EXPERIMENT 5

AIM: 1. Create a simple topology of two nodes (Node1, Node2) separated by a p2p link.
2. Setup a UdpClient on one Node1 and a UdpServer on Node2.

NS-3 CODE:

```
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-module.h"

//       n1 ------ n2
//            p2p
//
// - UDP flows from n1 to n2

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("Lab1");

int main(int argc, char* argv[])
{
    double lat = 2.0;
    uint64_t rate = 5000000; // Data rate in bps
    // change it to 2 for sending each packet at interval of 2 (good for viewing in pyviz)
    double interval = 0.05;

    CommandLine cmd;
    cmd.AddValue("latency", "P2P link Latency in miliseconds", lat);
    cmd.AddValue("rate", "P2P data rate in bps", rate);
    cmd.AddValue("interval", "UDP client packet interval", interval);

    cmd.Parse(argc, argv);

    LogComponentEnable("UdpClient", LOG_LEVEL_INFO);
    LogComponentEnable("UdpServer", LOG_LEVEL_INFO);

    // explicitly create the nodes required by the topology (shown above).
    NodeContainer n;
    n.Create(2);

    // explicitly create the channels required by the topology (shown above).
    PointToPointHelper p2p;
    p2p.SetChannelAttribute("Delay", TimeValue(MilliSeconds(lat)));
    p2p.SetDeviceAttribute("DataRate", DataRateValue(DataRate(rate)));
    p2p.SetDeviceAttribute("Mtu", UintegerValue(1400));

    NetDeviceContainer dev = p2p.Install(n);

    // install Internet Stack
    InternetStackHelper internet;
    internet.Install(n);

    Ipv4AddressHelper ipv4;

    ipv4.SetBase("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer i = ipv4.Assign(dev);
```

```
    Ipv4GlobalRoutingHelper::PopulateRoutingTables();

    // create one udpServer application on node one.
    uint16_t port1 = 8000; // need different port numbers to ensure there is no conflict

    UdpServerHelper server1(port1);

    ApplicationContainer apps;
    apps = server1.Install(n.Get(1));

    apps.Start(Seconds(1.0));
    apps.Stop(Seconds(10.0));

    // create one UdpClient application to send UDP datagrams from node zero to node one.

    uint32_t MaxPacketSize = 1024; // size of packet
    Time interPacketInterval = Seconds(interval);
    uint32_t maxPacketCount = 1; // how many packets to send

    UdpClientHelper client1(i.GetAddress(1), port1);

    client1.SetAttribute("MaxPackets", UintegerValue(maxPacketCount));
    client1.SetAttribute("Interval", TimeValue(interPacketInterval));
    client1.SetAttribute("PacketSize", UintegerValue(MaxPacketSize));

    ApplicationContainer Capps = client1.Install(n.Get(0));

    Capps.Start(Seconds(2.0));
    Capps.Stop(Seconds(9.0));

    p2p.EnablePcapAll("udp");

    // now do the actual simulation.
    Simulator::Run();

    Simulator::Destroy();
}
```

OUTPUT

```
>> ./waf --run "examples/udp-client-server/udp-client-server"

Waf: Entering directory `/home/temp/Desktop/ns-allinone-3.28.1/ns-3.28.1/build'
[2533/2737] Linking build/examples/udp-client-server/ns3.28.1-udp-client-server-debug
Waf: Leaving directory `/home/temp/Desktop/ns-allinone-3.28.1/ns-3.28.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.821s)
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 0 Time: 2
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 0 Uid: 0 TXtime: +2000000000.0ns
.
.
.
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 163 Time: 9.95
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 159 Uid: 163 TXtime: +9950000000.0ns
RXtime: +9953711999.0ns Delay: +3711999.0ns
```