 6
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 7
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 8
[sender] waiting for ack of packet #2
[sender] Waiting for acknowledgement for packet #2, timeout = 9
./client_receiver
→ environment git:(master) ✗ ./client_receiver
ARQ Simulator :: receiver
by Yash Kumar Verma
Clock Speed:One Cycle every 1 seconds
[receiver] ack being transmitted :1.ack
[receiver] acknowledgement lost for packet # 2
[receiver] acknowledgement lost for packet # 2
[receiver] acknowledgement lost for packet # 2
[receiver] acknowledgement lost for packet # 2
[receiver] acknowledgement lost for packet # 2
[receiver] acknowledgement lost for packet # 2
[receiver] acknowledgement lost for packet # 2
[receiver] acknowledgement lost for packet # 2





Selective Repeat

Sender Code

```
/**
 *@author: YashKumarVerma
 *@desc: main entry point of application
 *@time: Tuesday 30-March-2021 01:18:27
 **/

#include<iostream>
```

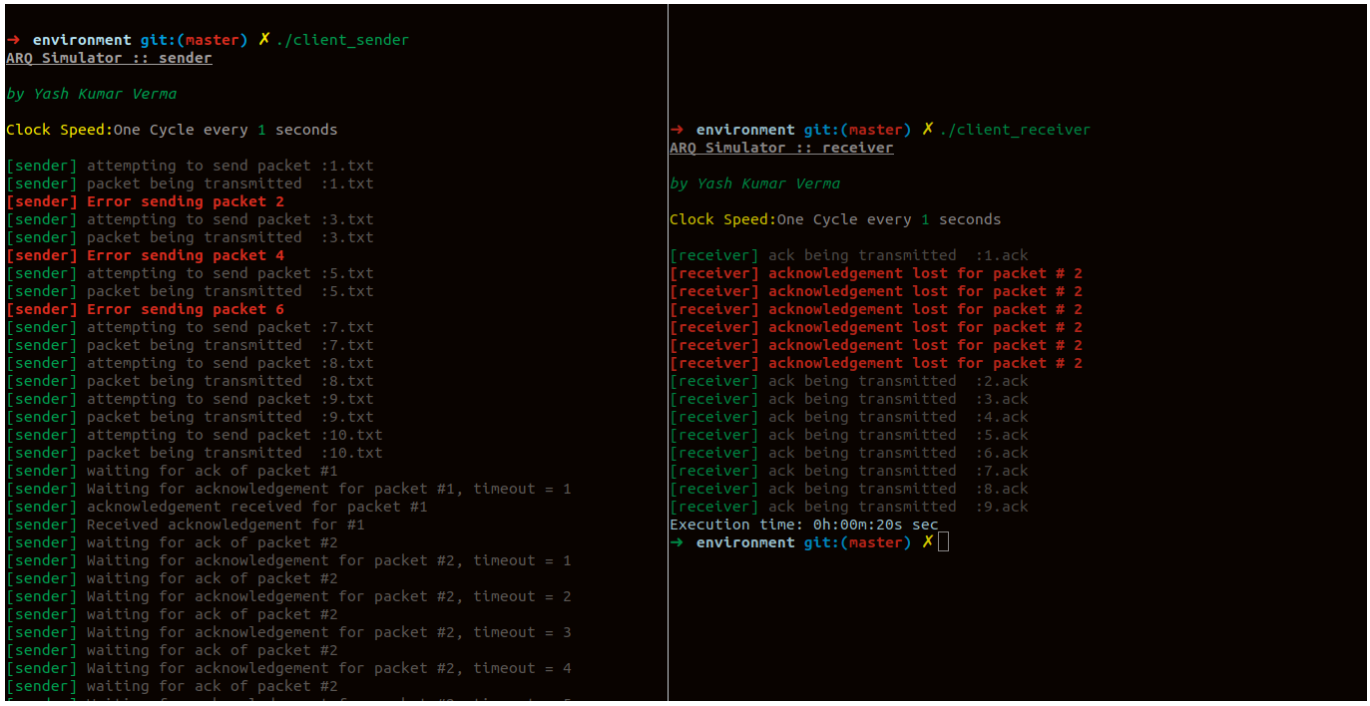
```
#include<vector>

#include<ctime>
#include<cstdio>

#include"./../includes/rainbow/rainbow.h"
#include"./../includes/simulator/simulator.h"
#include"./../includes/log/log.h"

using namespace std;

/** to facilitate creating problems in transmission based on frequency **/
bool problemCreator(float errorRate){
    int number = rand()%(11);
    return number < errorRate;
}

/** main function **/
int main(){
    srand(time(0));

    /** set the fraction of error rate. This is applied on all operations
**/
    int error_rate = 3;
    int total_packets = 10;
    float simulator_frequency = 1.0;

    /** set window size **/
    int n = 4;

    /** set max timeout limit **/
    int max_timeout_range = 10;
    int timeout = 0;

    /** all configurations end here, now application codebase **/
    Simulator simulator(total_packets, "sender");
    simulator.setClockFrequency(simulator_frequency);
    simulator.setErrorRate(error_rate);

    // anchor contains how many items have been successfully transmitted
    int anchor = 1;

    // lower and upper to keep track of sliding window
    int lower = 1;
    int upper = lower + n;

    /** initialize total packets**/
    vector<int>verifiedDeliveryTable(total_packets+1, 0);


    // till we get callback for each packet
    while(simulator.senderTransmissionNotComplete()){

        while(lower <= total_packets){
```

```
            /** this simulates packet loss **/
            if(problemCreator(simulator.errorRate)){
                log::sender_error("Error sending packet " +
to_string(lower));
                lower++;
            }
            simulator.sendDataPacket(lower++);
        }

        //    the first element of sliding window is verified
        while(simulator.acceptAcknowledgementIfExist(anchor)){
            log::sender_info("Received acknowledgement for #" +
to_string(anchor));
            verifiedDeliveryTable.at(anchor) = 1;
            anchor++;
            timeout = 0;
        }

        if(anchor == total_packets){
            break;
        }

        if(timeout < max_timeout_range){
            timeout++;
            log::sender_info("Waiting for acknowledgement for packet #" +
to_string(anchor) + ", timeout = " + to_string(timeout));
        }
        else{
            /** do nothing, just send the first missing acknowledgment
index **/
            for(int i=0; i<total_packets; i++){
                if(verifiedDeliveryTable.at(i) == 0){
                    simulator.sendDataPacket(i);
                    log::sender_error("Request timed out, resending
individual packet " + to_string(i));
                }
            }

            timeout = 0;
        }

        simulator.resendNegativeAcknowledgement();
        // tick the process cycle
        simulator.tick();
    }
    return 0;
}
```

Receiver Code

```
/**
 *@author: YashKumarVerma
 *@desc: main entry point of application
 *@time: Tuesday 30-March-2021 01:18:27
 **/

#include<iostream>
#include<vector>
#include"./../includes/rainbow/rainbow.h"
#include"./../includes/simulator/simulator.h"
#include"./../includes/log/log.h"

using namespace std;


/** to facilitate creating problems in transmission based on frequency **/
bool problemCreator(float errorRate){
    int number = rand()%(11);
    return number < errorRate;
}


int main(){
    srand(time(0));

    /** simulator configurations **/
    int error_rate = 3;
    int total_packets = 10;
    float simulator_frequency = 1.0;


    vector<int>verifiedDeliveryTable(total_packets+1, 0);


    Simulator simulator(total_packets, "receiver");
    simulator.setClockFrequency(simulator_frequency);
    simulator.setErrorRate(error_rate);

    int counter = 1;
    int timer = 0;
    while(simulator.receiverTransmissionNotComplete() && counter != 
total_packets){

        /** simulate acknowledgement loss **/
        if(problemCreator(simulator.errorRate)){
            log::receiver_error("acknowledgement lost for packet # " + 
to_string(counter));
        }

        // check if there is any packet received
        if(simulator.sendAcknowledgementForPacketIfExist(counter)){
            verifiedDeliveryTable.at(counter) = 1;
            // expect next item now
```

```
                counter++;
            }


            /**
             * send negative acknowledgement for all who were not
             * sent any response, either due to acknowledgemet loss
             * or timeout
             **/
            if(timer <= 5){
                for(int i=1; i < total_packets && i < counter-1; i++){
                    if(verifiedDeliveryTable.at(i) == 0){
                        simulator.initiateNegativeAcknowledgement(i);
                    }
                }

                /** timer implemented to avoid frequent negative responses **/
                timer = 0;
            }

            timer++;
            simulator.tick();
        }
        return 0;
    }
```



Hold and Wait

Sender Code

```
/**
 *@author: YashKumarVerma
 *@desc: main entry point of application
 *@time: Tuesday 30-March-2021 01:18:27
 **/

#include<iostream>
#include<vector>
#include"./../includes/rainbow/rainbow.h"
#include"./../includes/simulator/simulator.h"
#include"./../includes/log/log.h"

using namespace std;


/** to facilitate creating problems in transmission based on frequency **/
bool problemCreator(float errorRate){
    int number = rand()%(11);
    return number < errorRate;
}


int main(){
    srand(time(0));

    /** set the fraction of error rate. This is applied on all operations
**/
    int error_rate = 3;
    int total_packets = 10;
    float simulator_frequency = 1.0;


    /** all configurations end here, now application codebase **/
    Simulator simulator(total_packets, "sender");
    simulator.setClockFrequency(simulator_frequency);
    simulator.setErrorRate(error_rate);


    int counter = 1;
    while(simulator.senderTransmissionNotComplete()){

       if(simulator.waiting == false){
            /** this simulates packet loss **/
            if(problemCreator(simulator.errorRate)){
                log::sender_error("Packet loss encountered, packet # " +
to_string(counter));
                counter++;
            }

            simulator.sendDataPacket(counter++);
            simulator.halt();
       }
       simulator.acceptAcknowledgementIfExist(counter-1);
```

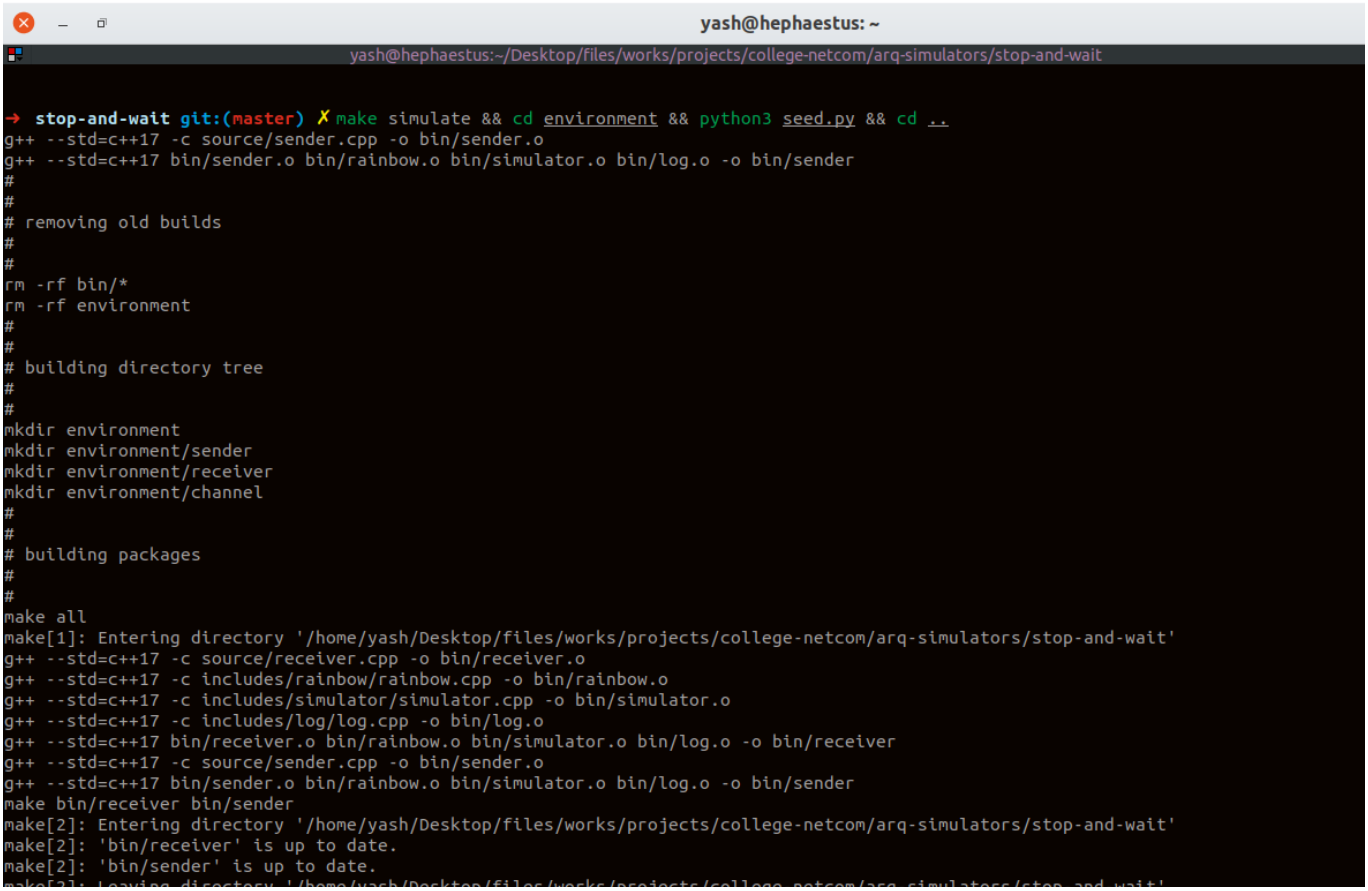
```
        // tick the process cycle
        simulator.tick();
    }
    return 0;
}
```

Receiver Code

```
/**
 *@author: YashKumarVerma
 *@desc: main entry point of application
 *@time: Tuesday 30-March-2021 01:18:27
 **/

#include<iostream>
#include<vector>
#include"./../includes/rainbow/rainbow.h"
#include"./../includes/simulator/simulator.h"
#include"./../includes/log/log.h"

using namespace std;


/** to facilitate creating problems in transmission based on frequency **/
bool problemCreator(float errorRate){
    int number = rand()%(11);
    return number < errorRate;
}


int main(){
    srand(time(0));

    /** simulator configurations **/
    int error_rate = 3;
    int total_packets = 10;
    float simulator_frequency = 1.0;


    Simulator simulator(total_packets, "receiver");
    simulator.setClockFrequency(simulator_frequency);
    simulator.setErrorRate(error_rate);

    int counter = 1;
    while(simulator.receiverTransmissionNotComplete()){

        /** simulate acknowledgement loss **/
        if(problemCreator(simulator.errorRate)){
            log::receiver_error("acknowledgement lost for packet # " +
to_string(counter));
        }
```

```
            // check if there is any packet received
            if(simulator.sendAcknowledgementForPacketIfExist(counter)){
                // expect next item now
                counter++;
            }
            simulator.tick();
        }
        return 0;
    }
```

Installation

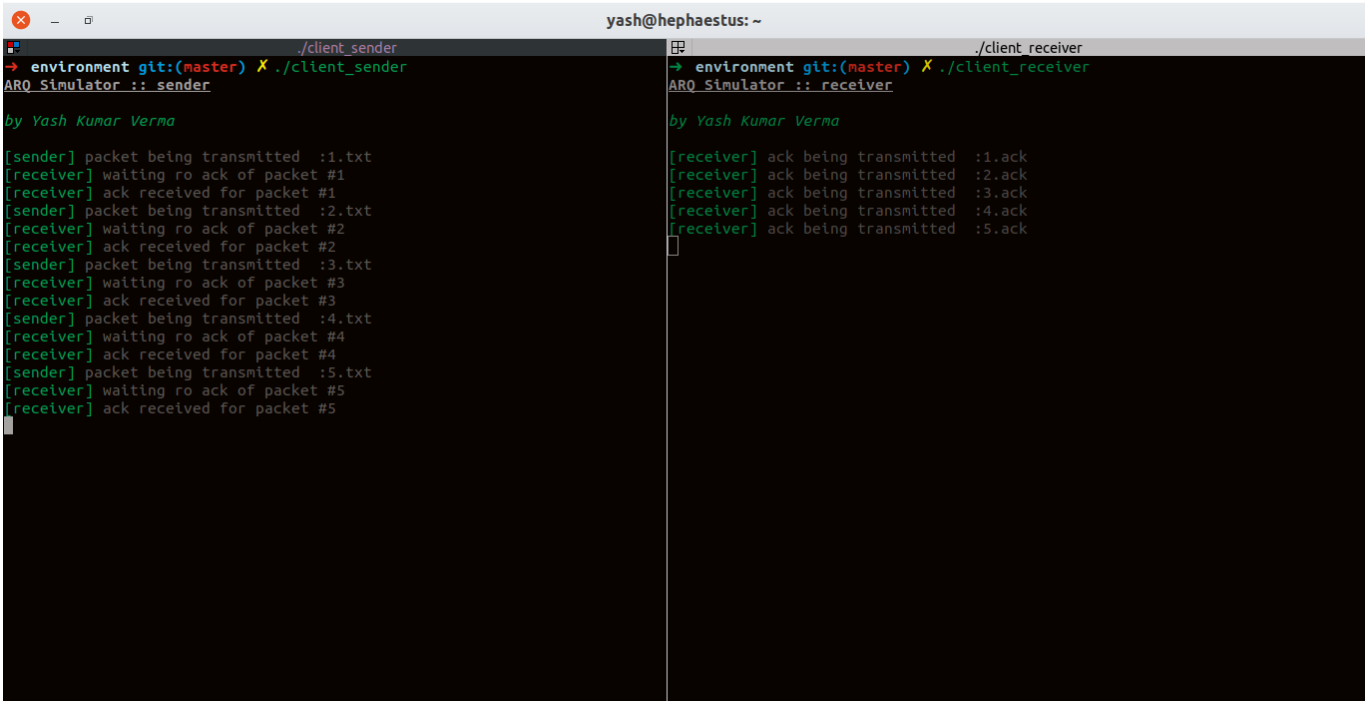


```
yash@hephaestus: ~
yash@hephaestus:~/Desktop/files/works/projects/college-netcom/arq-simulators/stop-and-wait
g++ --std=c++17 bin/receiver.o bin/rainbow.o bin/simulator.o bin/log.o -o bin/receiver
g++ --std=c++17 -c source/sender.cpp -o bin/sender.o
g++ --std=c++17 bin/sender.o bin/rainbow.o bin/simulator.o bin/log.o -o bin/sender
make bin/receiver bin/sender
make[2]: Entering directory '/home/yash/Desktop/files/works/projects/college-netcom/arq-simulators/stop-and-wait'
make[2]: 'bin/receiver' is up to date.
make[2]: 'bin/sender' is up to date.
make[2]: Leaving directory '/home/yash/Desktop/files/works/projects/college-netcom/arq-simulators/stop-and-wait'
make[1]: Leaving directory '/home/yash/Desktop/files/works/projects/college-netcom/arq-simulators/stop-and-wait'
#
#
# take artifacts into environment
#
#
cp bin/sender environment/client_sender
cp bin/receiver environment/client_receiver
cp seed.py environment/seed.py
#
#
# Enter environment
# python3 seed.py
# ./client_sender
#
#
Total 0 packets removed from ./receiver
Total 0 packets removed from ./sender
Total 0 packets removed from ./channel
Generating network packet 1
Generating network packet 2
Generating network packet 3
Generating network packet 4
Generating network packet 5
Generating network packet 6
Generating network packet 7
Generating network packet 8
Generating network packet 9
Generating network packet 10
Execution time: 0h:00m:04s sec
→ stop-and-wait git:(master) ✗
```

Output

```
yash@hephaestus: ~
./client_sender
→ environment git:(master) ✗ ./client_sender
ARQ Simulator :: sender

by Yash Kumar Verma

[sender] packet being transmitted  :1.txt
[receiver] waiting ro ack of packet #1
[receiver] ack received for packet #1
[sender] packet being transmitted  :2.txt
[receiver] waiting ro ack of packet #2
[receiver] ack received for packet #2
```

```
./client_receiver
→ environment git:(master) ✗ ./client_receiver
ARQ Simulator :: receiver

by Yash Kumar Verma

[receiver] ack being transmitted  :1.ack
[receiver] ack being transmitted  :2.ack
```


yash@hephaestus: ~
./client_sender
➜ environment git:(master) ✗ ./client_sender
ARQ Simulator :: sender
by Yash Kumar Verma
[sender] packet being transmitted :1.txt
[receiver] waiting ro ack of packet #1
[receiver] ack received for packet #1
[sender] packet being transmitted :2.txt
[receiver] waiting ro ack of packet #2
[receiver] ack received for packet #2
[sender] packet being transmitted :3.txt
[receiver] waiting ro ack of packet #3
[receiver] ack received for packet #3
[sender] packet being transmitted :4.txt
[receiver] waiting ro ack of packet #4
[receiver] ack received for packet #4
[sender] packet being transmitted :5.txt
[receiver] waiting ro ack of packet #5
[receiver] ack received for packet #5
./client_receiver
➜ environment git:(master) ✗ ./client_receiver
ARQ Simulator :: receiver
by Yash Kumar Verma
[receiver] ack being transmitted :1.ack
[receiver] ack being transmitted :2.ack
[receiver] ack being transmitted :3.ack
[receiver] ack being transmitted :4.ack
[receiver] ack being transmitted :5.ack