#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 "ns3/udp-socket-factory.h"  
#include "ns3/log.h"

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE("PdcpCipheringApplication");

class PdcpCipheringApplication : public Application {  
private:  
    Ptr<Socket> m\_socket;   // Socket for communication  
    Address m\_peerAddress;  // Peer address  
    uint16\_t m\_peerPort;    // Peer port

protected:  
    virtual void StartApplication() override {  
        NS\_LOG\_INFO("Starting PDCP Ciphering Application");  
        m\_socket = Socket::CreateSocket(GetNode(), UdpSocketFactory::GetTypeId());  
        m\_socket->Bind();  
        m\_socket->Connect(InetSocketAddress(Ipv4Address::ConvertFrom(m\_peerAddress), m\_peerPort));  
        Simulator::Schedule(Seconds(1.0), &PdcpCipheringApplication::SendPacket, this);  
    }

    virtual void StopApplication() override {  
        NS\_LOG\_INFO("Stopping PDCP Ciphering Application");  
        if (m\_socket) {  
            m\_socket->Close();  
            m\_socket = 0;  
        }  
    }

    void SendPacket() {  
        NS\_LOG\_INFO("Sending packet");  
        Ptr<Packet> packet = Create<Packet>(1024);  // Create a packet of 1024 bytes  
        m\_socket->Send(packet);  
    }

public:  
    PdcpCipheringApplication() : m\_peerPort(0) {}

    void Setup(Address address, uint16\_t port) {  
        m\_peerAddress = address;  
        m\_peerPort = port;  
    }  
};

int main(int argc, char \*argv[]) {  
    CommandLine cmd;  
    cmd.Parse(argc, argv);

    NodeContainer nodes;  
    nodes.Create(2);

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

    NetDeviceContainer devices = p2p.Install(nodes);  
    InternetStackHelper stack;  
    stack.Install(nodes);

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

    uint16\_t port = 9;  
    Ptr<PdcpCipheringApplication> pdcpApp = CreateObject<PdcpCipheringApplication>();  
    pdcpApp->Setup(interfaces.GetAddress(1), port);  
    nodes.Get(0)->AddApplication(pdcpApp);  
    pdcpApp->SetStartTime(Seconds(1.0));  
    pdcpApp->SetStopTime(Seconds(10.0));

    Simulator::Run();  
    Simulator::Destroy();

    return 0;  
}

**Code Explanation:**

* **Includes and Namespace**: The code includes necessary NS-3 modules and uses the ns3 namespace to simplify code writing.
* **Logging Component**: The NS\_LOG\_COMPONENT\_DEFINE macro defines a logging component for this application, allowing for easy logging of events.
* **PdcpCipheringApplication Class**: This class inherits from the Application class and encapsulates the logic for sending and receiving PDCP packets.
  + **Private Members**: It includes a socket for communication, the peer address, and the peer port.
  + **StartApplication()**: This method is called when the application starts. It creates a socket, binds it, connects to the peer address, and schedules the first packet to be sent after 1 second.
  + **StopApplication()**: This method is called when the application stops, closing the socket.
  + **SendPacket()**: This method creates and sends a packet of 1024 bytes over the socket.
  + **Setup()**: This public method initializes the peer address and port.
* **Main Function**: The main function sets up the simulation environment:
  + Creates two nodes.
  + Sets up a point-to-point link between them.
  + Installs the Internet stack and assigns IP addresses.
  + Creates an instance of the PdcpCipheringApplication, sets it up with the peer address and port, and schedules it to start and stop at specific times.
  + Finally, it runs the simulator.

2. Specification Number

The code is inspired by the PDCP specifications outlined in the 3GPP documents. Specifically, it relates to:

* **3GPP TS 38.323**: This document defines the Packet Data Convergence Protocol for NR (New Radio) in 5G systems, including aspects like sequence numbering, ciphering, and header compression.
* **3GPP TS 25.323**: This document pertains to the LTE PDCP specifications and provides foundational knowledge that can be applied to NR PDCP.

3. Expected Outcome When Executing the Code

When you execute the code, the expected outcome includes:

* **Logging Output**: The application will log messages indicating when it starts sending packets.
* **Packet Transmission**: The application will create and send UDP packets of 1024 bytes from one node to another over a point-to-point link.
* **Simulation Run**: The simulation will run for a specified duration (from 1 to 10 seconds), during which packets are sent at regular intervals. The console will display logs related to packet transmission.

4. Detailed Code Explanation

Here’s a line-by-line explanation of the code:

#**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** "ns3/udp-socket-factory.h"

#**include** "ns3/log.h"

* **Includes**: These lines include the necessary NS-3 modules for core functionalities, networking, internet stack, point-to-point connections, applications, UDP socket factory, and logging.

**using** **namespace** ns3;

* **Namespace**: This allows the use of NS-3 classes and functions without needing to prefix them with ns3::.

NS\_LOG\_COMPONENT\_DEFINE("PdcpCipheringApplication");

* **Logging Component**: Defines a logging component for this application, which can be used for debugging and logging purposes.

**class** PdcpCipheringApplication : **public** Application {

* **Class Definition**: Defines the PdcpCipheringApplication class, which inherits from the Application class.

**private**:

Ptr<Socket> m\_socket; *// Socket for communication*

Address m\_peerAddress; *// Peer address*

**uint16\_t** m\_peerPort; *// Peer port*

* **Private Members**: Declares private member variables for the socket, peer address, and peer port.

**protected**:

**virtual** **void** StartApplication() **override** {

* **StartApplication()**: This method is called when the application starts. The override keyword indicates that this method overrides a base class method.

NS\_LOG\_INFO("Starting PDCP Ciphering Application");

* **Logging Start**: Logs a message indicating that the PDCP ciphering application is starting.

m\_socket = Socket::CreateSocket(GetNode(), UdpSocketFactory::GetTypeId());

* **Socket Creation**: Creates a UDP socket for communication.

m\_socket->Bind();

* **Binding**: Binds the socket to the local address.

m\_socket->Connect(InetSocketAddress(Ipv4Address::ConvertFrom(m\_peerAddress), m\_peerPort));

* **Connection**: Connects the socket to the peer address and port.

Simulator::Schedule(Seconds(1.0), &PdcpCipheringApplication::SendPacket, **this**);

* **Packet Scheduling**: Schedules the SendPacket method to be called after 1 second.

**virtual** **void** StopApplication() **override** {

* **StopApplication()**: This method is called when the application stops.

NS\_LOG\_INFO("Stopping PDCP Ciphering Application");

* **Logging Stop**: Logs a message indicating that the PDCP ciphering application is stopping.

**if** (m\_socket) {

m\_socket->Close();

m\_socket = 0;

}

* **Socket Closure**: Closes the socket if it is open and sets it to null.

**void** SendPacket() {

* **SendPacket()**: This method creates and sends a packet.

NS\_LOG\_INFO("Sending packet");

* **Logging Packet Send**: Logs a message indicating that a packet is being sent.

Ptr<Packet> packet = Create<Packet>(1024); *// Create a packet of 1024 bytes*

* **Packet Creation**: Creates a packet of 1024 bytes.

m\_socket->Send(packet);

* **Sending Packet**: Sends the created packet over the socket.

**public**:

PdcpCipheringApplication() : m\_peerPort(0) {}

* **Constructor**: Initializes m\_peerPort to zero.

**void** Setup(Address address, **uint16\_t** port) {

m\_peerAddress = address;

m\_peerPort = port;

}

* **Setup Method**: Sets up the peer address and port for the application.

**int** main(**int** argc, **char** \*argv[]) {

* **Main Function**: The entry point of the program.

CommandLine cmd;

cmd.Parse(argc, argv);

* **Command Line Parsing**: Parses command line arguments.

NodeContainer nodes;

nodes.Create(2);

* **Node Creation**: Creates a container for two nodes.

PointToPointHelper p2p;

p2p.SetDeviceAttribute("DataRate", StringValue("5Mbps"));

p2p.SetChannelAttribute("Delay", StringValue("2ms"));

* **Point-to-Point Setup**: Sets up a point-to-point link with a data rate of 5 Mbps and a delay of 2 ms.

NetDeviceContainer devices = p2p.Install(nodes);

* **Device Installation**: Installs the point-to-point devices on the nodes.

InternetStackHelper stack;

stack.Install(nodes);

* **Internet Stack Installation**: Installs the Internet stack on the nodes.

Ipv4AddressHelper address;

address.SetBase("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign(devices);

* **IP Address Assignment**: Assigns IP addresses to the devices.

**uint16\_t** port = 9;

Ptr<PdcpCipheringApplication> pdcpApp = CreateObject<PdcpCipheringApplication>();

pdcpApp->Setup(interfaces.GetAddress(1), port);

* **Application Setup**: Creates an instance of the PDCP application and sets it up with the peer address and port.

nodes.Get(0)->AddApplication(pdcpApp);

* **Adding Application**: Adds the PDCP application to the first node.

pdcpApp->SetStartTime(Seconds(1.0));

pdcpApp->SetStopTime(Seconds(10.0));

* **Start and Stop Times**: Sets the start and stop times for the application.

Simulator::Run();

Simulator::Destroy();

* **Run and Destroy**: Runs the NS-3 simulator and cleans up resources.

**return** 0;

}

* **Return Statement**: Ends the main function.

This detailed explanation provides an understanding of how the PDCP ciphering application is structured and operates within the NS-3 simulation environment.