Complete ZigBee ns-3 Project Guide

Project Overview

This comprehensive ZigBee smart home network simulation demonstrates all major features of the ZigBee implementation in ns-3, including:

Key Features

- ✅ **Network Formation**: Coordinator initialization and PAN establishment
- ✅ **Network Joining**: Association-based joining with staggered device connection
- ✅ **Routing Protocols**: Both Mesh routing and Many-to-One routing
- ✅ **APS Layer**: Unicast and Groupcast data transmission
- ✅ **Group Management**: Room-based lighting control with multiple endpoints
- ✅ **Multiple Device Types**: Coordinator, Routers, and sensor/actuator endpoints
- ✅ **Smart Home Use Cases**: Temperature monitoring and lighting control
- ✅ **Diagnostics**: Route tracing, routing tables, and statistics

Technology Stack

- **ns-3 Version**: 3.46 or later
- **ZigBee Module**: src/zigbee
- **LR-WPAN Module**: IEEE 802.15.4-2011 MAC layer
- **Language**: C++23

Installation & Setup

Prerequisites

```
# System requirements
Ubuntu 20.04+ (or similar Linux distribution)
GCC/Clang compiler with C++23 support
CMake 3.10+
Python 3.6+

# Check your system
g++ --version    # Should be 11+
cmake --version  # Should be 3.10+
python3 --version
```

Step 1: Navigate to ns-3 Directory

```
cd ~/ns3-workspace/ns-3-dev/
```

Step 2: Create the Main Simulation File

Create the file `src/zigbee/examples/smart-home-zigbee-complete.cc` with the complete source code provided in the artifact above.

Step 3: Update CMakeLists.txt

Edit `src/zigbee/examples/CMakeLists.txt` to include the new example:

```
set(base_examples
    zigbee-nwk-direct-join
    zigbee-nwk-association-join
    zigbee-nwk-routing
    zigbee-nwk-routing-grid
    zigbee-aps-data
    smart-home-zigbee-complete
)

foreach(
  example
  ${base_examples}
)
  build_lib_example(
    NAME ${example}
    SOURCE_FILES ${example}.cc
    LIBRARIES_TO_LINK ${libzigbee}
                      ${liblr-wpan}
  )
endforeach()
```

Step 4: Enable Examples in Configuration

```
# Configure ns-3 with examples enabled
./ns3 configure --enable-examples

# Verify configuration
./ns3 show profile
```

Step 5: Build the Project

```
# Build ns-3 with the ZigBee module
./ns3 build

# Check for any compilation errors
# If successful, you should see: "Build completed successfully"
```

Step 6: Verify Installation

```
# List available examples
./ns3 run --list | grep zigbee

# You should see:
# - zigbee-nwk-direct-join
# - zigbee-nwk-association-join
# - zigbee-nwk-routing
# - zigbee-nwk-routing-grid
# - zigbee-aps-data
# - smart-home-zigbee-complete
```

Project Files

Directory Structure

```
ns-3-dev/
├── src/
│   └── zigbee/
│       ├── model/
│       │   ├── zigbee-nwk.h/cc          # Network layer implementation
│       │   ├── zigbee-aps.h/cc          # Application support layer
│       │   └── zigbee-stack.h/cc        # Stack management
│       ├── helper/
│       │   ├── zigbee-helper.h/cc       # Installation helper
│       │   └── zigbee-stack-container.h # Stack container
│       └── examples/
│           ├── CMakeLists.txt           # Build configuration
│           ├── smart-home-zigbee-complete.cc  # Main project (YOUR FILE)
│           ├── zigbee-nwk-direct-join.cc
│           ├── zigbee-nwk-association-join.cc
│           ├── zigbee-nwk-routing.cc
│           ├── zigbee-nwk-routing-grid.cc
│           └── zigbee-aps-data.cc
└── build/
    └── lib/
        └── libns3-dev-zigbee-default.so
```

Key Files Description

File	Description
`smart-home-zigbee-complete.cc`	Main simulation - Complete smart home network
`zigbee-nwk-direct-join.cc`	Direct join (orphaning) procedure example
`zigbee-nwk-association-join.cc`	Association-based joining example
`zigbee-nwk-routing.cc`	Mesh routing demonstration
`zigbee-nwk-routing-grid.cc`	Many-to-One routing in grid topology
`zigbee-aps-data.cc`	APS layer data transmission example

Network Architecture

Device Topology

```
Network Layout:
  Coordinator (ZC) --- Router1 (ZR1) --- Router2 (ZR2) --- Router3 (ZR3)
  [Node 0: 00:00]       [Node 1]           [Node 2]          [Node 3]
                           |                  |              [Temp Sensor]
                           |                  |
                      Router4 (ZR4)      Router5 (ZR5)
                      [Node 4]           [Node 5]
                   [Living Room]         [Bedroom]
                   [Light1, Light2]      [Light3]
```

Device Roles

Node	Role	Short Address	IEEE Address	Function
0	Coordinator	00:00	00:00:00:00:00:00:00:01	Network root, data collector

Node	Role	Short Address	IEEE Address	Function
1	Router 1	Assigned	00:00:00:00:00:00:00:02	Hub router
2	Router 2	Assigned	00:00:00:00:00:00:00:03	Intermediate router
3	Router 3	Assigned	00:00:00:00:00:00:00:04	Temperature sensor
4	Router 4	Assigned	00:00:00:00:00:00:00:05	Living room lights
5	Router 5	Assigned	00:00:00:00:00:00:00:06	Bedroom lights

Group Configuration

```
Group 0x0001: Living Room
  ├─ Node 4, Endpoint 1 (Light 1)
  └─ Node 4, Endpoint 2 (Light 2)

Group 0x0002: Bedroom
  └─ Node 5, Endpoint 1 (Light 3)

Group 0x0003: All Lights
  ├─ Node 4, Endpoint 1 (Light 1)
  ├─ Node 4, Endpoint 2 (Light 2)
  └─ Node 5, Endpoint 1 (Light 3)
```

Position Layout

```
Grid Configuration (80m x 60m spacing):
  (0,0)      (80,0)     (160,0)
    N0 ------ N1 ------- N2 ------- N3 (240,0)
               |          |
               |          |
           (80,60)   (160,60)
             N4         N5
```

Running the Simulation

Basic Execution

```
# Run with default parameters
./ns3 run smart-home-zigbee-complete
```

With Command Line Parameters

```
# Run with custom simulation time (500 seconds)
./ns3 run "smart-home-zigbee-complete --simTime=500"

# Enable verbose logging (debug mode)
./ns3 run "smart-home-zigbee-complete --verbose=1"

# Disable Many-to-One routing (use Mesh routing instead)
./ns3 run "smart-home-zigbee-complete --manyToOne=0"

# Combined parameters
./ns3 run "smart-home-zigbee-complete --simTime=600 --verbose=1 --manyToOne=1"
```

Command Line Parameters

Parameter	Type	Description	Default	Valid Values
`--simTime`	uint32	Simulation duration in seconds	300	1 - 10000
`--verbose`	bool	Enable detailed debug logging	0 (false)	0, 1
`--manyToOne`	bool	Enable Many-to-One routing	1 (true)	0, 1

Expected Runtime

- **Default (300s simulation)**: ~5-10 seconds real time
- **With verbose logging**: ~15-30 seconds real time
- **Longer simulations (600s+)**: Proportionally longer

Understanding the Output

Output Structure

The simulation produces several sections of output:

1. Simulation Header

```
========================================
SMART HOME ZIGBEE NETWORK SIMULATION
========================================
Devices: 6
Simulation Time: 300s
Many-to-One Routing: Enabled
========================================
```

2. Network Formation

```
[+1.983s] Node 0 [00:00]: Network formation SUCCESSFUL
```

- Coordinator forms network
- Selects channel (typically 11-14)
- Assigns PAN ID

3. Device Joining Events

```
[+3.307s] Node 1 [ff:ff]: Network discovery completed - Found 1 network(s)
[+3.807s] Node 1 [d9:26]: Joined network successfully
  Short Address: d9:26
  Extended PAN ID: 0x1
```

- Each device discovers network
- Performs association
- Receives short address
- Promoted to router

4. Group Configuration

```
[+15.000s] Node 4 [a3:53]: Adding endpoint 1 to group 'Living Room' [00:01]
[+15.100s] Node 4 [a3:53]: Adding endpoint 2 to group 'Living Room' [00:01]
```

- Endpoints added to groups
- Multiple endpoints per device supported

5. Trace Route

```
========================================
TRACE ROUTE at +20.000s
From: 00:00 To: eb:0a
========================================
  1. Node 0 [00:00] -> NextHop [d9:26] (Direct Neighbor)
  2. Node 1 [d9:26] -> NextHop [47:f0]
  3. Node 2 [47:f0] -> NextHop [eb:0a] (Direct Neighbor)
  Route Complete! Total hops: 3
========================================
```

- Shows hop-by-hop path
- Identifies neighbors
- Counts total hops

6. Routing Tables

```
========================================
ROUTING TABLES at +21.000s
========================================

--- Node 0 [00:00] ---
Destination  Next hop  Status   Many-to-one
00:00        00:00     ACTIVE   TRUE
```

- Shows all routing entries
- Status indicators
- Route types

7. Data Transmission

```
[+22.000s] Node 3 [eb:0a]: Sending temperature reading to Coordinator
[+22.004s] Node 0 [00:00]: RECEIVED UNICAST DATA (Size: 2 bytes, Endpoint: 1,
Cluster: 1026)

[+27.000s] Node 0 [00:00]: Sending GROUP command 'Turn ON Living Room' to group
[00:01]
[+27.038s] Node 4 [a3:53]: RECEIVED GROUPCAST DATA (Size: 1 bytes, Endpoint: 1,
Cluster: 6)
[+27.038s] Node 4 [a3:53]: RECEIVED GROUPCAST DATA (Size: 1 bytes, Endpoint: 2,
Cluster: 6)
```

- Unicast: Point-to-point communication
- Groupcast: One message to multiple endpoints

8. Network Statistics

```
========================================
NETWORK STATISTICS
========================================
  Join Attempts:        5
  Join Successes:       5
  Route Discoveries:    1
  Packets Transmitted:  9
  Packets Received:     14
  Group Commands:       9
  Packet Success Rate:  155.56%
========================================
```

Interpreting Results

Success Indicators ✅

- Join Success Rate = 100% (all devices joined)
- Packet Success Rate > 100% (groupcast to multiple endpoints)
- All routes marked as ACTIVE
- Low end-to-end delay (<50ms)

Potential Issues ⚠️

- Join failures (< 100% success)
- Routes marked as DISCOVERY_UNDERWAY or VALIDATION_UNDERWAY
- High packet loss
- Extremely high delays (>500ms)

Code Architecture

Main Components

1. Global Variables

```
ZigbeeStackContainer g_zigbeeStacks;  // All device stacks
NetworkStats g_stats;                  // Statistics tracking
```

2. Callback Functions

Network Layer (NWK) Callbacks:

```
void NwkNetworkFormationConfirm(...)  // Formation result
void NwkNetworkDiscoveryConfirm(...)  // Discovery result
void NwkJoinConfirm(...)              // Join result
void NwkRouteDiscoveryConfirm(...)    // Route discovery result
```

Application Support Layer (APS) Callbacks:

```
void ApsDataIndication(...)           // Data reception
```

3. Helper Functions

Communication Functions:

```
void SendTemperatureReading(...)      // Sensor → Coordinator
void SendGroupCommand(...)            // Group control
void AddToGroup(...)                  // Group membership
```

Diagnostic Functions:

```
void TraceRoute(...)                  // Hop-by-hop tracing
void PrintAllRoutingTables()          // Table display
void PrintStatistics()                // Network metrics
void PrintMessage(...)                // Formatted output
```

4. Simulation Flow

```
main() {
    // 1. Parse command line arguments
    // 2. Configure logging
    // 3. Create nodes
    // 4. Install LR-WPAN devices
    // 5. Configure propagation channel
    // 6. Set node positions
    // 7. Install ZigBee stack
    // 8. Register callbacks
    // 9. Schedule events:
    //    - Network formation (1s)
    //    - Device joining (3-11s, staggered)
    //    - Group configuration (15s)
    //    - Route discovery (17s)
    //    - Data transmission (22s+)
    //    - Diagnostics (20s, 21s, 299s)
    // 10. Run simulation
    // 11. Cleanup
}
```

Event Timeline

Time (s)	Event	Description
0.0	Initialization	Create nodes, devices, stacks
1.0	Network Formation	Coordinator starts network
3.0 - 11.0	Device Joining	Routers join sequentially (2s apart)
15.0	Group Setup	Configure group memberships
17.0	Route Discovery	Establish routes (Mesh or Many-to-One)
20.0	Trace Route	Print hop-by-hop path
21.0	Print Tables	Display routing tables
22.0+	Data Transmission	Temperature reports (every 20s)
27.0+	Group Commands	Light control commands
299.0	Statistics	Print final network statistics
300.0	Simulation End	Cleanup and exit

Extending the Project

1. Add New Device Types

```
// Create a motion sensor class
class MotionSensor {
public:
    void DetectMotion() {
        // Send event to coordinator
        Ptr<Packet> p = Create<Packet>(1);
        ApsdeDataRequestParams params;
        params.m_clusterId = 0x0406;  // Occupancy Sensing cluster
        // ... configure and send
    }
};

// Add to simulation
Simulator::Schedule(Seconds(30), &MotionSensor::DetectMotion, motionSensor);
```

2. Implement Door Lock Control

```
void SendLockCommand(Ptr<ZigbeeStack> src, Ptr<ZigbeeStack> dst, bool lock) {
    uint8_t command[1] = {lock ? 0x01 : 0x00};
    Ptr<Packet> p = Create<Packet>(command, 1);

    ApsdeDataRequestParams params;
    params.m_clusterId = 0x0101;  // Door Lock cluster
    params.m_profileId = 0x0104;   // Home Automation
    params.m_srcEndPoint = 1;
    params.m_dstEndPoint = 1;
    params.m_dstAddr16 = dst->GetNwk()->GetNetworkAddress();
    params.m_dstAddrMode = ApsDstAddressMode::DST_ADDR16_DST_ENDPOINT_PRESENT;

    src->GetAps()->ApsdeDataRequest(params, p);
}

// Schedule lock/unlock
Simulator::Schedule(Seconds(50), &SendLockCommand, coordinator, doorLock, true);
Simulator::Schedule(Seconds(100), &SendLockCommand, coordinator, doorLock, 
false);
```

3. Add Thermostat Control

```
struct ThermostatData {
    int16_t currentTemp;   // x100 (2350 = 23.50°C)
    int16_t setpoint;      // x100
    uint8_t mode;          // 0=Off, 1=Heat, 2=Cool
};

void SendThermostatSetpoint(Ptr<ZigbeeStack> src, Ptr<ZigbeeStack> thermostat,
                            int16_t setpoint) {
    uint8_t data[2];
    data[0] = setpoint & 0xFF;
    data[1] = (setpoint >> 8) & 0xFF;

    Ptr<Packet> p = Create<Packet>(data, 2);

    ApsdeDataRequestParams params;
    params.m_clusterId = 0x0201;  // Thermostat cluster
    params.m_profileId = 0x0104;
    // ... configure and send
}
```

4. Test Network Resilience (Node Failure)

```
void SimulateNodeFailure(Ptr<Node> node, Time failTime, Time recoverTime) {
    // Fail node
    Simulator::Schedule(failTime, [node]() {
        Ptr<LrWpanNetDevice> dev = node->GetDevice(0)-
>GetObject<LrWpanNetDevice>();
        dev->GetPhy()-
>SetTrxState(LrWpanPhyEnumeration::IEEE_802_15_4_PHY_TRX_OFF);
        std::cout << "Node " << node->GetId() << " FAILED" << std::endl;
    });
    // Recover node
    Simulator::Schedule(recoverTime, [node]() {
        Ptr<LrWpanNetDevice> dev = node->GetDevice(0)-
>GetObject<LrWpanNetDevice>();
        dev->GetPhy()-
>SetTrxState(LrWpanPhyEnumeration::IEEE_802_15_4_PHY_RX_ON);
        std::cout << "Node " << node->GetId() << " RECOVERED" << std::endl;
    });
}

// Test resilience
SimulateNodeFailure(nodes.Get(4), Seconds(50), Seconds(100));
```

5. Add Node Mobility

```
// Instead of ConstantPositionMobilityModel, use:
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
                         "Mode", StringValue("Time"),
                         "Time", StringValue("2s"),
                         "Speed",
StringValue("ns3::ConstantRandomVariable[Constant=1.0]"),
                         "Bounds", RectangleValue(Rectangle(-50, 250, -50,
150)));

// Apply to specific nodes
NodeContainer mobileNodes;
mobileNodes.Add(nodes.Get(3));  // Make temperature sensor mobile
mobileNodes.Add(nodes.Get(5));  // Make bedroom light mobile
mobility.Install(mobileNodes);

// Monitor position changes
void PrintPosition(Ptr<Node> node) {
    Ptr<MobilityModel> mobility = node->GetObject<MobilityModel>();
    Vector pos = mobility->GetPosition();
    std::cout << "Node " << node->GetId() << " at ("
              << pos.x << ", " << pos.y << ")" << std::endl;
}

// Schedule periodic position printing
for (uint32_t i = 0; i < 300; i += 10) {
    Simulator::Schedule(Seconds(i), &PrintPosition, nodes.Get(3));
}
```

6. Implement Custom Data Structures

```
// Complex sensor data
struct SensorReading {
    uint16_t temperature;  // x10
    uint8_t humidity;      // percentage
    uint16_t pressure;     // hPa
    uint32_t timestamp;    // seconds
    uint16_t battery;      // mV
};

void SendSensorReading(Ptr<ZigbeeStack> sensor, Ptr<ZigbeeStack> coordinator) {
    SensorReading reading;
    reading.temperature = 235;  // 23.5°C
    reading.humidity = 65;
    reading.pressure = 1013;
    reading.timestamp = Simulator::Now().GetSeconds();
    reading.battery = 3000;     // 3.0V
    uint8_t buffer[11];
    buffer[0] = reading.temperature & 0xFF;
    buffer[1] = (reading.temperature >> 8) & 0xFF;
    buffer[2] = reading.humidity;
    buffer[3] = reading.pressure & 0xFF;
    buffer[4] = (reading.pressure >> 8) & 0xFF;
    buffer[5] = reading.timestamp & 0xFF;
    buffer[6] = (reading.timestamp >> 8) & 0xFF;
    buffer[7] = (reading.timestamp >> 16) & 0xFF;
    buffer[8] = (reading.timestamp >> 24) & 0xFF;
    buffer[9] = reading.battery & 0xFF;
    buffer[10] = (reading.battery >> 8) & 0xFF;

    Ptr<Packet> p = Create<Packet>(buffer, sizeof(buffer));

    ApsdeDataRequestParams params;
    params.m_clusterId = 0x0402;  // Temperature cluster
    // ... send packet
}
```

7. Add Traffic Patterns

```
// Periodic reporting with jitter
void SchedulePeriodicReports(Ptr<ZigbeeStack> sensor, Ptr<ZigbeeStack> 
coordinator,
                            Time interval, Time jitter, uint32_t count) {
    Ptr<UniformRandomVariable> random = CreateObject<UniformRandomVariable>();
    random->SetAttribute("Min", DoubleValue(-jitter.GetSeconds()));
    random->SetAttribute("Max", DoubleValue(jitter.GetSeconds()));

    for (uint32_t i = 0; i < count; i++) {
        double offset = random->GetValue();
        Time schedTime = Seconds(interval.GetSeconds() * i + offset);
        Simulator::Schedule(schedTime, &SendTemperatureReading, sensor, 
coordinator);
    }
}

// Use it
SchedulePeriodicReports(router3, coordinator, Seconds(20), Seconds(2), 15);
```

8. Implement Event-Driven Reporting

```
class EventDrivenSensor {
private:
    Ptr<ZigbeeStack> m_stack;
    Ptr<ZigbeeStack> m_coordinator;
    double m_threshold;
    double m_lastValue;

public:
    void CheckThreshold() {
        Ptr<UniformRandomVariable> rand = CreateObject<UniformRandomVariable>();
        double currentValue = rand->GetValue() * 100;

        if (std::abs(currentValue - m_lastValue) > m_threshold) {
            std::cout << "Threshold exceeded! Sending alert..." << std::endl;
            SendAlert(currentValue);
            m_lastValue = currentValue;
        }
        // Check again in 5 seconds
        Simulator::Schedule(Seconds(5), &EventDrivenSensor::CheckThreshold,
this);
    }

    void SendAlert(double value) {
        // Send urgent notification
        SendTemperatureReading(m_stack, m_coordinator);
    }
};
```

9. Create Larger Networks

```
// Grid topology with 50 nodes
uint32_t gridWidth = 10;
uint32_t gridHeight = 5;
uint32_t totalNodes = gridWidth * gridHeight;

nodes.Create(totalNodes);

// Position nodes in grid
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                             "MinX", DoubleValue(0.0),
                             "MinY", DoubleValue(0.0),
                             "DeltaX", DoubleValue(30.0),
                             "DeltaY", DoubleValue(30.0),
                             "GridWidth", UintegerValue(gridWidth),
                             "LayoutType", StringValue("RowFirst"));
mobility.Install(nodes);

// Stagger joining
for (uint32_t i = 1; i < totalNodes; i++) {
    Ptr<ZigbeeStack> stack = zigbeeStacks.Get(i)->GetObject<ZigbeeStack>();
    double joinTime = 3.0 + (i * 2.0);

    NlmeNetworkDiscoveryRequestParams params;
    params.m_scanChannelList.channelPageCount = 1;
    params.m_scanChannelList.channelsField[0] = 0x7800;
    params.m_scanDuration = 2;

    Simulator::Schedule(Seconds(joinTime),
                       &ZigbeeNwk::NlmeNetworkDiscoveryRequest,
                       stack->GetNwk(), params);
}
```

10. Add Performance Metrics

```
class PerformanceMonitor {
private:
    std::map<uint32_t, Time> m_sentTimes;
    std::vector<double> m_delays;
    uint32_t m_packetsSent = 0;
    uint32_t m_packetsReceived = 0;

public:
    void PacketSent(uint32_t uid) {
        m_sentTimes[uid] = Simulator::Now();
        m_packetsSent++;
    }

    void PacketReceived(uint32_t uid) {
        if (m_sentTimes.find(uid) != m_sentTimes.end()) {
            Time delay = Simulator::Now() - m_sentTimes[uid];
            m_delays.push_back(delay.GetMilliSeconds());
            m_packetsReceived++;
        }
    }

    void PrintStatistics() {
        double avgDelay = 0;
        for (double d : m_delays) {
            avgDelay += d;
        }
        avgDelay /= m_delays.size();

        double pdr = (double)m_packetsReceived / m_packetsSent * 100.0;

        std::cout << "Performance Metrics:" << std::endl;
        std::cout << "  Packets Sent: " << m_packetsSent << std::endl;
        std::cout << "  Packets Received: " << m_packetsReceived << std::endl;
        std::cout << "  PDR: " << pdr << "%" << std::endl;
        std::cout << "  Avg Delay: " << avgDelay << " ms" << std::endl;
    }
};
```

Troubleshooting

Common Issues and Solutions

Issue 1: Program Not Found

```
Exception: Couldn't find the specified program: smart-home-zigbee-complete
```

Solutions:

1. Check file exists: `ls src/zigbee/examples/smart-home-zigbee-complete.cc`
2. Verify CMakeLists.txt updated correctly
3. Rebuild: `./ns3 clean && ./ns3 build`
4. Check for typos in filename

Issue 2: Compilation Errors

Error: Missing includes

```
error: 'MobilityHelper' was not declared
```

Solution: Add missing include:

```
#include "ns3/mobility-module.h"
```

Error: No member named 'GetAsString'

```
// Wrong:
std::string addr = macAddress.GetAsString();

// Correct:
std::ostringstream oss;
```

```
oss << macAddress;
std::string addr = oss.str();
```

Issue 3: Devices Fail to Join

Symptoms:

- Join success rate < 100%
- Devices stuck at "Network discovery"

Solutions:

1. Check scan duration: Increase from 2 to 4

```
netDiscParams.m_scanDuration = 4;
```

2. Check channel mask:

```
netDiscParams.m_scanChannelList.channelsField[0] = 0x7800;  // Ch 11-14
```

3. Check node positions (must be in range):

```
// Reduce distances if needed
"DeltaX", DoubleValue(50.0),  // Instead of 80
```

Issue 4: Route Discovery Fails

Symptoms:

- Routes not established
- "Destination Unreachable" in trace

Solutions:

1. Ensure devices have joined:

```
// Route discovery should happen AFTER all joins complete
double routingTime = lastJoinTime + 5.0;
```

2. Check routing is enabled:

```
// For routers:
NlmeStartRouterRequestParams startParams;
Simulator::ScheduleNow(&ZigbeeNwk::NlmeStartRouterRequest, ...);
```

3. Verify network topology (devices in range)

Issue 5: Group Commands Not Received

Symptoms:

- Groupcast packets not delivered
- No "RECEIVED GROUPCAST DATA" messages

Solutions:

1. Verify group membership:

```
// Check group was added
ApsmeGroupRequestParams params;
```

```
params.m_groupAddress = Mac16Address("00:01");
params.m_endPoint = 1;
stack->GetAps()->ApsmeAddGroupRequest(params);
```

2. Ensure routes exist to group members:

```
// Add route discovery before group commands
Simulator::Schedule(Seconds(17), &RouteDiscovery, ...);
Simulator::Schedule(Seconds(27), &SendGroupCommand, ...);  // 10s later
```

3. Check destination address mode:

```
params.m_dstAddrMode = ApsDstAddressMode::GROUP_ADDR_DST_ENDPOINT_NOT_PRESENT;
```

Issue 6: High Packet Loss

Symptoms:

- Packet success rate < 80%
- Many packets not received

Solutions:

1. Reduce traffic load:

```
// Increase reporting interval
Simulator::Schedule(Seconds(30), ...);  // Instead of 20
```

2. Adjust propagation model:

```
Ptr<LogDistancePropagationLossModel> propModel =
    CreateObject<LogDistancePropagationLossModel>();
propModel->SetPathLossExponent(2.0);  // Default is 3.0
```

3. Check buffer sizes:

```
Config::SetDefault("ns3::LrWpanNetDevice::QueueSize", UintegerValue(1000));
```

Issue 7: Simulation Crashes

Common causes:

1. Null pointer access
2. Invalid timing (negative times)
3. Memory issues

Debug steps:

```
# Enable all debug info
export NS_LOG="*=level_all|prefix_func|prefix_time"

# Run with gdb
gdb --args ./build/examples/zigbee/smart-home-zigbee-complete

# Inside gdb:
run
# When crash occurs:
backtrace
```

Issue 8: Verbose Logging Too Much Output

Solution: Enable selective logging:

```
// Instead of LogComponentEnableAll
LogComponentEnable("ZigbeeNwk", LOG_LEVEL_INFO);
LogComponentEnable("ZigbeeAps", LOG_LEVEL_WARN);
```

Additional Examples

Example 1: Direct Join Network

```
# Run the direct join example
./ns3 run zigbee-nwk-direct-join

# This demonstrates:
# - Orphaning procedure
# - Manual device registration
# - Faster joining (no discovery needed)
```

Example 2: Association Join

```
./ns3 run zigbee-nwk-association-join

# This demonstrates:
# - Standard association procedure
# - Network discovery process
# - Multi-hop network formation
```

Example 3: Mesh Routing

```
./ns3 run zigbee-nwk-routing

# This demonstrates:
# - On-demand route discovery (RREQ/RREP)
# - Route table management
# - Simple linear topology
```

Example 4: Large-Scale Many-to-One

```
./ns3 run zigbee-nwk-routing-grid

# This demonstrates:
# - 50-node grid network
# - Many-to-One routing efficiency
# - Scalability testing
```

Example 5: APS Layer Features

```
./ns3 run zigbee-aps-data

# This demonstrates:
# - Unicast vs Groupcast
# - Endpoint addressing
# - Group management
```

Performance Tuning

Optimize Join Time

```
// Reduce scan duration for faster joining (less thorough)
netDiscParams.m_scanDuration = 0;  // Minimum scan

// Reduce interval between joins
double joinInterval = 1.0;  // Instead of 2.0
```

Optimize Routing

```
// Adjust RREQ retry parameters
Config::SetDefault("ns3::ZigbeeNwk::NwkcRREQRetries", UintegerValue(2));  //
Default: 3
Config::SetDefault("ns3::ZigbeeNwk::NwkcRREQRetryInterval",
TimeValue(MilliSeconds(50)));
```

Reduce Memory Usage

```
// Limit routing table size
Config::SetDefault("ns3::ZigbeeNwk::MaxRoutingTableSize", UintegerValue(50));

// Limit pending packet queue
Config::SetDefault("ns3::ZigbeeNwk::MaxPendingTxQueueSize", UintegerValue(10));
```

Reference Documentation

ZigBee Specification

- **ZigBee Pro Specification 2017 (R22 1.0)**
- Available from: Connectivity Standards Alliance

ns-3 Documentation

- **ns-3 Manual**: https://www.nsnam.org/documentation/
- **ns-3 Tutorial**: https://www.nsnam.org/docs/tutorial/html/
- **ZigBee Module**: https://www.nsnam.org/docs/models/html/zigbee.html

IEEE Standards

- **IEEE 802.15.4-2011**: LR-WPAN MAC and PHY
- Available from: https://standards.ieee.org/

Books

1. Farahani, Shahin. "ZigBee Wireless Networks and Transceivers" (2008)
2. Gislason, Drew. "ZigBee Wireless Networking" (2008)

Conclusion

You now have a complete, working ZigBee smart home network simulation that demons