ERODE SENGUNTHAR
ENGINEERING COLLEGE
(An Autonomous Institution)
Approved by AICTE, New Delhi, Permanently Affiliated to Anna University- Chennai,
Accredited by National Board of Accreditation (NBA), New Delhi &
National Assessment and Accreditation Council (NAAC), Bangalore with 'A' Grade
PERUNDURAI -638 057, TAMILNADU, INDIA.

INSTITUTION'S INNOVATION COUNCIL
(Ministry of HRD Initiative)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COURSE TITLE

MODELLING MOBILE AD-HOC NETWORKS THROUGH NS3 SIMULATION ENVIRONMENT

PREPARED BY :

J. S. Sharanyanivasini,

Assistant Professor /ECE

1. Introduction to MANETs

1.1 Definition and Characteristics

A Mobile Ad-Hoc Network (MANET) is a self-configuring network of mobile devices that can communicate with each other without the need for a fixed infrastructure. These devices, often referred to as nodes, can dynamically form a network and establish communication links among themselves.

Key Characteristics of MANETs:

- **Dynamic Topology:** Nodes can move freely, leading to frequent changes in network topology.
- **Limited Resources:** Nodes have limited battery power, processing capacity, and memory.
- **Broadcast Nature:** Wireless communication is inherently broadcast, leading to potential security and interference issues.
- **Self-Organization:** Nodes must be able to autonomously discover and maintain routes.
- **Multi-hop Communication:** Data may need to be relayed through multiple nodes to reach its destination.

1.2 Applications of MANETs

- **Disaster Recovery:** MANETs can be used to establish communication networks in disaster-stricken areas where infrastructure has been damaged.
- **Military Communications:** MANETs can be deployed in military operations for secure and flexible communication among troops.
- **Emergency Response:** First responders can use MANETs to coordinate their efforts during emergencies.
- **Sensor Networks:** MANETs can be used to collect data from sensors deployed in various environments.
- **Vehicular Networks:** MANETs can enable communication between vehicles for applications such as traffic management and cooperative driving.

1.3 Challenges in MANETs

- **Dynamic Topology:** Maintaining routes and connectivity in a constantly changing network is a major challenge.
- **Limited Resources:** Efficient resource management is crucial to prolong network lifetime.

- **Security:** Broadcast nature of wireless communication makes MANETs vulnerable to security threats.
- **Scalability:** Designing protocols that can efficiently handle a large number of nodes is challenging.
- **Interference:** Wireless interference can degrade network performance.

2. NS3 for MANET Simulation

NS3 (Network Simulator 3) is a discrete-event network simulator that provides a powerful platform for modeling and simulating various network scenarios, including MANETs. It offers a flexible and modular architecture, allowing researchers and developers to customize and extend its capabilities.

Key Features of NS3 for MANET Simulation:

- **Wireless Channel Models:** NS3 provides a variety of wireless channel models to simulate realistic propagation conditions.
- **Mobility Models:** NS3 supports various mobility models to simulate the movement of nodes in different scenarios.
- **Routing Protocols:** NS3 includes implementations of several popular routing protocols for MANETs, such as AODV, DSR, and OLSR.
- **Application Support:** NS3 allows users to develop and simulate various applications for MANETs, such as file transfer, video streaming, and sensor data collection.
- **Visualization Tools:** NS3 can be integrated with visualization tools to provide insights into network behavior.

3. Basic MANET Simulation in NS3

To create a basic MANET simulation in NS3, you can follow these steps:

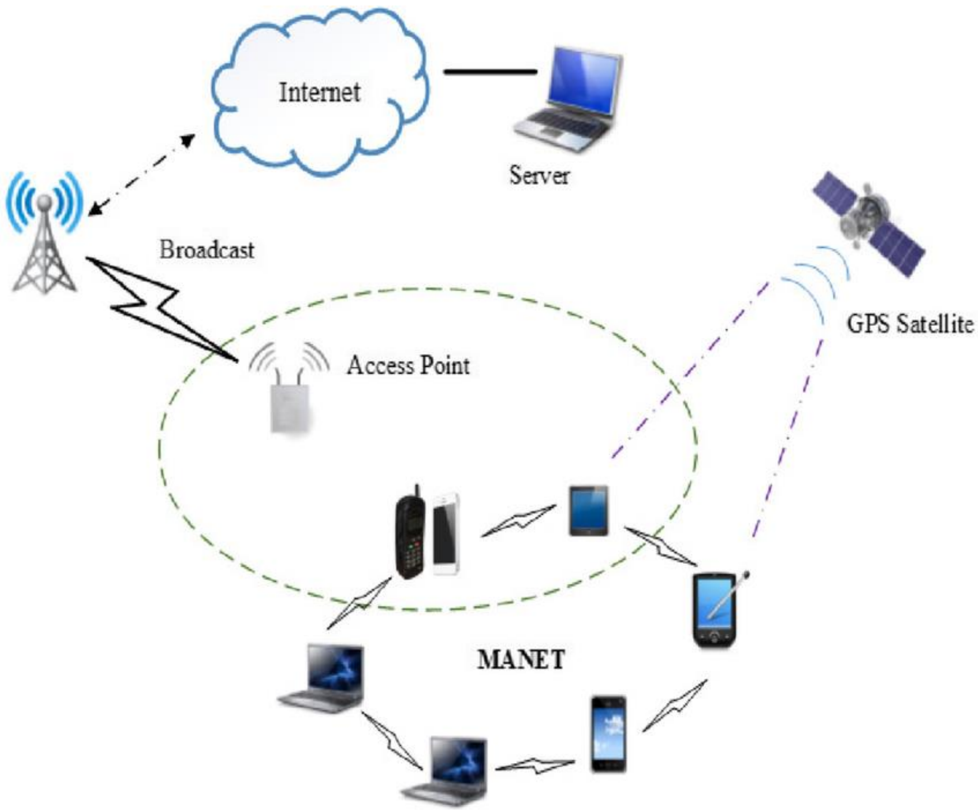
1. **Install NS3:** Download and install the NS3 software on your system.
2. **Create a Simulation Script:** Write a C++ script to define the simulation scenario, including the number of nodes, mobility models, routing protocol, and simulation time.
3. **Configure Nodes and Links:** Create nodes in the simulation and configure their wireless interfaces and mobility models.
4. **Install Protocols:** Install the selected routing protocol on the nodes.
5. **Run the Simulation:** Execute the simulation script to generate simulation results.
6. **Analyze Results:** Analyze the simulation results to evaluate network performance metrics, such as throughput, delay, and packet loss rate.

4. Advanced MANET Simulation in NS3

NS3 provides advanced features for more complex MANET simulations, such as:

- **Interference Modeling:** Simulate the impact of interference from other wireless devices.
- **Energy Consumption Modeling:** Model the energy consumption of nodes and their impact on network lifetime.
- **Security Modeling:** Simulate security attacks and evaluate the effectiveness of security mechanisms.
- **QoS Support:** Model Quality of Service (QoS) requirements for different applications.
- **Cross-Layer Optimization:** Optimize network performance by considering interactions between different layers of the protocol stack.

By leveraging the capabilities of NS3, researchers and developers can gain valuable insights into the behavior of MANETs and design more efficient and robust protocols and applications.



2. Overview of NS-3 Simulator

2.1 Introduction to NS-3 and its Features

NS-3 (Network Simulator 3) is a discrete-event network simulator designed for research and education in computer networking. It provides a flexible and modular platform for modeling and simulating various network scenarios, including wired, wireless, and optical networks.

Key Features of NS-3:

- **C++-based:** NS-3 is written in C++, offering high performance and flexibility for customization and extension.
- **Modular Architecture:** NS-3 employs a modular design, allowing users to easily add or replace components like protocols, channel models, and mobility models.
- **Wide Range of Models:** NS-3 provides a comprehensive library of models for various network components, including physical layer, data link layer, network layer, transport layer, and application layer.
- **Support for MANETs:** NS-3 offers extensive support for simulating Mobile Ad-Hoc Networks, including wireless channel models, mobility models, and routing protocols specifically designed for MANETs.
- **Visualization Tools:** NS-3 can be integrated with visualization tools to provide insights into network behavior.

2.2 Comparison with Other Simulators

NS-3 offers several advantages over other network simulators like NS-2 and OMNeT++:

Feature	NS-3	NS-2	OMNeT++
Programming Language	C++	C++/Tcl	C++
Architecture	Modular	Less modular	Modular
Performance	Generally faster	Can be slower for complex scenarios	Can be slower for complex scenarios
Documentation	Good	Can be less comprehensive	Good
Community Support	Growing	Large and established	Large and established

Export to Sheets

2.3 Setting Up NS-3 on Linux/Windows

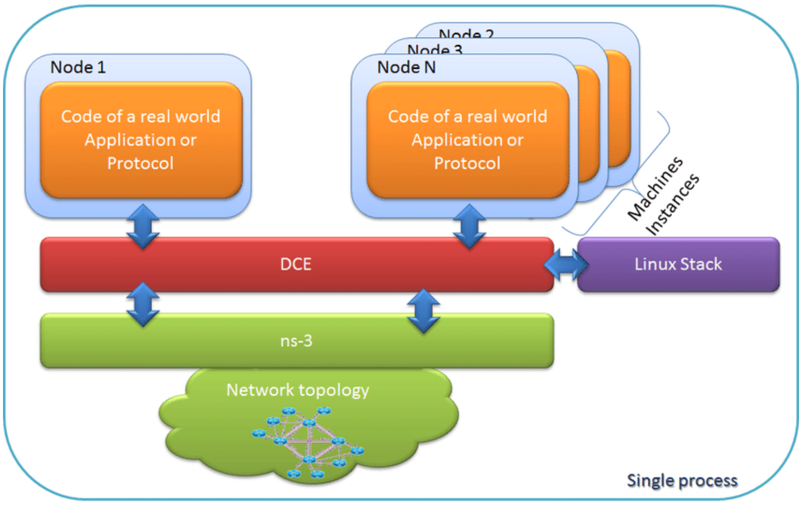
Linux:

1. **Install dependencies:** Install required packages like build-essential, cmake, python, and python-dev.
2. **Download and extract:** Download the NS-3 source code from the official website and extract it.
3. **Configure and build:** Create a build directory and run cmake to configure the build process. Then, use make to build the NS-3 libraries and examples.

Windows:

1. **Install Visual Studio:** Install a suitable version of Visual Studio with C++ support.
2. **Download and extract:** Download the NS-3 source code and extract it.
3. **Configure and build:** Open the NS-3 solution file in Visual Studio and build the solution.

2.4 NS-3 Architecture and Simulation Workflow



Simulation Workflow:

1. **Create a Simulation Script:** Write a C++ script to define the simulation scenario, including the number of nodes, mobility models, routing protocol, and simulation time.
2. **Configure Nodes and Links:** Create nodes in the simulation and configure their wireless interfaces and mobility models.
3. **Install Protocols:** Install the selected routing protocol on the nodes.
4. **Run the Simulation:** Execute the simulation script to generate simulation results.

5. **Analyze Results:** Analyze the simulation results to evaluate network performance metrics, such as throughput, delay, and packet loss rate.
6. **3. Basics of Network Simulation in NS-3**
7. **3.1 Creating and Managing Nodes**
8. In NS-3, nodes represent the individual devices in the network. They can be created using the NodeContainer class, which provides a convenient way to manage a collection of nodes.

```
NodeContainer nodes;

nodes.Create(10); // Create 10 nodes
```

3.2 Implementing Wireless Communication Models

Wireless communication in NS-3 is modeled using a layered approach:

- **Physical Layer:** Handles signal propagation, interference, and noise. NS-3 provides various channel models, such as the **YansRayleigh fading model** and the **ConstantPosition model**.
- **Data Link Layer:** Responsible for error detection and correction, as well as medium access control (MAC). The **WiFi** module in NS-3 implements the 802.11 standard, including various MAC protocols like **DCF** and **RTS/CTS**.

```
YansWifiPhyHelper wifiPhy;

wifiPhy.SetPtxInDbm(20); // Set transmit power


YansWifiChannelHelper wifiChannel;

wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelay");


WifiMacHelper wifiMac;

wifiMac.SetType("ns3::AdhocWifiMac");


NetDeviceContainer devices;

devices = wifiPhy.Install(wifiMac, nodes);
```

3.3 Mobility Models in NS-3

Mobility models simulate the movement of nodes in the network. NS-3 provides several built-in mobility models:

- **RandomWaypoint:** Nodes move randomly between randomly chosen waypoints.
- **Gauss-Markov:** Nodes move with a velocity that follows a Gauss-Markov process.
- **ConstantVelocity:** Nodes move with a constant velocity in a specific direction.

```
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::UniformDiscPositionAllocator",
                        "X", 0.0, "Y", 0.0, "R", 50.0); // Position allocator
mobility.SetMobilityModel("ns3::RandomWaypointMobilityModel",
                      "Speed", 20.0, "PauseTime", 0); // Mobility model
mobility.Install(nodes);
```

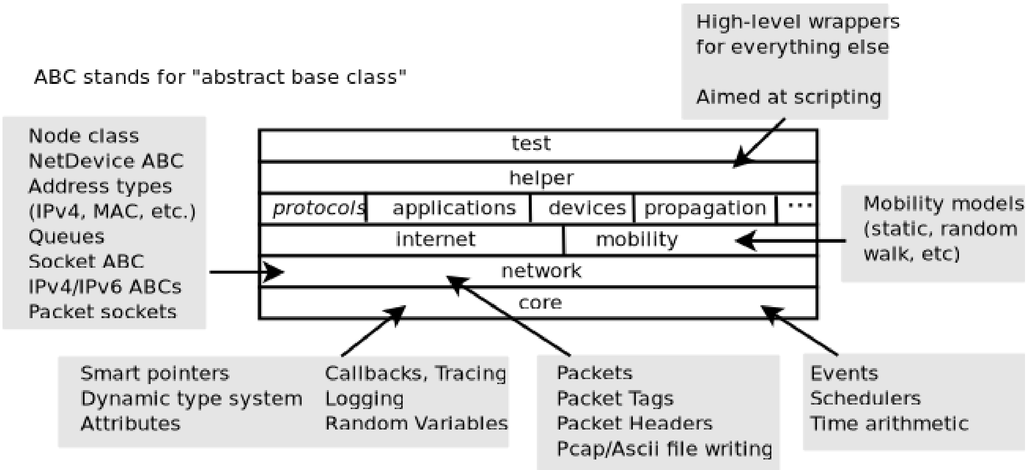
3.4 Ad-hoc Routing Protocols

Ad-hoc routing protocols enable nodes to dynamically discover routes to other nodes in the network. NS-3 supports several popular ad-hoc routing protocols:

- **AODV (Ad hoc On-Demand Distance Vector):** A reactive routing protocol that discovers routes on-demand.
- **DSR (Dynamic Source Routing):** A source routing protocol that maintains route information at the source node.
- **OLSR (Optimized Link State Routing):** A proactive routing protocol that periodically broadcasts link state information.

```
AodvHelper aodv;
InternetStackHelper internet;
internet.Install(nodes);
aodv.Install(nodes);
```

Diagram: Basic MANET Simulation Setup in NS-3



4. Hands-on Simulation of MANET using NS-3

4.1 Setting up a MANET Scenario in NS-3

1. **Create a Simulation Script:** Start by creating a C++ script file (e.g., manet_simulation.cc) to define your simulation.

2. **Include Necessary Headers:** Include the necessary headers for NS-3 components:

#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/mobility-module.h"

#include "ns3/wifi-module.h"

#include "ns3/aodv-module.h"

4.2 Configuring Wireless Networks and Mobility Models

- **Create Nodes:** Create a NodeContainer to hold your nodes:

C++

NodeContainer nodes;

nodes.Create(10); // Create 10 nodes

- **Configure Wireless Interfaces:**

- Create a WifiPhyHelper and a WifiChannelHelper to define the physical layer and channel characteristics:

-
  - C++
  - YansWifiPhyHelper wifiPhy;
  - wifiPhy.SetPtxInDbm(20);
  - YansWifiChannelHelper wifiChannel;
  - wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelay");
  -
  - WifiMacHelper wifiMac;
  - wifiMac.SetType("ns3::AdhocWifiMac");
  -
  - NetDeviceContainer devices;
  - devices = wifiPhy.Install(wifiMac, nodes);
  - Install the wireless interfaces on the nodes.
- **Configure Mobility:**
  - Create a MobilityHelper and choose a mobility model (e.g., RandomWaypoint):

C++

```
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::UniformDiscPositionAllocator",
                        "X", 0.0, "Y", 0.0, "R", 50.0);
mobility.SetMobilityModel("ns3::RandomWaypointMobilityModel",
                      "Speed", 20.0, "PauseTime", 0);
mobility.Install(nodes);
```

4.3 Implementing Routing Protocols for MANETs

- **Install Internet Stack:** Install the internet stack on each node:

C++

```
InternetStackHelper internet;
```

internet.Install(nodes);

- **Install Routing Protocol:** Install the chosen routing protocol (e.g., AODV):

C++

```
AodvHelper aodv;
aodv.Install(nodes);
```

4.4 Capturing Simulation Output in Trace Files

- **Create a Packet Sink:** Create a PacketSink application to receive packets at a destination node:

C++

```
Ptr<Node> sinkNode = nodes.Get(9); // Select node 9 as the sink
PacketSinkHelper sink("ns3::UdpSocketFactory", InetSocketAddress(Ipv4Address::GetAny(), 9));
ApplicationContainer sinkApp = sink.Install(sinkNode);
sinkApp.Start(Seconds(0.0));
sinkApp.Stop(Seconds(10.0));
```

- **Create a Packet Source:** Create a UdpClient application to send packets from a source node:

C++

```
Ptr<Node> sourceNode = nodes.Get(0); // Select node 0 as the source
UdpClientHelper client(InetSocketAddress(Ipv4Address::GetAny(), 9));
client.SetFillTraffic(true);
ApplicationContainer apps = client.Install(sourceNode);
apps.Start(Seconds(1.0));
apps.Stop(Seconds(10.0));
```

- **Enable Tracing:** Enable tracing for specific events (e.g., packet receptions) to capture simulation output:

C++

```
AsciiTraceHelper asciiTraceHelper;
```

Ptr<OutputStreamWrapper> stream = asciiTraceHelper.CreateFileStream("manet_simulation.tr");

sinkApp.TraceConnectWithoutContext("Rx", MakeCallback(&asciiTraceHelper, &AsciiTraceHelper::

5. Visualization with Matplotlib

5.1 Introduction to Matplotlib

Matplotlib is a powerful and versatile Python library for creating static, animated, and interactive visualizations in various formats, including line plots, bar charts, histograms, and more. It provides a flexible and customizable environment for data visualization and exploration.

5.2 Installing and Integrating Matplotlib with NS-3

- **Install Matplotlib:** Install the Matplotlib library using pip:

Bash

pip install matplotlib

- **Python Integration:** NS-3 can be integrated with Python for scripting and visualization. You can use the ns3::PyViz module to create visualizations directly within your NS-3 simulation script.

5.3 Parsing NS-3 Trace Files for Data Analysis

- **Generate Trace Files:** During the simulation, enable tracing for relevant events (e.g., packet receptions, transmissions, routing table updates) to generate trace files in various formats (e.g., ASCII, PCAP).
- **Parse Trace Files:** Use Python scripts to parse the trace files and extract the necessary data for analysis. You can use libraries like csv or pandas for efficient data handling.

5.4 Visualizing Network Metrics

- **Throughput:** Plot the throughput of the network over time.
- **Delay:** Plot the end-to-end delay experienced by packets.
- **Packet Delivery Ratio:** Plot the percentage of packets successfully delivered to the destination.
- **Node Mobility:** Visualize the movement of nodes in the network using scatter plots or animations.
- **Routing Table Updates:** Visualize the frequency and impact of routing table updates.

Example Python Script (Simplified):

Python

```
import matplotlib.pyplot as plt
import numpy as np

# Load data from trace file (replace with your data loading logic)
time = np.array([1, 2, 3, 4, 5])
throughput = np.array([10, 15, 20, 18, 22])

# Create the plot
plt.plot(time, throughput)
plt.xlabel("Time (seconds)")
plt.ylabel("Throughput (Mbps)")
plt.title("Network Throughput")
plt.grid(True)

# Show the plot
plt.show()
```

Diagram: Visualization Workflow

[Image of a diagram illustrating the visualization workflow:

1. NS-3 Simulation (generates trace files)
2. Python Script (parses trace files, performs data analysis)
3. Matplotlib (generates visualizations)]

By combining the power of NS-3 for simulation and Matplotlib for visualization, you can gain valuable insights into the performance and behavior of your MANET simulations. You can use these visualizations to identify bottlenecks, evaluate the effectiveness of different protocols, and fine-tune your network parameters.

6. Performance Metrics for MANETs

6.1 Throughput

- **Definition:** The amount of data successfully transmitted per unit time (e.g., bits per second, bytes per second).
- **Significance:** A crucial metric for evaluating the efficiency of data transfer in the network. Higher throughput indicates better performance.

6.2 Packet Delivery Ratio (PDR)

- **Definition:** The ratio of the number of packets successfully received at the destination to the total number of packets transmitted.
- **Significance:** Measures the reliability of data transmission in the network. A higher PDR indicates lower packet loss and better network stability.

6.3 End-to-End Delay

- **Definition:** The time taken for a packet to travel from the source node to the destination node.
- **Significance:** A critical metric for real-time applications, such as voice and video communication. Lower delay ensures timely delivery of data.

6.4 Network Lifetime

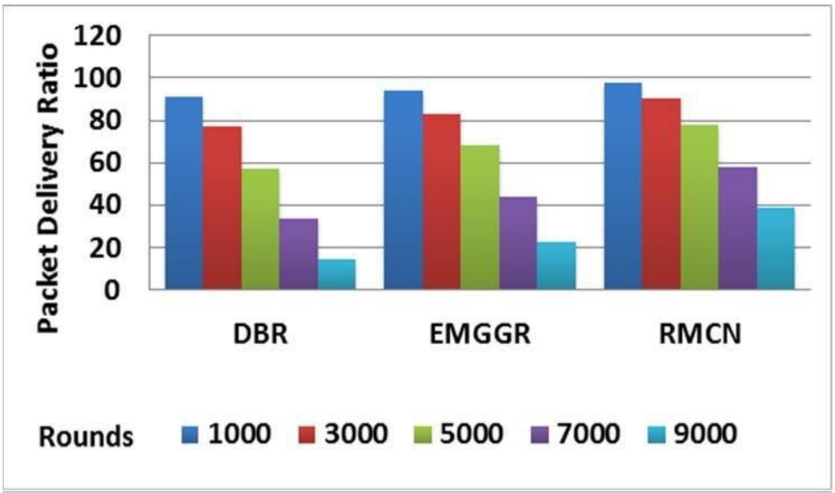
- **Definition:** The duration for which the network remains operational before the first node runs out of energy.
- **Significance:** A crucial metric for energy-constrained networks, such as sensor networks. A longer network lifetime indicates better energy efficiency.

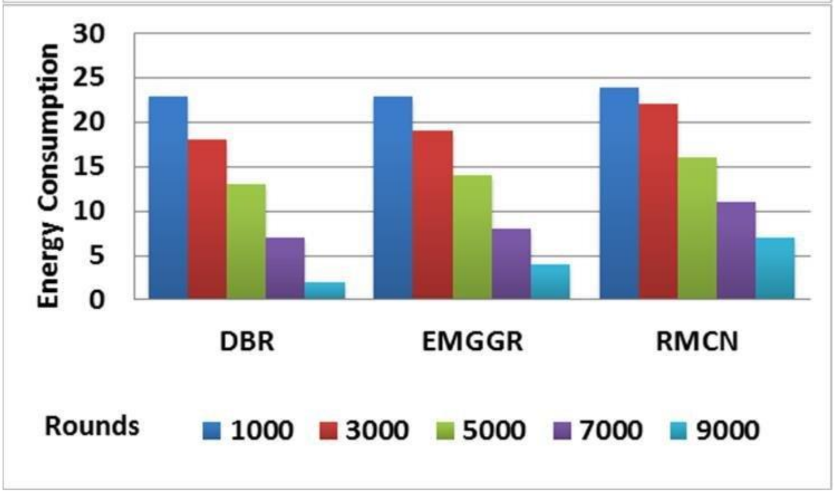
6.5 Scalability

- **Definition:** The ability of the network to maintain acceptable performance levels as the number of nodes increases.
- **Significance:** A scalable network can handle a growing number of devices without significant degradation in performance.

6.6 Energy Consumption

- **Definition:** The amount of energy consumed by nodes in the network for various operations, such as transmission, reception, and routing.
- **Significance:** Energy consumption is a critical factor for battery-powered devices in MANETs. Minimizing energy consumption can extend network lifetime.





Evaluating Performance Metrics in NS-3

NS-3 provides mechanisms to capture and analyze these performance metrics:

1. **Tracing:** Enable tracing for relevant events (e.g., packet receptions, transmissions) to generate trace files.
2. **Packet Sinks:** Use PacketSink applications to capture received packets and measure metrics like PDR and delay.
3. **Energy Models:** Utilize energy models in NS-3 to simulate energy consumption of nodes.
4. **Custom Metrics:** Develop custom scripts to extract and analyze specific performance metrics from simulation results.

7. Advanced Topics in MANET Simulation

7.1 Implementing Custom Routing Protocols in NS-3

NS-3 provides a flexible framework for developing and implementing custom routing protocols. Here's a general approach:

1. **Create a New Routing Protocol Class:** Inherit from the InternetProtocol class and implement the necessary routing protocol logic (e.g., route discovery, route maintenance, packet forwarding).
2. **Override Core Methods:** Override key methods like RouteOutput, RouteInput, and NotifyInterfaceUp to handle packet forwarding, route updates, and interface state changes.
3. **Integrate with NS-3:** Install your custom routing protocol on the nodes in your simulation.

7.2 Adding Energy Models for MANET Nodes

NS-3 allows you to incorporate energy consumption models into your simulations:

1. **Choose an Energy Model:** Select an appropriate energy model (e.g., basic battery model, more complex models considering different power states).
2. **Integrate with Nodes:** Associate the energy model with each node in the simulation.
3. **Track Energy Consumption:** Monitor energy consumption during the simulation and update node energy levels accordingly.
4. **Handle Node Failures:** Simulate node failures due to energy depletion.

7.3 Handling Scalability in Large MANET Simulations

Scalability is crucial for simulating large-scale MANETs. Here are some strategies:

1. **Efficient Data Structures:** Use efficient data structures (e.g., hash tables, balanced trees) to store and retrieve routing information.
2. **Distributed Algorithms:** Implement distributed algorithms for routing and other network functions to reduce communication overhead.
3. **Parallelization:** Utilize parallel and distributed computing techniques to speed up simulations.

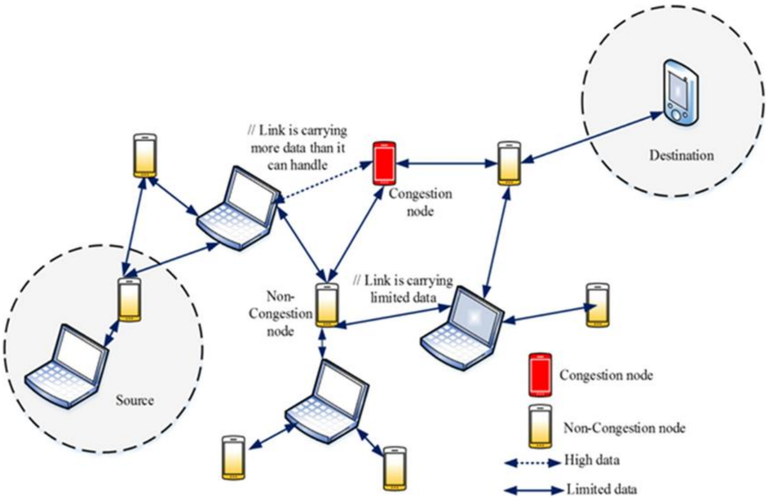
7.4 Introducing Security Models for MANETs

Security is a major concern in MANETs. You can incorporate security models into your simulations:

1. **Implement Security Mechanisms:** Simulate security mechanisms like encryption, authentication, and intrusion detection.

2. **Introduce Security Attacks:** Model various security attacks, such as eavesdropping, jamming, and denial-of-service attacks.
3. **Evaluate Security Protocols:** Evaluate the effectiveness of different security protocols in mitigating attacks.

Diagram: Advanced MANET Simulation Concepts



By exploring these advanced topics, you can create more realistic and insightful MANET simulations that address the challenges and complexities of real-world wireless networks.

8. Case Studies and Projects

8.1 Simulating Disaster Recovery Scenarios with MANETs

- **Scenario:** Model a disaster scenario (e.g., earthquake, flood) where communication infrastructure is disrupted.
- **Simulation:**
    - Create a dense deployment of mobile nodes (e.g., first responders, volunteers) in the affected area.
    - Simulate node mobility based on realistic movement patterns in a disaster situation.
    - Implement a robust routing protocol (e.g., AODV with link quality information) to maintain connectivity.
    - Evaluate the network's ability to establish and maintain communication links for critical tasks like:
        - **Emergency response coordination:** Data sharing between first responders.

- **Victim location and tracking:** Locating and assisting injured individuals.
- **Resource allocation:** Managing the distribution of supplies and medical assistance.

8.2 Performance Analysis of Routing Protocols (AODV vs. DSR)

- **Scenario:** Compare the performance of AODV and DSR routing protocols in a dynamic MANET environment.
- **Simulation:**
    - Implement both AODV and DSR on different sets of nodes in the same simulation.
    - Vary network parameters (e.g., node density, mobility patterns, traffic load) and observe the impact on:
        - **Throughput:** Data transmission rate.
        - **Delay:** Time taken for packet delivery.
        - **Packet Delivery Ratio:** Percentage of packets successfully received.
        - **Routing Overhead:** Control traffic generated by the routing protocols.
- **Analysis:** Analyze the simulation results to determine which protocol performs better under different conditions and identify their strengths and weaknesses.

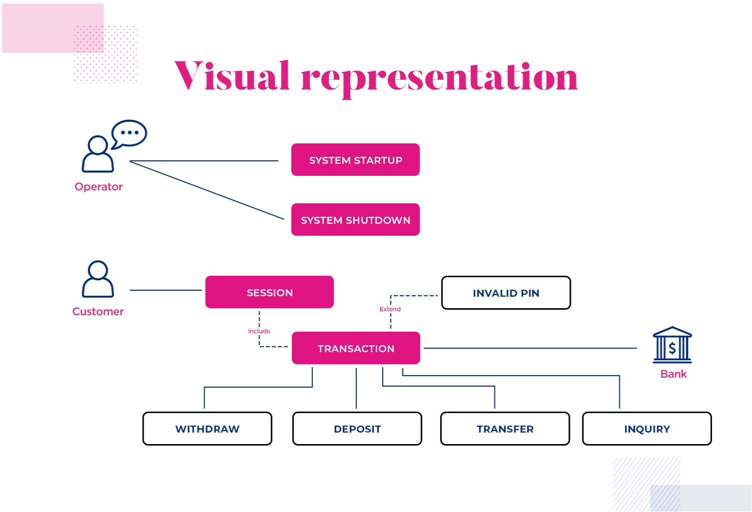
8.3 Energy-Efficient Routing in MANET

- **Scenario:** Design and evaluate energy-efficient routing protocols for MANETs.
- **Simulation:**
    - Implement energy models for nodes in the simulation.
    - Develop custom routing protocols that consider energy consumption during route discovery and maintenance (e.g., by selecting routes with lower energy costs).
    - Compare the energy efficiency of your custom protocols with existing protocols like AODV and DSR.
    - Evaluate the impact of energy-efficient routing on network lifetime and overall performance.

8.4 Real-time Visualization of Dynamic Topology Changes

- **Scenario:** Visualize the dynamic topology changes in a MANET in real-time.

- **Implementation:**
    - Use a visualization library (e.g., Matplotlib, VTK) to create a graphical representation of the network.
    - Update the visualization in real-time as nodes move and link states change.
    - Visualize key network parameters (e.g., node positions, link connections, routing paths) to provide insights into network behavior.



By undertaking these case studies and projects, you can gain practical experience in applying NS-3 for real-world MANET scenarios and develop a deeper understanding of the challenges and complexities involved in designing and optimizing these networks.

9. Conclusion and Future Scope

9.1 Current Trends in MANET Research

- **Software-Defined Networking (SDN) for MANETs:** Exploring the use of SDN principles to enhance control and programmability in MANETs.
- **Cognitive Radio for MANETs:** Utilizing cognitive radio techniques to dynamically adapt to the wireless environment and improve spectrum utilization.
- **Blockchain for MANETs:** Investigating the use of blockchain technology for secure and decentralized communication and resource management.
- **Internet of Things (IoT) and MANETs:** Integrating IoT devices into MANETs to enable a wide range of applications, such as smart cities and industrial automation.
- **Artificial Intelligence (AI) and Machine Learning for MANETs:** Applying AI and ML techniques for intelligent routing, resource allocation, and security in MANETs.

9.2 Challenges and Open Issues in MANET Simulation

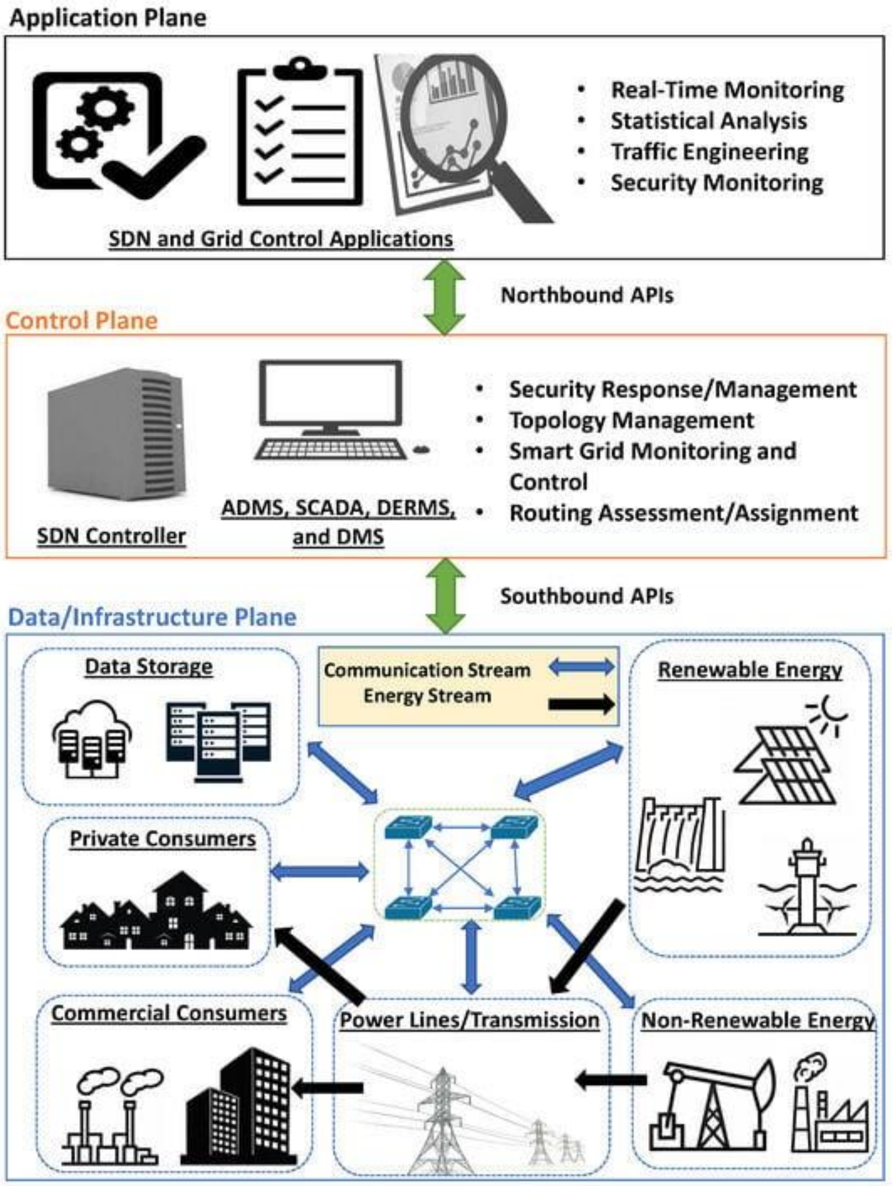
- **Realistic Modeling:** Developing more accurate and realistic models for wireless channels, mobility, and energy consumption.
- **Scalability and Performance:** Improving the scalability and performance of simulators to handle large-scale and complex MANET scenarios.
- **Integration with Real-World Environments:** Integrating simulations with real-world environments for more accurate and realistic evaluations.
- **Security and Privacy:** Developing and evaluating security mechanisms and privacy-preserving techniques in simulation environments.
- **Cross-Layer Optimization:** Exploring cross-layer optimization techniques to improve overall network performance.

9.3 Extending NS-3 for Other Wireless Networks

NS-3 can be extended to simulate other types of wireless networks:

- **Cellular Networks:** Modeling cellular networks, including LTE, 5G, and beyond.
- **Vehicular Networks (V2V, V2I):** Simulating communication between vehicles and between vehicles and infrastructure.
- **Wireless Sensor Networks (WSNs):** Modeling sensor networks with specific characteristics and applications.
- **Satellite Networks:** Simulating satellite communication systems and their integration with terrestrial networks.

Diagram: Future Directions of MANET Research



Conclusion

NS-3 provides a powerful platform for researching and developing innovative solutions for Mobile Ad-Hoc Networks. By addressing the challenges and exploring the emerging trends, researchers can continue to advance the field of MANETs and unlock their full potential for a wide range of applications in the future.

10. Deliverables

10.1 NS-3 Simulation Scripts

- **Core Simulation Script:**
    - A central C++ script that defines the simulation scenario (e.g., manet_simulation.cc).
    - Includes:
        - Node creation and configuration.
        - Wireless interface setup (phy, channel, MAC).
        - Mobility model installation.
        - Internet stack and routing protocol installation.
        - Application setup (packet sources/sinks).
        - Simulation time and tracing configuration.
- **Custom Routing Protocol Script (if applicable):**
    - A separate C++ script for implementing a custom routing protocol (e.g., my_routing_protocol.cc).
    - Inherits from the InternetProtocol class and implements necessary routing logic.

10.2 Matplotlib Scripts

- **Data Extraction Script:**
    - A Python script to parse NS-3 trace files (e.g., parse_trace.py).
    - Reads trace data, extracts relevant metrics (throughput, delay, PDR), and stores them in appropriate data structures (e.g., lists, arrays).
- **Visualization Scripts:**
    - Python scripts to generate various plots using Matplotlib (e.g., plot_throughput.py, plot_delay.py).
    - Creates plots for different metrics, including:
        - Throughput vs. time.
        - End-to-end delay vs. time.
        - Packet Delivery Ratio vs. time.
        - Node mobility trajectories.
        - Routing table updates.

10.3 Final Project Report

The final project report should include:

- **Introduction:**
    - Background on MANETs and their applications.
    - Motivation for the chosen research topic.
- **Methodology:**
    - Detailed description of the simulation setup:
        - Number of nodes, mobility model, wireless channel model.
        - Routing protocol(s) used.
        - Simulation parameters (time, traffic load).
        - Performance metrics evaluated.
- **Results and Analysis:**
    - Presentation of simulation results using tables, graphs, and visualizations generated by Matplotlib.
    - In-depth analysis of the results:
        - Comparison of different routing protocols.
        - Evaluation of the impact of mobility, traffic load, and other factors on performance.
        - Discussion of the limitations and assumptions of the simulation.
- **Conclusion:**
    - Summary of key findings and conclusions.
    - Discussion of future work and potential extensions.