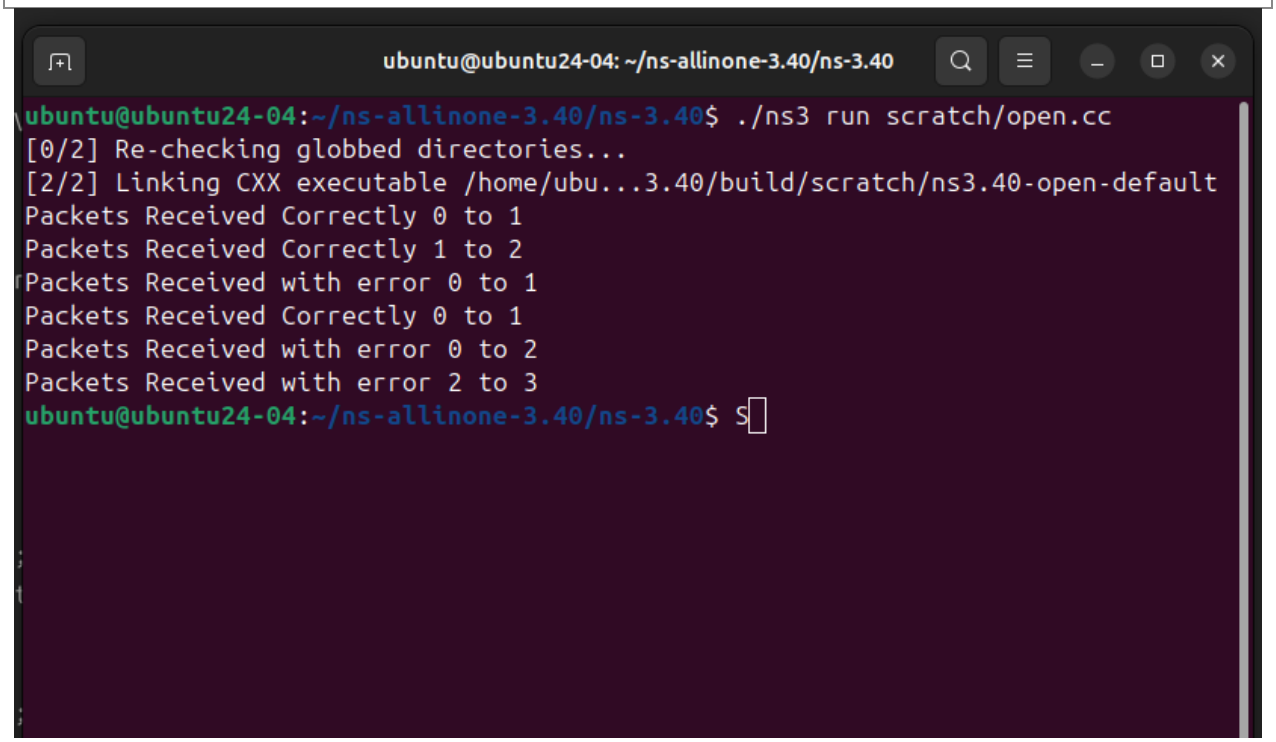
Task 1

```
#include "ns3/object.h"
#include "ns3/simulator.h"
#include "ns3/trace-source-accessor.h"
#include "ns3/traced-value.h"
#include "ns3/uinteger.h"

#include <iostream>

using namespace ns3;

class MyObject : public Object
{
  public:
    /**
     * Register this type.
     * \return The TypeId.
     */
  static TypeId GetTypeId()
    {
        static TypeId tid = TypeId("MyObject")
                                .SetParent<Object>()
                                .SetGroupName("Tutorial")
                                .AddConstructor<MyObject>()
                                .AddTraceSource("MyInteger",
                                                "An integer value to trace.",
                                                MakeTraceSourceAccessor(&MyObject::m_pakOk),
                                                "ns3::TracedValueCallback::Int32")
                               .AddTraceSource("MyIntegerError",
                                                "An integer value to trace.",
                                                MakeTraceSourceAccessor(&MyObject::m_pakError),
                                                "ns3::TracedValueCallback::Int32");

       return tid;

    }

    MyObject()
    {
    }

    TracedValue<int32_t> m_pakOk;
    TracedValue<int32_t> m_pakError;
};

void PacketsOK(int32_t oldValue, int32_t newValue)
{
    std::cout << "Packets Received Correctly "  << oldValue << " to " << newValue << std::endl;
}
void PacketError(int32_t oldValue, int32_t newValue)
{
    std::cout << "Packets Received with error "    << oldValue << " to " << newValue << std::endl;
}


int main(int argc, char* argv[])
{
    Ptr<MyObject> Node1 = CreateObject<MyObject>();
    Node1->TraceConnectWithoutContext("MyInteger", MakeCallback(&PacketsOK));
    Node1->TraceConnectWithoutContext("MyIntegerError", MakeCallback(&PacketError));

    Ptr<MyObject> Node2 = CreateObject<MyObject>();
    Node2->TraceConnectWithoutContext("MyInteger", MakeCallback(&PacketsOK));
    Node2->TraceConnectWithoutContext("MyIntegerError", MakeCallback(&PacketError));
```

```
                Node1->m_pakOk = 1;
                Node1->m_pakOk = 2;
                Node1->m_pakError = 1;

            Node2->m_pakOk = 1;
            Node2->m_pakError = 2;
            Node2->m_pakError = 3;


    return 0;

}
```



Task 2

```
 #include "ns3/core-module.h"
 #include "ns3/network-module.h"
 #include "ns3/internet-module.h"
 #include "ns3/point-to-point-module.h"
 #include "ns3/applications-module.h"

#include <iostream>

using namespace ns3;
using namespace std;

static int32_t packetOk;
static int32_t packetError;
```

```
void pak_Ok(Ptr<const Packet>packet){

  packetOk++;
  cout << "Packets Received Correctly "  << packet->GetSize() <<"bytes"<< endl;

}


void pak_Error(Ptr<const Packet>packet){

  packetError++;
  cout << "Packets Received Correctly "  <<  packet->GetSize() <<"bytes"<< endl;

}


int main(int argc, char *argv[])
{
    LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

      NodeContainer nodes;
      nodes.Create(2);


    PointToPointHelper p2p;
    p2p.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    p2p.SetChannelAttribute("Delay", StringValue("2ms"));

    NetDeviceContainer devices = p2p.Install(nodes);

    // Install Internet stack
    InternetStackHelper stack;
    stack.Install(nodes);

    Ipv4AddressHelper address;
    address.SetBase("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign(devices);

    // Set up UDP Echo Server on Node 1
       UdpEchoServerHelper echoServer(9);
       ApplicationContainer serverApp = echoServer.Install(nodes.Get(1));
       serverApp.Start(Seconds(1.0));
       serverApp.Stop(Seconds(10.0));

        UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 9);
       echoClient.SetAttribute("MaxPackets", UintegerValue(5));
       echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
       echoClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApp = echoClient.Install(nodes.Get(0));

    clientApp.Start(Seconds(2.0));
    clientApp.Stop(Seconds(10.0));

    devices.Get(1)->TraceConnectWithoutContext("PhyRxEnd", MakeCallback(&pak_Ok));

    evices.Get(1)->TraceConnectWithoutContext("PhyRxDrop", MakeCallback(&pak_Error));

    Simulator::Stop(Seconds(10.0));
    Simulator::Run();
    Simulator::Destroy();
```

```
    cout<<"Total received oK "  << packetOk << endl;
    cout<<"Total received with error "  << packetError << endl;

     return 0;
}
```