```python
""" Channel program for Cosc264 Assignment

    Authors: Josh Bernasconi 68613585
             James Toohey    27073776
"""

import random
import select
import socket
import sys
import time

from helpers import *


def channel(CSin_port, CSout_port, CRin_port, CRout_port, Sin_port, Rin_port, Precision):
    """ Checks ports, sets up connections, then hands over to the main loop """

    ports_ok = check_ports(CSin_port, CSout_port, CRin_port, CRout_port, Sin_port, Rin_port)

    if ports_ok:
        print("Port numbers all valid\n")
    else:
        print("There is a problem with the supplied port numbers!\n Exiting")
        sys.exit()

    # Socket init
    CSin = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    CSout = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    CRin = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    CRout = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Bind
    try:  # Catching errors binding the ports
        print("Binding port CSin")
        CSin.bind(('localhost', CSin_port))
        print("CSin successfully bound\n")
        print("Binding port CSout")
        CSout.bind(('localhost', CSout_port))
        print("CSout successfully bound\n")
        CRin.bind(('localhost', CRin_port))
        print("CRin successfully bound\n")
        CRout.bind(('localhost', CRout_port))
        print("CRout successfully bound\n")
    except socket.error as msg:
        print("Bind failed. Exiting.\n Error: " + str(msg))
        sys.exit()

    # Listen and accept CSin
    CSin.listen(50)
    CSin, _ = CSin.accept()

    print("CSin accepted")

    # Connect CRout to Rin
    connected = False
    connect_attempts = 0
    while not connected:
        try:
            print("Connecting CRout to Rin")
            CRout.connect(('localhost', Rin_port))
            print("Connection successful\n")
            connected = True
        except socket.error as msg:
            connect_attempts += 1
            if msg.errno in [111, 10061] and connect_attempts < 6:
                print("Connection refused {} time(s), sleeping and retrying".format(connect_attempts))
                time.sleep(5)
                pass
            else:
                print("Connect failed. Exiting\n Error: " + str(msg))
                sys.exit()

    # Listen and accept CRin
    CRin.listen(50)
    CRin, _ = CRin.accept()

    # Connect CSout to Sin
    try:
        print("Connecting CSout to Sin")
        CSout.connect(('localhost', Sin_port))
        print("Connection successful\n")
    except socket.error as msg:
        print("Connect failed. Exiting\n Error: " + str(msg))
        sys.exit()

    # Receive, select and send
    process(CSin, CSout, CRin, CRout, CSin_port, CRin_port, Precision)

    CSin.close()
    CSout.close()
    CRin.close()
    CRout.close()
    return None


def bitError(data_in):
    """ Randomly adds/ doesn't add a bit error to the packet"""
    v = random.uniform(0, 1)
    if v < 0.1:
        print("bit error")
        packet, valid = get_packet(data_in)
        if valid:
            new_packet = Packet(packet.pac_type,
                                packet.seqno,
                                packet.data_len + int(random.uniform(1, 10)),
                                packet.data,
                                packet.checksum)
            data_in = pack_data(new_packet)

    return data_in


def process(CSin, CSout, CRin, CRout, CSin_port, CRin_port, Precision):
    """Main infinite loop which receives and processes all packets. Then sends them to the destination"""
    finished = False
    while not finished:  # while CRin doesnt receive terminating packet
        readable, _, _ = select.select([CSin, CRin], [], []) # Blocking call for input
        for sock in readable:
            host, port = sock.getsockname()

            data_in, address = sock.recvfrom(1024)

            if len(data_in) != 0:
                header = get_header_object(data_in)

                if header.magicno != 0x497E:
                    print("Sender Packet magic number != 0x497E, dropping packet.\n")
                    continue
                else:  # potentially drop packet
                    u = random.uniform(0, 1)
                    if u < Precision:  # drop packet
                        print("drop packet")
                        continue
                    else:  # potentially introduce bit error
                        data_in = bitError(data_in)

                    if port == CSin_port:
                        CRout.send(data_in)
                    elif port == CRin_port:
                        CSout.send(data_in)
                    else:  # received from invalid port number
                        print("Invalid port number")
                        continue
            else:
                print("empty data packet received, finished")
                finished = True
                break

    print("Precision {}".format(Precision))

if __name__ == '__main__':

    if len(sys.argv) != 8:
        print("Invalid command.")
        print("Usage: channel.py [CSin port] [CSout port] [CRin port] [CRout port] {Sin port] [Rin port] [Precision]")
    else:
        CSin = int(sys.argv[1])
        CSout = int(sys.argv[2])
        CRin = int(sys.argv[3])
        CRout = int(sys.argv[4])
        Sin = int(sys.argv[5])
        Rin = int(sys.argv[6])
        Precision = float(sys.argv[7])

        channel(CSin, CSout, CRin, CRout, Sin, Rin, Precision)
```