NETWORKS PROJECT REPORT “I<3NETWORKS”

**Jana Alaa El-Owainy:** 55-0593 T08 [jana.elewainy@student.guc.edu.eg](mailto:jana.elewainy@student.guc.edu.eg)

Contribution; shared in writing the pseudocode and implemented the bonus

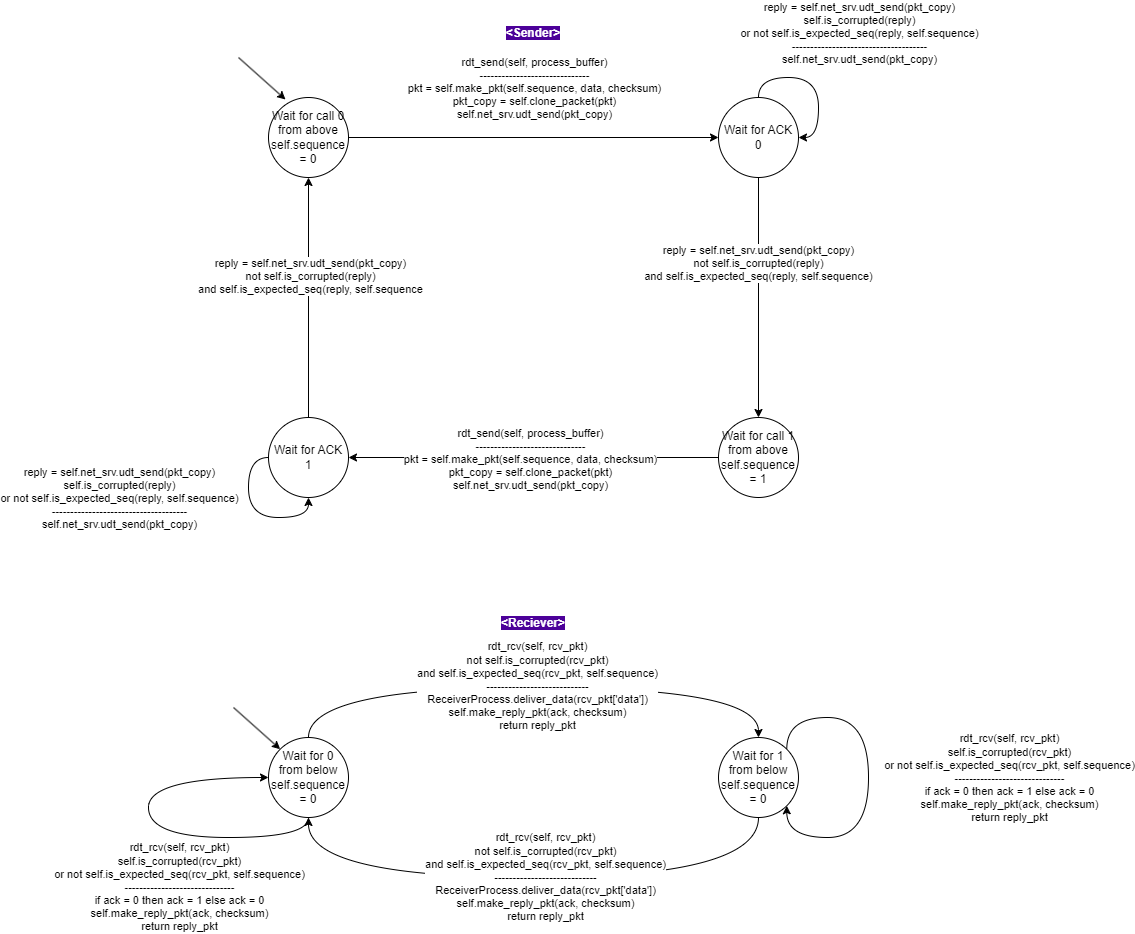
**Ziad Mohamed Al-Gendy**: 55-3395 T08 [ziad.elgendy@student.guc.edu.eg](mailto:ziad.elgendy@student.guc.edu.eg)

Contribution; shared in writing the pseudocode and implementing python code

**Yara Mohamed Al-Tantawy:** 55-24812 T08 [yara.eltantawy@student.guc.edu.eg](mailto:yara.eltantawy@student.guc.edu.eg)

Contribution; shared in writing the pseudocode and made the report

**FSM Diagram**

****

**Pseudocode**

**<Sender>**

//Add the colors library

CLASS SenderProcess{

Create a list store it in \_\_buffer

STATIC FUNCTION set\_outgoing\_data(buffer){

//To set the message the process would send out over the network

Set \_\_buffer of SenderProcess to be buffer (input)

RETURN

}

STATIC FUNCTION get\_outgoing\_data(){

// to get the message the process would send out over the network

RETURN \_\_buffer of SenderProcess

}

}

CLASS RDTSender{

//RDT V2.2 Sender side implementation

//class construction

Initialization(self, net\_srv){

self.sequence = ‘0’ //initialize sequence number as ‘0’

self.net\_srv = net\_srv //and net\_srv as the second input

}

STATIC FUNCTION get\_checksum(data){

// Calculate the checksum for outgoing data

checksum = ASCII conversion of data (input)

RETURN checksum

}

STATIC FUNCTION clone\_packet(packet){

//Make a copy of the outgoing packet

//Define a new dictionary called pkt\_clone and insert all attributes of packet into it

Pkt\_clone = {

‘sequence\_number’ of Pkt\_clone = sequence\_number’ of packet,

‘data’ of Pkt\_clone : ‘data’ of packet,

‘checksum’ of Pkt\_clone : ‘checksum’ of packet

}

RETURN Pkt\_clone

}

STATIC FUNCTION Is\_corrupted (reply) {

// Check if the received reply from receiver is corrupted or not

IF (the ‘checksum’ from the input reply dictionary == the ASCII conversion of the string conversion of ‘ack’ from reply dictionary)

RETURN ‘False’

ELSE {

Display ‘Sender: Incorrect checksum’ in red

RETURN ‘True’

}

}

STATIC FUNCTION is\_expected\_seq(reply, exp\_seq){

// Check if the received reply from receiver has the expected sequence number

IF (the ‘ack’ from the first input reply dictionary == exp\_seq, the second input)

RETURN ‘True’

ELSE

Display ‘Sender: Incorrect checksum number’ in red

RETURN ‘False’

}

STATIC FUNCTION make\_pkt(seq, data, checksum){

//Takes as input sequence number, data and checksum and uses them to make a new dictionary named packet to be sent to receiver

packet = {

//having attributes ‘sequence\_number’, ‘ data’ and ‘checksum’

‘sequence\_number’ = seq,

‘ data’ = data,

‘checksum’ = checksum

}

RETURN packet

}

STATIC FUNCTION rdt\_send(self, process\_buffer){

FOR (every character (data) in the buffer (process\_buffer)){

checksum = the return of function get\_checksum(data) on self

pkt = the returned packet of function make.pkt(self.sequemce, data, checksum)

Display ‘Sender is sending: {pkt}’ in green on a new line

pkt\_copy = the returned packet of function self.clone\_packet(pkt)

reply = the returned packet of function self.net\_srv.udt\_send(pkt\_copy)

Display ‘Sender expected seq number: {self.sequence}’ in green

Display ‘Sender received : {reply}’ in green

WHILE ( self.is\_corrupted(reply) returns True OR self.is\_expected\_seq(reply, self.sequence) returns False ){

Display ‘Sender is resending: {pkt}, in green

Display ‘Sender expected seq number: {self.sequence}’ in green

pkt\_copy = the return packet from function self.clone\_packet(pkt)

reply = the returned packet of function self.net\_srv.udt\_send(pkt\_copy)

Display ‘sender received : {reply}’ in green

}

CASE OF self.sequence {

‘0’ : self.sequence = ‘1’

‘1’ : self.sequence = ‘0’

}

}

Display ‘Sender Done!’

RETURN //NOTHING

}

}

**<Receiver>**

//Add the colors library

CLASS ReceiverProcess{

Create a list store it in \_\_buffer

STATIC FUNCTION deliver\_data(data){

// deliver data from the transport layer RDT receiver to the application layer

APPEND the input data to end of \_\_buffer in ReceiverProcess

RETURN //NOTHING

}

STATIC FUNCTION get\_buffer(){

RETURN \_\_buffer from ReceiverProcess

}

}

CLASS RDTReceiver{

//RDT V2.2 Receiver side implementation

//class construction

Initialization(self){

self.sequence = ‘0’

}

STATIC FUNCTION Is\_corrupted (packet){

// Check if the received packet from sender is corrupted or not

IF (‘checksum’ from the input packet dictionary == the ASCII conversion of the string conversion of ‘data’ from packet dictionary)

RETURN False

ELSE{

Display ‘Receiver: Incorrect checksum’ in red

RETURN True

}

}

STATIC FUNCTION is\_expected\_seq(rcv\_pkt, exp\_seq){

// Check if the received reply from receiver has the expected sequence number

IF (the ‘sequence\_number’ from the first input rcv\_pkt dictionary == exp\_seq, the second input)

RETURN True

ELSE{

Display ‘Receiver: Incorrect checksum number’ in red

RETURN False

}

}

STATIC FUNCTION make\_reply\_pkt(seq, checksum){

// Create a reply (feedback) packet with to acknowledge the received packet with attribute ‘ack’ and ‘checksum’

reply\_pck = {

‘ack’ : seq,

‘checksum’ : checksum

}

RETURN reply\_pck

}

STATIC FUNCTION rdt\_rcv(self, rcv\_pkt){

// Implement the RDT v2.2 for the receiver

Display ‘Receiver expected seq number: {self.sequence}’ in blue

Display ‘Receiver received: {rcv\_pkt}’ in blue

ack = self.sequence

IF (self.is\_corrupted(rcv\_pkt) returns True OR self.is\_expected \_seq(rcv\_pkt, self.sequence) returns False){

CASE OF ack{

‘0’ : ack = ‘1’

‘1’ : ack = ‘0’

}

reply\_pkt = self.make\_reply\_pkt(ack, conversion of ASCII of String of ack)

Display ‘Receiver is sending: {reply\_pkt}’ in blue

RETURN reply\_pkt

}

ELSE{

CASE OF self.sequence{

‘0’ : self.sequence = ‘1’

‘1’ : self.sequence = ‘0’

}

Call function deliver\_data(rcv\_pkt[‘data’] on data from packet rcv\_pkt) from ReceiverProcess class

reply\_pkt = self.make\_reply\_pkt(ack, conversion of ASCII of String of ack)

Display ‘Receiver is sending: {reply\_pkt}’ in blue

RETURN reply\_pkt

}

}

}

**Changes to original code**

**> Import os**

**> os.system(‘ ’)**

In main.py; To allow outputs in the Windows Command Line to be printed in color.

**Test Cases**



