**Program to simulate Point to Point Topology.**

**ns3 - nsallinone-3.35 - ns-3.35**

**Examples – Tutorial - First.cc(copy the file)**

**Now again in ns-3.35**

**Scratch(paste first.cc)**

**Go back to ns-3.35—right click open in terminal**

* **./waf - - run first**
* **./waf - - run first - - vis**
* **cd scratch (line 95,98ncsna)**
* **gedit first.cc**

****

* **close terminal(again open terminal\_**

**./waf - - run scratch/first**

**./waf - - run scratch/first --vis**

**OR**

**Now in scratch folder right click - open in terminal**

* **gedit first.cc**

**Now in ns-3.35 open terminal**

* **./waf - - run scratch/first**

**Bus Topology**

**ns-3.35 – Examples – Tutorial - Second.cc**

**line 92 and 98 change port no 9to15 ctr+0 and exit**

**./waf - - run scratch/second**

**Star topology**

**ns3 - nsallinone-3.35 - ns3.35 - examples - Tcp folder**

**star.cc there.Copy the star.cc file to scratch folder**

**Open the terminal in ns3.35**

**./waf - - run star**

**./waf - - run star - - vis**

**Now in ns-3.35 open terminal OR**

**./waf - - run scratch/star cd scratch**

**./waf - - run scratch/star –vis gedit star.cc**

**Line no 57(unit32\_tnspokes = 8)change 8 to 20 save and exit**

**In terminal**

**./waf - - run scratch/star**

**./waf - - run scratch/star –vis**

**Line no 80(unit16\_t port=5000)change 5000 to 3000**

**./waf - - run scratch/star**

**./waf - - run scratch/star –vis**

**Line no 84 and 85(hubApp .Start(seconds(1,0));)change to(2,0)**

**./waf - - run scratch/star**

**./waf - - run scratch/star –vis**

**Line no 49(config change UintegerValue from (137) to (50))**

**./waf - - run scratch/star**

**./waf - - run scratch/star –vis**

**(./waf --run "scratch/ns3\_star\_topology --nSpokes=20")change this according to slip if necessary**

**Enable Logging to see output:**

**NS\_LOG=StarTopologySimulation ./waf --run "scratch/ns3\_star\_topology --nSpokes=20"**

**mesh topology.**

**ns3 - nsallinone-3.35 - ns3.35 – src – mesh – examples - mesh.cc**

**Copy the mesh.cc file to scratch folder**

**Open terminal in ns3.35**

**./waf - - run mesh**

**./waf - - run mesh - - vis**

**Hybrid Topology / Netanim**

**Home terminal**

**sudo apt update**

**sudo apt upgrade**

**sudo apt install qt5-default mercurial**

**ns3 - nsallinone-3.35 - ns-3.35 - examples – tutorial**

**copy paste the third.cc file to scratch**

**Rename the file name to hybridanim**

**ns3.35 open in terminal**

* **./waf - - run hybridanime**

**Now in scratch folder double click - open in terminal**

* **gedit first.cc**

****

**AnimationInterface anim("hybridanim.xml");**

**anim.SetConstantPosition(p2pNodes.Get(0), 10.0, 10.0);**

**anim.SetConstantPosition(p2pNodes.Get(0), 20.0, 20.0);**

**Now in ns-3.35 open terminal**

* **./waf - - run scratch/hybridamin**

**ns3 - nsallinone-3.35 – netanim3.108 (open in terminal)**

* **./Netanim**

**Search your xml file and double click**

**For bus topology copy paste second.cc in scratch and rename it hybridanim**

**For star topology copy paste star.cc in scratch and rename it hybridanim**

**simulate UDP Client server**

**ns3 - nsallinone-3.35 – ns3.35**

**copy paste the udp-client-server file to scratch.**

**ns 3.35(open in terminal)**

**./waf - - run udp-client-server**

**./waf - - run udp-client-server - - vis**

**DHCP Server and clients**

**Same like udp..  
search for dhcp server client …copy and paste it in scratch  
compile using the commands and vizualize**

**simulate FTP using TCP**

**ns3 - nsallinone-3.35 –**

**ns3.35 - copy paste the tcp bulk send file to scratch.**

**ns 3.35(open in terminal)**

* **./waf - - run tcp-bulk-send**
* **./waf - - run tcp-bulk-send - - vis**

**Wireshark**

**Open wireshark in windows  
select ethernet**

**As per the questions ..apply the filter and capture the packets**

**Performance metrices**

**ns 3.35 – examples - tutorial - copy first.cc file.**

**Paste this file in scratch folder and rename it to Performance Metrices.**

**Open terminal in scratch folder.**

**Make the changes in code using gedit command and then compile the code.**

**Add this code in the file after** Simulator::Run ();

monitor->CheckForLostPackets();

Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>(flowHelper.GetClassifier());

std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats();

for (auto iter = stats.begin(); iter != stats.end(); ++iter) {

Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow(iter->first); // Fixed declaration of t

double duration = iter->second.timeLastRxPacket.GetSeconds() - iter->second.timeFirstTxPacket.GetSeconds();

double throughput = (iter->second.rxBytes \* 8.0) / duration / 1000000.0; // Mbps

double avgDelay = (iter->second.rxPackets > 0) ? (iter->second.delaySum.GetSeconds() / iter->second.rxPackets) : 0;

std::cout << "Flow " << iter->first << " (" << t.sourceAddress << " -> " << t.destinationAddress << ")\n";

std::cout << " Tx Packets: " << iter->second.txPackets << "\n";

std::cout << " Rx Packets: " << iter->second.rxPackets << "\n";

std::cout << " Packet Loss: " << (iter->second.txPackets - iter->second.rxPackets) << "\n";

std::cout << " Delay Sum: " << iter->second.delaySum.GetSeconds() << " s\n";

std::cout << " Response Time: " << avgDelay << " s\n";

std::cout << " Dropped Packets: " << iter->second.lostPackets << "\n";

std::cout << " Throughput: " << throughput << " Mbps\n";

}

**Save the file and then**

**Now in ns-3.35 open terminal**

**./waf - - run scratch/Performance Metrices.cc**

**UDP Client Server communication**

**In terminal**

* **sudo apt update**
* **sudo apt install build-essential**

**In home create new folder open folder (Practical 8) and open in terminal**

* **nano udp\_server.c**

**Paste code of server**

**2. UDP Server Program**

**Save the following code in a file named udp\_server.c:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <arpa/inet.h>**

**#define PORT 8080**

**#define BUFFER\_SIZE 1024**

**int main() {**

**int sockfd;**

**struct sockaddr\_in server\_addr, client\_addr;**

**char buffer[BUFFER\_SIZE];**

**socklen\_t addr\_len = sizeof(client\_addr);**

**if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {**

**perror("Socket creation failed");**

**exit(EXIT\_FAILURE);**

**}**

**server\_addr.sin\_family = AF\_INET;**

**server\_addr.sin\_addr.s\_addr = INADDR\_ANY;**

**server\_addr.sin\_port = htons(PORT);**

**if (bind(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {**

**perror("Bind failed");**

**close(sockfd);**

**exit(EXIT\_FAILURE);**

**}**

**Wait server response on screen**

**open folder (Practical 8) and open in terminal**

* **nano udp\_server.c**

**Paste code of server**

**3. UDP Client Program**

**Save the following code in a file named udp\_client.c:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <arpa/inet.h>**

**#define SERVER\_IP "127.0.0.1"**

**#define PORT 8080**

**#define BUFFER\_SIZE 1024**

**int main() {**

**int sockfd;**

**struct sockaddr\_in server\_addr;**

**char buffer[BUFFER\_SIZE];**

**socklen\_t addr\_len = sizeof(server\_addr);**

**if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {**

**perror("Socket creation failed");**

**exit(EXIT\_FAILURE);**

**}**

**server\_addr.sin\_family = AF\_INET;**

**server\_addr.sin\_port = htons(PORT);**

**server\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_IP);**

**while (1) {**

**printf("Enter message: ");**

**fgets(buffer, BUFFER\_SIZE, stdin);**

**buffer[strcspn(buffer, "\n")] = 0;**

**sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr\*)&server\_addr, addr\_len);**

**int n = recvfrom(sockfd, buffer, BUFFER\_SIZE, 0, (struct sockaddr\*)&server\_addr, &addr\_len);**

**buffer[n] = '\0';**

**printf("Server: %s\n", buffer);**

**}**

**close(sockfd);**

**return 0;**

**}**

**Again open in terminal**

**gcc udp\_server.c -o udp\_server**

**gcc udp\_client.c -o udp\_client**

**Again open terminal**

**TCP v/s UDP using wired network**

**ns -3 – nsallinone – ns 3.39 (open in terminal)**

**./ns3 run scratch/tcp/-udp-comparison**

**Copy Tcp-udp comparison xml folder in analysis folder (open terminal)**

**Python3 ns3\_analysis.py**