packet.py

```
"""Packet"""

class Packet():
    def __init__(self, magicno, packType, seqno, dataLen, data):
        self.magicno = magicno
        self.packType = packType
        self.seqno = seqno
        self.dataLen = dataLen
        self.data = data
```

sender.py

```
"""Sender"""

import socket   #for sockets
import sys      #for exit
import os.path  #for checking filename
import pickle   #for converting packet to and from bytes
import packet

HOST = "127.0.0.1"  #localhost
MAGIC_NO = 0x497E
DATA_PACKET = 0
ACK_PACKET = 1



def check_port_num(port_num):
    """Requests port number from user and checks it conforms to requirements"""
    if port_num <= 1024 or port_num  >= 64000:
        sys.exit()

try:
    port_s_in = int(sys.argv[1])
    port_s_out = int(sys.argv[2])
    port_cs_in = int(sys.argv[3])
    filename = sys.argv[4]
except ValueError:
    sys.exit()

check_port_num(port_s_in)
check_port_num(port_s_out)
check_port_num(port_cs_in)

# Getting filename of file to be sent and checking it exists:
if os.path.isfile(filename) is False:
    print("File does not exist. Exiting Sender program.\n")
    sys.exit()


    # Creating dgram udp sockets:
try:
    s_in =socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s_out = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
except socket.error:
    print("Failed to create socket")
    sys.exit()

# Binding both ports:
s_in.bind((HOST, port_s_in))
s_out.bind((HOST, port_s_out))

# Connecting s_out to cs_in:
s_out.connect((HOST, port_cs_in))

# Setting timeout:
s_in.settimeout(1.0)



# Converting file into bytes:
original_file = open(filename, 'rb')
remaining_bytes = original_file.read()

# Initialising local variables:
next_seq_no = 0
packets_sent = 0
exitFlag = False

# Processing file, 512 bytes at a time and placing into packets
```

```
while 1:
    if len(remaining_bytes) >= 512:
        n = 512
    else:
        n = len(remaining_bytes)
    if n == 0:
        new_packet = packet.Packet(MAGIC_NO, DATA_PACKET, next_seq_no, 0, None)
        exitFlag = True
    else:
        data_buffer = remaining_bytes[:n]
        new_packet = packet.Packet(MAGIC_NO, DATA_PACKET, next_seq_no, n, data_buffer)
        remaining_bytes = remaining_bytes[n:]

    # Sending each packet:
    packet_buffer = pickle.dumps(new_packet)
    acknowledged = False
    while acknowledged is False:
        s_out.send(packet_buffer)
        packets_sent += 1
        try:
            rcvd_tuple = s_in.recvfrom(1024)
            rcvd = pickle.loads(rcvd_tuple[0])
            if rcvd.magicno == MAGIC_NO and rcvd.packType == ACK_PACKET and rcvd.dataLen == 0:
                if rcvd.seqno == next_seq_no:
                    next_seq_no = 1 - next_seq_no
                    acknowledged = True
                    if exitFlag is True:
                        original_file.close()
                        s_in.close()
                        s_out.close()
                        print("Number of packets sent is: " + str(packets_sent))
                        sys.exit()
        except socket.timeout:
            continue
```

channel.py

```
"""Channel"""

import socket
import sys
import random
import select
import pickle
import packet

HOST = "127.0.0.1"  #localhost
MAGIC_NO = 0x497E



def check_port_num(port_num):
    """Requests port number from user and checks it conforms to requirements"""
    if port_num <= 1024 or port_num  >= 64000:
        sys.exit()



try:
    port_cs_in = int(sys.argv[1])
    port_cs_out = int(sys.argv[2])
    port_cr_in = int(sys.argv[3])
    port_cr_out = int(sys.argv[4])
    port_s_in = int(sys.argv[5])
    port_r_in = int(sys.argv[6])
    P = float(sys.argv[7])

except ValueError:
    sys.exit()

check_port_num(port_cs_in)
check_port_num(port_cs_out)
check_port_num(port_cr_in)
check_port_num(port_cr_out)
check_port_num(port_s_in)
check_port_num(port_r_in)

# Getting P value
signal = 0
while signal == 0:
    if P >= 0 and P < 1:
        signal = 1

# Creating sockets:
try :
    cs_in = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    cs_out = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    cr_in = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    cr_out = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
except socket.error as err_msg :
    print("Failed to create a socket. Error Code : " + str(err_msg[0]) + " Message " + err_msg[1])
    sys.exit()

# Binding sockets:
cs_in.bind((HOST, port_cs_in))
cs_out.bind((HOST, port_cs_out))
cr_in.bind((HOST, port_cr_in))
cr_out.bind((HOST, port_cr_out))

# Connecting sockets:
cs_out.connect((HOST, port_s_in))
cr_out.connect((HOST, port_r_in))

input_sockets = [cs_in, cr_in]

# Entering infinite loop:
while 1:
    input_ready, output_ready, except_ready = select.select(input_sockets,[],[])
```

```
    for sock in input_ready:
        if sock == cs_in:
            rcvd_tuple = cs_in.recvfrom(1024)
            rcvd = pickle.loads(rcvd_tuple[0])
            if rcvd.magicno == MAGIC_NO:
                u = random.random()
                if u < P:
                    continue
                else:
                    cr_out.send(rcvd_tuple[0])
        elif sock == cr_in:
            rcvd_tuple = cr_in.recvfrom(1024)
            rcvd = pickle.loads(rcvd_tuple[0])
            if rcvd.magicno == MAGIC_NO:
                u = random.random()
                if u < P:
                    continue
                else:
                    cs_out.send(rcvd_tuple[0])
```

receiver.py

```
"""Receiver"""

import socket   #for sockets
import sys      #for exit
import os.path  #for checking filename
import pickle   #for converting packet to and from bytes
import packet
import select

HOST = "127.0.0.1"  #localhost
MAGIC_NO = 0x497E
DATA_PACKET = 0
ACK_PACKET = 1



def check_port_num(port_num):
    """Requests port number from user and checks it conforms to requirements"""
    if port_num <= 1024 or port_num  >= 64000:
        sys.exit()



try:
    port_r_in = int(sys.argv[1])
    port_r_out = int(sys.argv[2])
    port_cr_in = int(sys.argv[3])
    filename = sys.argv[4]

except ValueError:
    sys.exit()

check_port_num(port_r_in)
check_port_num(port_r_out)
check_port_num(port_cr_in)


# Creating dgram udp sockets:
try:
    r_in =socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    r_out = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
except socket.error:
    print("Failed to create socket")
    sys.exit()

# Binding both ports:
r_in.bind((HOST, port_r_in))
r_out.bind((HOST, port_r_out))

# Connecting s_out to cs_in:
r_out.connect((HOST, port_cr_in))

# Getting filename of file to be copied to and checking it doesn't already exist:

if os.path.isfile(filename) is True:
    print("File already exists. Exiting Receiver program.\n")
    sys.exit()

# Opening new file:
new_file = open(filename, 'wb')
```

```
exp_seq_no = 0
input_sockets = [r_in]

# Entering loop:
while 1:
    input_ready, output_ready, except_ready = select.select(input_sockets,[],[])
    d_new = r_in.recvfrom(1024)
    rcvd = pickle.loads(d_new[0])
    if rcvd.magicno == MAGIC_NO and rcvd.packType == DATA_PACKET:
        new_packet = packet.Packet(MAGIC_NO, ACK_PACKET, rcvd.seqno, 0, None)
        packet_buffer = pickle.dumps(new_packet)
        r_out.send(packet_buffer)
        if rcvd.seqno == exp_seq_no:
            exp_seq_no = 1 - exp_seq_no
            if rcvd.dataLen > 0:
                new_data = rcvd.data
                new_file.write(new_data)
            else:
                new_file.close()
                r_in.close()
                r_out.close()
                sys.exit()
```