CSE351s: Computer Networks

Networks Project – Go Back N Protocol

Team Members

Name	Id	section
Beshoy Ashraf Faheem	1805453	2
Engy Fayek Adeeb	1806935	4
Kirollos Ashraf Sedky	1807624	3
Susanna Michael Faheem	1809355	2
Kirollos Medhat Massoud	1809488	3

GitHub Repo Link

https://github.com/kirollosashrafsedky/nodejs---Go-Back-N-protocol

Languages used

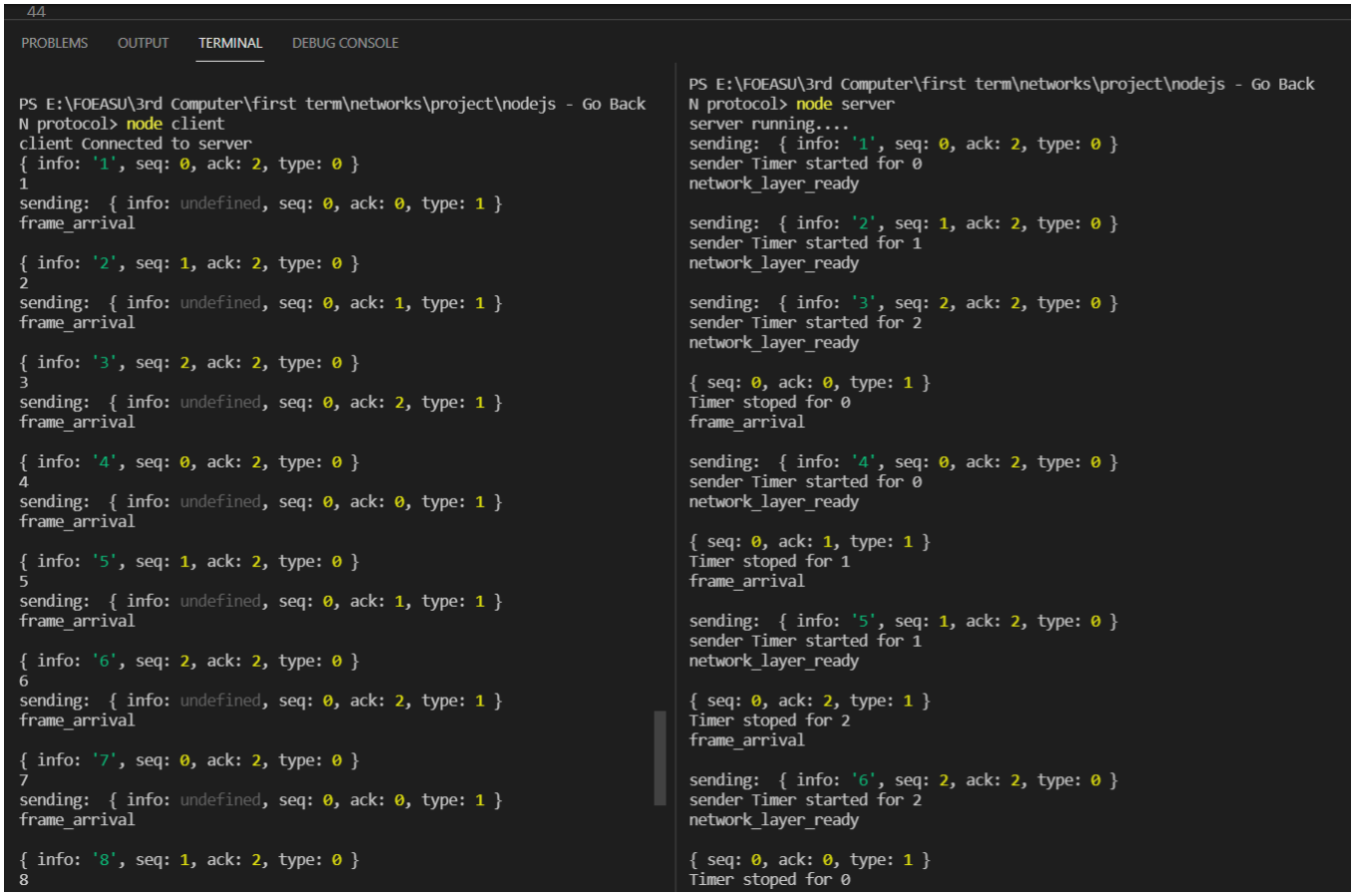
In this project we used Node.js along with socket.io to run two programs in parallel, one acting as the server (sender) and the other as the client (receiver)

- To run the program in the above repo.
  1. Clone the repo to the local machine
  2. Make sure node.js is installed https://nodejs.org/en/
  3. Run "npm install", to install the required packages in the package.json file
  4. Open two terminals, one for the server and the other for the client
  5. Run in the first one "node server" and in the second one "node client"

Screenshots of the program when no error happens

In the following program we are using MAX_SEQ = 2, which is a window size of 3, and the data we are sending is 1,2,3,4,5,6,7,8,9…
Here the code runs where no error happens so every data sent from the server is received at the client successfully
To the right is the server, and to the left is the client sending acks,

The server sends data each second "this time is for demonstration purpose only" and it takes time till it reaches the receiver "to resemble the actual delay in real life" then an ack is sent from the receiver with the same delay, thus after the server sends the third data, the ack for the first data will arrive, then the server will send the forth data, then an ack for data 1 will arrive and so on
Every time the server sends data a timer for that data is started and on receiving an ack for that data the timer is stopped, in this case since no error occurs, the timer is always stopped on receiving an ack



Screenshots of the program when an error occurs

Here in this program we added this portion of code

```
// fail to send the 5th data each time
if ((nthDataToSend % 5 != 0 && isSender) || !isSender)
    to_physical_layer(s);
else
    console.log("error sending" + buffer[frame_nr])
if(s.type == frameTypeData)
    start_timer(frame_nr);
```

this is placed in the send_data() function, it resembles an error in sending the fifth data each

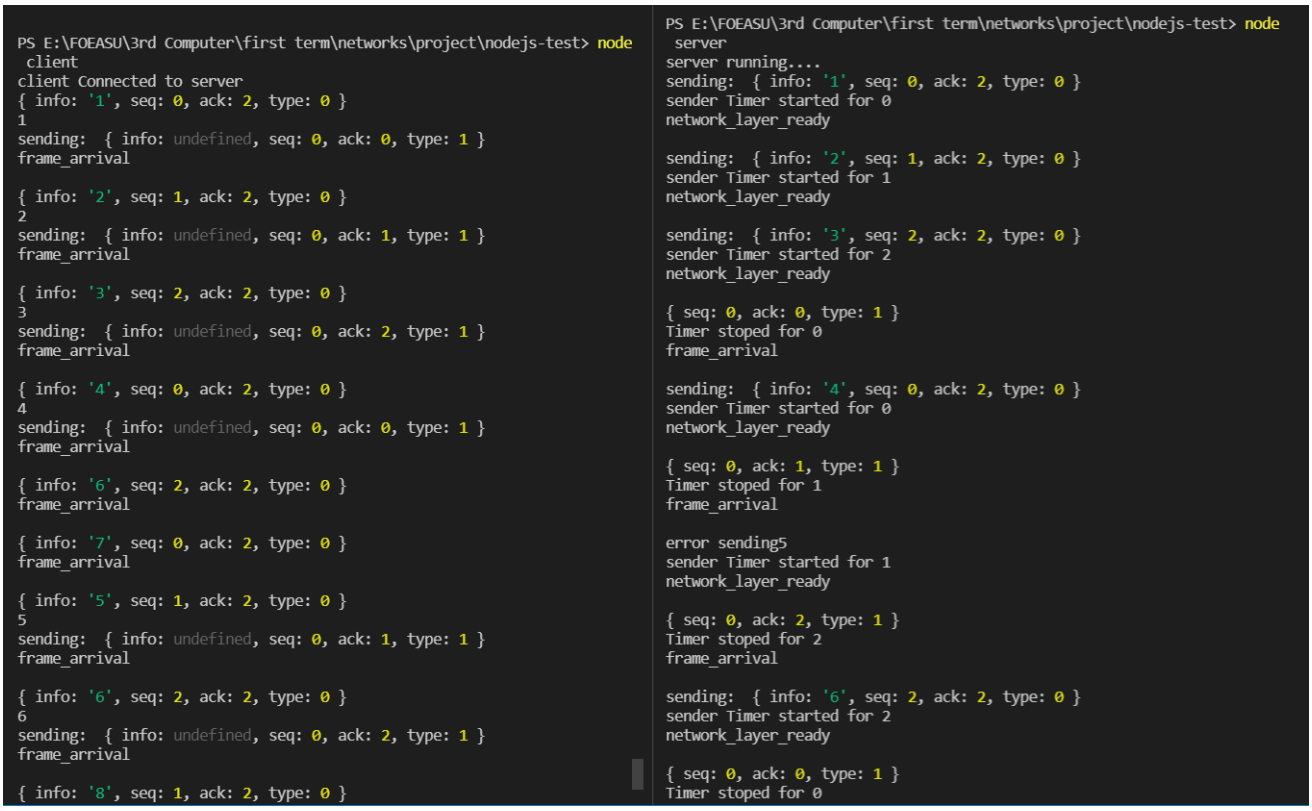
time, the data no. 5 or its multiples. This is from the server side, but from the client the ack is always send and received by the server, of course this could also be changed

Here, the fifth data is lost "error sending5" is shown on the console in the server side, and in the client it stops sending acks after the 4^{th} data, till data is retransmitted again after the timeout of the timer of the data lost.

And it's shown that the client side starts to send acks again after the data transmitted = the data sequence expected.

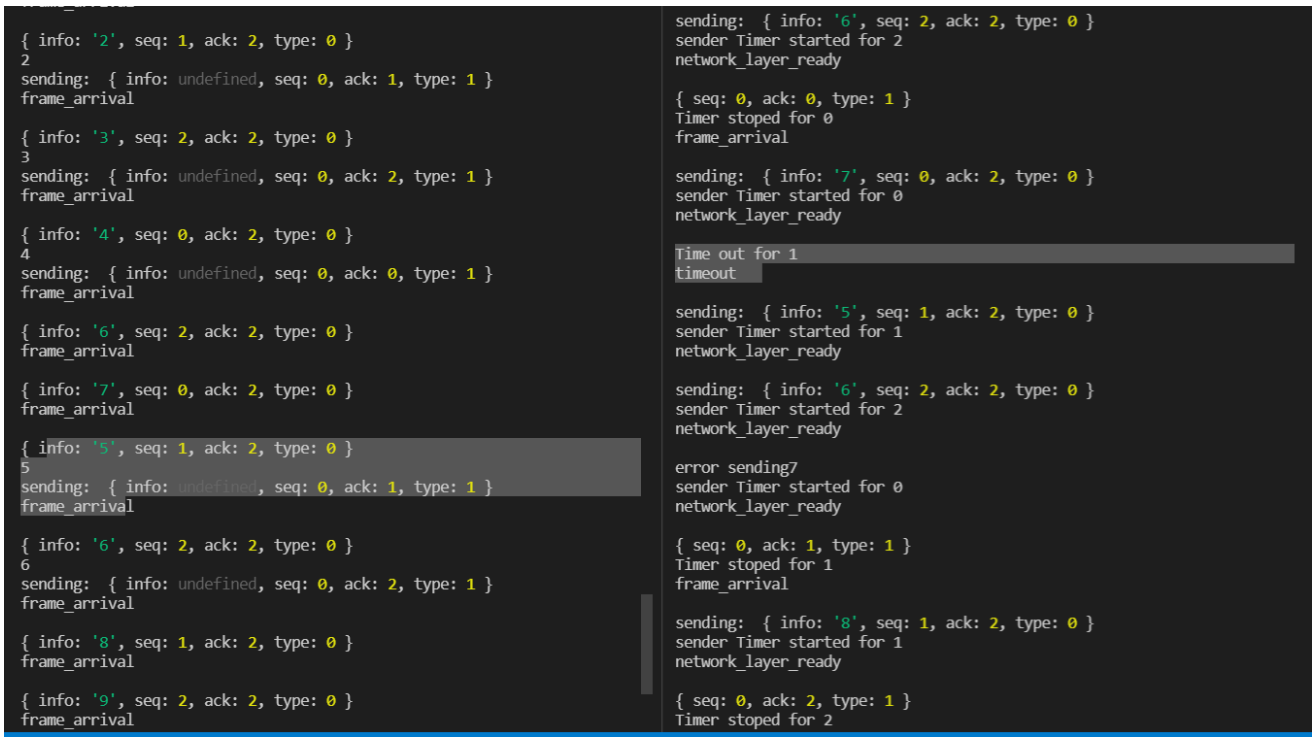
And to make sure that this scenario is right, We counted the number of times the data is sent from the server till data no.9 is transmitted successfully and we found it to be 16 times, exactly as this video demonstrates https://www.youtube.com/watch?v=cqPWjo2iLgk

Were we used 3 window size and the fifth data is lost just as the question in the video



```
PS E:\FOEASU\3rd Computer\first term\networks\project\nodejs-test> node
 client
client Connected to server
{ info: '1', seq: 0, ack: 2, type: 0 }
1
sending:  { info: undefined, seq: 0, ack: 0, type: 1 }
frame_arrival

{ info: '2', seq: 1, ack: 2, type: 0 }
2
sending:  { info: undefined, seq: 0, ack: 1, type: 1 }
frame_arrival

{ info: '3', seq: 2, ack: 2, type: 0 }
3
sending:  { info: undefined, seq: 0, ack: 2, type: 1 }
frame_arrival

{ info: '4', seq: 0, ack: 2, type: 0 }
4
sending:  { info: undefined, seq: 0, ack: 0, type: 1 }
frame_arrival

{ info: '6', seq: 2, ack: 2, type: 0 }
frame_arrival

{ info: '7', seq: 0, ack: 2, type: 0 }
frame_arrival

{ info: '5', seq: 1, ack: 2, type: 0 }
5
sending:  { info: undefined, seq: 0, ack: 1, type: 1 }
frame_arrival

{ info: '6', seq: 2, ack: 2, type: 0 }
6
sending:  { info: undefined, seq: 0, ack: 2, type: 1 }
frame_arrival

{ info: '8', seq: 1, ack: 2, type: 0 }
```

```
PS E:\FOEASU\3rd Computer\first term\networks\project\nodejs-test> node
 server
server running....
sending:  { info: '1', seq: 0, ack: 2, type: 0 }
sender Timer started for 0
network_layer_ready

sending:  { info: '2', seq: 1, ack: 2, type: 0 }
sender Timer started for 1
network_layer_ready

sending:  { info: '3', seq: 2, ack: 2, type: 0 }
sender Timer started for 2
network_layer_ready

{ seq: 0, ack: 0, type: 1 }
Timer stoped for 0
frame_arrival

sending:  { info: '4', seq: 0, ack: 2, type: 0 }
sender Timer started for 0
network_layer_ready

{ seq: 0, ack: 1, type: 1 }
Timer stoped for 1
frame_arrival

error sending5
sender Timer started for 1
network_layer_ready

{ seq: 0, ack: 2, type: 1 }
Timer stoped for 2
frame_arrival

sending:  { info: '6', seq: 2, ack: 2, type: 0 }
sender Timer started for 2
network_layer_ready

{ seq: 0, ack: 0, type: 1 }
Timer stoped for 0
```



```
{ info: '2', seq: 1, ack: 2, type: 0 }
2
sending:  { info: undefined, seq: 0, ack: 1, type: 1 }
frame_arrival

{ info: '3', seq: 2, ack: 2, type: 0 }
3
sending:  { info: undefined, seq: 0, ack: 2, type: 1 }
frame_arrival

{ info: '4', seq: 0, ack: 2, type: 0 }
4
sending:  { info: undefined, seq: 0, ack: 0, type: 1 }
frame_arrival

{ info: '6', seq: 2, ack: 2, type: 0 }
frame_arrival

{ info: '7', seq: 0, ack: 2, type: 0 }
frame_arrival

{ info: '5', seq: 1, ack: 2, type: 0 }
5
sending:  { info: undefined, seq: 0, ack: 1, type: 1 }
frame_arrival

{ info: '6', seq: 2, ack: 2, type: 0 }
6
sending:  { info: undefined, seq: 0, ack: 2, type: 1 }
frame_arrival

{ info: '8', seq: 1, ack: 2, type: 0 }
frame_arrival

{ info: '9', seq: 2, ack: 2, type: 0 }
frame_arrival
```

```
sending:  { info: '6', seq: 2, ack: 2, type: 0 }
sender Timer started for 2
network_layer_ready

{ seq: 0, ack: 0, type: 1 }
Timer stoped for 0
frame_arrival

sending:  { info: '7', seq: 0, ack: 2, type: 0 }
sender Timer started for 0
network_layer_ready

Time out for 1
timeout

sending:  { info: '5', seq: 1, ack: 2, type: 0 }
sender Timer started for 1
network_layer_ready

sending:  { info: '6', seq: 2, ack: 2, type: 0 }
sender Timer started for 2
network_layer_ready

error sending7
sender Timer started for 0
network_layer_ready

{ seq: 0, ack: 1, type: 1 }
Timer stoped for 1
frame_arrival

sending:  { info: '8', seq: 1, ack: 2, type: 0 }
sender Timer started for 1
network_layer_ready

{ seq: 0, ack: 2, type: 1 }
Timer stoped for 2
```

Structure of the project

- server.js
- client.js
- gobackn.js => contains the actual library code called in both the server and the client

Functions used

- inc()
- send_data()
- to_physical_layer()
- from_physical_layer()
- from_network_layer()
- to_network_layer()
- enableNetworkLayer()
- disableNetworkLayer()
- start_timer()
- stop_timer()
- getEvent()
- goBackN_init()
- goBackN_sender()
- goBackN_receiver()