#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;  
}

1. **Includes and Namespace**:
   * The code includes necessary NS-3 modules for core functionalities, networking, internet stack, point-to-point connections, and applications.
   * The using namespace ns3; statement allows us to use NS-3 classes and functions without prefixing them with ns3::.
2. **Logging Component**:
   * NS\_LOG\_COMPONENT\_DEFINE("PdcpCipheringApplication"); defines a logging component for this application, which can be used for debugging and logging purposes.
3. **PdcpCipheringApplication Class**:
   * This class inherits from Application and represents the PDCP ciphering application.
   * **Private Members**:
     + Ptr<Socket> m\_socket: A pointer to a socket for communication.
     + Address m\_peerAddress: The address of the peer node.
     + uint16\_t m\_peerPort: The port number of the peer node.
   * **Protected Methods**:
     + StartApplication(): 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(): Called when the application stops. It closes the socket.
     + SendPacket(): Creates and sends a packet of 1024 bytes.
   * **Public Methods**:
     + The constructor initializes m\_peerPort to zero.
     + Setup(Address address, uint16\_t port): Sets up the peer address and port.
4. **Main Function**:
   * The main function initializes the NS-3 simulation environment:
     + It creates two nodes.
     + Sets up a point-to-point link with a data rate of 5 Mbps and a delay of 2 ms.
     + Installs the Internet stack on both nodes.
     + Assigns IP addresses to the devices.
     + Creates an instance of PdcpCipheringApplication, sets it up with the peer address and port, and adds it to the first node.
     + The application is scheduled to start at 1 second and stop at 10 seconds.
   * Finally, it runs the NS-3 simulator and cleans up.

Expected Output

When you run the simulation, you should see output indicating that packets are being sent. The output will typically include logs from the NS-3 logging system, which can be enabled by setting the logging level. You can enable logging by adding the following line before running the simulation:

bash

NS\_LOG=PdcpCipheringApplication ./waf --run pdcp\_ciphering\_app

This will provide detailed logs about the application's behavior, including when packets are sent and received.

Additional Considerations

* **Ciphering Implementation**: The current code does not implement actual ciphering logic. You would need to integrate a ciphering algorithm to secure the data being sent. This could involve using libraries such as OpenSSL for encryption.
* **Error Handling**: The application currently does not handle errors or acknowledgments. In a real PDCP implementation, you would need to manage retransmissions and acknowledgments for reliable communication.
* **Packet Size**: The packet size is set to 1024 bytes. You can adjust this based on your requirements.