Termwork 6

Problem statement/experiment:

Simulate a Full duplex connection in a wired network using NS3.

Objectives of the experiment:

- Set up a simulated wired network environment in NS3 with nodes and links.
- Implement a full-duplex communication model for the network, allowing nodes to transmit and receive data simultaneously.
- Compare the results of the full-duplex simulation with traditional half-duplex communication to highlight the advantages of full-duplex in a wired network.

Theory:

- Full-duplex communication is a crucial aspect of modern wired networks, enabling simultaneous data transmission and reception on the same communication channel.
- NS3, or Network Simulator 3, is a widely used network simulation tool that allows researchers and network engineers to model and evaluate various network scenarios.
- Full-duplex communication allows two communicating devices to transmit and receive data simultaneously on the same communication link.
- This capability is essential for improving network efficiency, reducing latency, and enhancing overall network performance.
- NS3 is an open-source discrete-event network simulator used for modelling and analysing network protocols, devices, and configurations.
- It provides a platform for studying the behaviour of various network technologies in a controlled and reproducible environment.

Process Execution Steps:

Step 1 : Open UBUNTU and locate and open ns-allinone-3.28 folder on Desktop.

Step 2 : Go to ns-3.28 folder and open examples->tutorial->first.cc

Step 3 : In first.cc , include the following code. (Before – “Simulator::Run ();”)

```
#include “ns3/netanim-module.h”

AnimationInterface anim(“first, xml”);

AsciiTraceHelper ascii;

pointToPoint.EnableAsciiAll(ascii.CreateFileStream(“first.tr”));

pointToPoint.EnablePcapAll(“first”);
```

Step 4 : Copy first.cc and paste it in ns-3.28->scratch folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratch-simulator.cc and other files can be deleted.

Step 5 : Open terminal and change working directory to Desktop by cd Desktop and type following commands to go to location where scratch executable file is located i.e. scratch folder.

Step 6 : cd ns-allinone-3.28

Step 7 : cd ns-3.28

Step 8 : Run the first.cc by entering following command.

./waf –run scratch/first

Step 9 : Once build is successful, return to ns-allinone-3.28 folder with cd ../ and enter into netanim-3.108 with cd netanim-3.108

Step 10 : Now to see the animation, we have to open NetAnim software. So open by entering ./NetAnim on terminal.

Step 11 : In NetAnim, open first.xml by clicking on open XML trace file icon.

Step 12 : Click on run option/icon to see the animation. To see the packet transfer, open Packets Tab.

Conclusion:

In this experiment, we successfully simulated a full-duplex connection in a wired network using NS3. The results demonstrate the feasibility and benefits of implementing full-duplex communication for improved network performance.

References:

- Barry Nance: “Network Programming in C”, PHI 2002
- Bob Quinn, Dave Shute: “Windows Socket Network Programming”, Pearson 2003.

Termwork 7

Problem statement/definition:

Simulate a simple Wireless UDP application using NS3.

Objectives of the experiment:

- Simulate Wireless Network Communication
- Implement a UDP Application

Theory:

- Wireless communication has become a fundamental part of our daily lives, with applications ranging from mobile devices to IoT devices and beyond. To develop and optimize wireless applications, simulation tools like NS3 (Network Simulator 3) play a crucial role.
- NS3 is an open-source, highly flexible network simulation framework that allows researchers and engineers to model and analyze complex wireless network scenarios.
- In the context of wireless communication, User Datagram Protocol (UDP) is widely used due to its low overhead and simplicity.
- It is particularly suitable for real-time applications like video streaming, online gaming, and voice over IP (VoIP).
- Simulating a simple wireless UDP application using NS3 enables researchers and engineers to evaluate network performance, test different configurations, and assess the impact of various parameters on the application's quality of service.

Process Execution Steps:

Step 1 : Open UBUNTU and locate and open ns-allinone-3.28 folder on Desktop.

Step 2 : Go to ns-3.28 folder and open examples->tutorial->second.cc

Step 3 : In second.cc , include the following code. (Before – “Simulator::Run ();”)

```
#include “ns3/netanim-module.h”

AnimationInterface anim("second, xml");

AsciiTraceHelper ascii;

pointToPoint.EnableAsciiAll(ascii.CreateFileStream("second.tr"));

pointToPoint.EnablePcapAll("second");
```

Step 4 : Copy second.cc and paste it in ns-3.28->scratch folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratch-simulator.cc and other files can be deleted.

Step 5 : Open terminal and change working directory to Desktop by cd Desktop and type following commands to go to location where scratch executable file is located i.e. scratch folder.

Step 6 : cd ns-allinone-3.28

Step 7 : cd ns-3.28

Step 8 : Run the second.cc by entering following command.

./waf –run scratch/second

Step 9 : Once build is successful, return to ns-allinone-3.28 folder with cd ../ and enter into netanim-3.108 with cd netanim-3.108

Step 10 : Now to see the animation, we have to open NetAnim software. So open by entering ./NetAnim on terminal.

Step 11 : In NetAnim, open second.xml by clicking on open XML trace file icon.

Step 12 : Click on run option/icon to see the animation. To see the packet transfer, open Packets Tab.

Conclusion:

In this experiment, we successfully simulated a full-duplex connection in a wired network using NS3. The results demonstrate the feasibility and benefits of implementing full-duplex communication for improved network performance.

References:

- Barry Nance: “Network Programming in C”, PHI 2002
- Bob Quinn, Dave Shute: “Windows Socket Network Programming”, Pearson 2003.

Termwork 8

Problem statement/Title:

Simulate a simple 5G Network application using NS3.

Objectives of the Experiment:

- Evaluate 5G Network Performance
- Explore Network Slicing

Theory:

- The advent of 5G technology has revolutionized the telecommunications industry, promising faster data speeds, lower latency, and enhanced connectivity.
- To explore and understand the capabilities of 5G networks, network simulation tools like NS3 (Network Simulator 3) are invaluable.
- NS3 is an open-source, discrete-event simulator that allows researchers and engineers to model and analyze network protocols, applications, and topologies in a controlled environment.
- Simulating a simple 5G network application using NS3 offers a practical approach to comprehending the fundamental principles of 5G networks.
- It enables researchers and students to experiment with various aspects of 5G, such as network slicing, massive MIMO (Multiple Input Multiple Output), and edge computing.
- Through this simulation, one can investigate the impact of 5G on real-time applications, IoT (Internet of Things) devices, and autonomous vehicles.

Process Execution Steps:

Step 1 : Open UBUNTU and locate and open ns-allinone-3.28 folder on Desktop.

Step 2 : Go to ns-3.28 folder and open examples->tutorial->third.cc

Step 3 : In third.cc , include the following code. (Before – “Simulator::Run ();”)

```
#include “ns3/netanim-module.h”

AnimationInterface anim("third, xml");

AsciiTraceHelper ascii;

pointToPoint.EnableAsciiAll(ascii.CreateFileStream("third.tr"));

pointToPoint.EnablePcapAll("third");
```

Step 4 : Copy third.cc and paste it in ns-3.28->scratch folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratch-simulator.cc and other files can be deleted.

Step 5 : Open terminal and change working directory to Desktop by cd Desktop and type following commands to go to location where scratch executable file is located i.e. scratch folder.

Step 6 : cd ns-allinone-3.28

Step 7 : cd ns-3.28

Step 8 : Run the third.cc by entering following command.

./waf –run scratch/third

Step 9 : Once build is successful, return to ns-allinone-3.28 folder with cd ../ and enter into netanim-3.108 with cd netanim-3.108

Step 10 : Now to see the animation, we have to open NetAnim software. So open by entering ./NetAnim on terminal.

Step 11 : In NetAnim, open third.xml by clicking on open XML trace file icon.

Step 12 : Click on run option/icon to see the animation. To see the packet transfer, open Packets Tab.

Conclusion:

In this experiment, we successfully simulated a basic 5G network application using NS3. The results demonstrate the potential of NS3 for modeling and testing 5G network technologies.

References:

- Barry Nance: “Network Programming in C”, PHI 2002
- Bob Quinn, Dave Shute: “Windows Socket Network Programming”, Pearson 2003.

Termwork 9:

Problem statement/Title:

Understanding the working of Ipv6 in Low power lossy network

Objectives of the Experiment:

- Evaluate IPv6 Performance in LPLNs
- Optimize IPv6 Configuration for LPLNs

Theory:

- In the realm of modern networking, the Internet of Things (IoT) has gained significant prominence, leading to the emergence of Low Power Lossy Networks (LPLNs).
- These networks are characterized by their ability to operate efficiently with low-power devices and in environments where packet loss is a common occurrence.
- One of the fundamental technologies underlying the operation of LPLNs is the Internet Protocol version 6 (IPv6).
- IPv6 is an essential protocol that offers a larger address space, improved security, and enhanced support for IoT devices.
- However, applying IPv6 in LPLNs presents unique challenges due to the resource constraints and intermittent connectivity inherent in these networks.

Process Execution steps:

Go to the Location contiki-ng/tools/cooja/ with commands

1. Goto root directory
2. cd contiki-ng
3. cd tools
4. cd cooja

Run the cooja simulator with

5. ant run

Steps to create motes and configure them as server and client

1. Goto File -> New Simulation
2. Name the simulation and click on create
3. Click on Motes -> Add motes -> Create a new mote type -> Sky mote
4. Click on Browse and select ipv6-hooks.c (/contiki-ng/examples/libs/ipv6-hooks)
5. Click on open and then on compile and then on create
6. Enter the number of motes as 4 and click on Add motes
7. Place all motes close to each other such that the coverage is 100% for each of them
8. Right click on mote 1 and then click More tools for Sky 1 and then on Serial Socket (SERVER). Mote 1 has been configured as Server.

9. Similarly, configure motes 2, 3 and 4 as clients.
10. Copy the server's listening port number and paste it as the port number for all clients.
11. Start the server and connect the client to the server.
12. Run the simulation by clicking on Simulation -> Run Simulation

Conclusion:

In our experiment, we examined IPv6 in low-power lossy networks. We found that IPv6 efficiently supports connectivity in such networks, offering promising prospects for IoT and sensor applications.

References:

- Barry Nance: "Network Programming in C", PHI 2002
- Bob Quinn, Dave Shute: "Windows Socket Network Programming", Pearson 2003.

Termwork 10

Problem Statement:

Understanding the working of IoT routing using RPL protocol

Objectives of the Experiment:

- Investigate RPL's Energy-Efficient Routing
- Assess RPL's Scalability and Reliability

Theory:

The Internet of Things (IoT) has revolutionized the way devices and sensors communicate, enabling smart and interconnected systems.

A critical aspect of IoT is efficient data routing, and the Routing Protocol for Low-Power and Lossy Networks (RPL) plays a pivotal role in achieving this.

RPL is designed specifically for IoT applications, offering lightweight and energy-efficient routing solutions.

In theory, RPL operates on the principle of Directed Acyclic Graphs (DAGs), which optimizes routing in low-power and lossy networks. Devices, known as nodes, create a hierarchical structure to route data, thereby minimizing energy consumption and reducing latency.

This theory underlines RPL's suitability for IoT deployments where resources are constrained, and reliability is crucial.

Understanding the theoretical underpinnings of RPL is essential for developing and optimizing IoT networks.

Process Execution steps:

1. Goto File -> New Simulation
2. Name the simulation and click on create
3. Click on Motes -> Add motes -> Create a new mote type -> Sky mote
4. Click on Browse and select rpl-udp(/contiki-ng/examples/libs/rpl-udp)
5. Create udp-server.c and add 1 mote by clicking Motes -> Add new Mote -> Browse
6. Create udp-client.c and add 1 mote
7. Place both the motes close to each other
8. Configure 1 as server and 2 as client
9. Copy the server’s port number to the client.
10. Start the server and connect the client.
11. Run the simulation.

Conclusion:

In this experiment, we delved into the operation of IPv6 in low-power lossy networks. Our findings underscored its adaptability and efficiency, offering promising prospects for IoT and similar applications.

References:

- Barry Nance: “Network Programming in C”, PHI 2002
- Bob Quinn, Dave Shute: “Windows Socket Network Programming”, Pearson 2003.