To set up a network simulation in NS-3 that incorporates both wired routing and WiFi, along with mobility for the WiFi nodes, you need to configure several components. Here, I'll provide a straightforward example to demonstrate how you can integrate these elements. This example assumes you're already familiar with the basic structure of an NS-3 program, including node creation, application setup, and running the simulation.

Step 1: Include Necessary Headers

First, include all necessary headers for WiFi, CSMA, Internet stack, and mobility:

```cpp
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/wifi-module.h"
#include "ns3/applications-module.h"
```

Step 2: Setup Wired Network (CSMA)

Create and configure your CSMA (Carrier-sense multiple access) nodes which represent the wired part of your network:

```cpp
NodeContainer csmaNodes;
csmaNodes.Create(3); // Creating 3 CSMA nodes

CsmaHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csma.SetChannelAttribute("Delay", TimeValue(MilliSeconds(2)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install(csmaNodes);
```

Step 3: Setup WiFi Network

For the WiFi part, configure your WiFi nodes and set up the mobility model for these nodes:

```cpp
NodeContainer wifiStaNodes, wifiApNode;
wifiStaNodes.Create(2); // Create 2 WiFi stations
wifiApNode = csmaNodes.Get(0); // Let's assume the first CSMA node is also a WiFi AP

YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
```

```
YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
phy.SetChannel(channel.Create());

WifiHelper wifi;
wifi.SetRemoteStationManager("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid("ns-3-ssid");
mac.SetType("ns3::StaWifiMac",
            "Ssid", SsidValue(ssid),
            "ActiveProbing", BooleanValue(false));

NetDeviceContainer staDevices;
staDevices = wifi.Install(phy, mac, wifiStaNodes);

mac.SetType("ns3::ApWifiMac",
            "Ssid", SsidValue(ssid));

NetDeviceContainer apDevices = wifi.Install(phy, mac, wifiApNode);
```

Step 4: Configure Mobility

Now, configure the mobility for the WiFi nodes. The mobility model defines how nodes move within the simulation environment:

```cpp
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                              "MinX", DoubleValue(0.0),
                              "MinY", DoubleValue(0.0),
                              "DeltaX", DoubleValue(5.0),
                              "DeltaY", DoubleValue(10.0),
                              "GridWidth", UintegerValue(3),
                              "LayoutType", StringValue("RowFirst"));

mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
                          "Bounds", RectangleValue(Rectangle(-50, 50, -50, 50)));
mobility.Install(wifiStaNodes); // Apply to WiFi station nodes
```

```
// Optionally set a fixed position for the WiFi AP
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiApNode);
```

Step 5: Install Internet Stack

Finally, install the internet stack on all nodes:

```cpp
InternetStackHelper stack;
stack.Install(csmaNodes);
stack.Install(wifiStaNodes);

Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces = address.Assign(csmaDevices);
address.NewNetwork();
Ipv4InterfaceContainer wifiStaInterfaces = address.Assign(staDevices);
address.Assign(apDevices);
```

Step 6: Running the Simulation

Setup applications, start simulation, and run the necessary events.

```cpp
Simulator::Run();
Simulator::Destroy();
```

This setup provides a basic scenario where some nodes are connected via a wired network and others can connect via WiFi, with mobility affecting only the WiFi stations. You can expand or modify this setup based on the specific requirements of your simulation.

HOW TO CONNECT WIRED AND WIFI

To connect wired nodes to WiFi access point nodes in an NS-3 simulation, you need to create a scenario where the access point (AP) serves as a bridge between the wired (typically using CSMA) and wireless networks. This involves configuring the access point to have both CSMA and WiFi interfaces, and setting up IP routing or bridging to allow communication between these different network segments. Here's a step-by-step guide on how to accomplish this:

Step 1: Include Necessary Headers

Ensure you have included the headers for both wired and wireless components, as well as those needed for bridging:

```cpp
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/bridge-module.h"
```

Step 2: Create Node Containers

Create separate node containers for the CSMA nodes and the WiFi stations. The AP node will be shared between the CSMA and WiFi networks:

```cpp
NodeContainer csmaNodes;
csmaNodes.Create(2); // Create 2 CSMA-only nodes

NodeContainer wifiStaNodes;
wifiStaNodes.Create(2); // Create 2 WiFi station nodes

NodeContainer wifiApNode;
wifiApNode.Create(1); // Create 1 WiFi AP node

csmaNodes.Add(wifiApNode.Get(0)); // Add the AP node to the CSMA node container
```

Step 3: Setup CSMA and WiFi Networks

Set up both the CSMA and WiFi networks. Note that the AP node has both CSMA and WiFi devices:

```cpp
// Setup CSMA
CsmaHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
```

```
csma.SetChannelAttribute("Delay", TimeValue(MilliSeconds(2)));

NetDeviceContainer csmaDevices = csma.Install(csmaNodes);

// Setup WiFi
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
phy.SetChannel(channel.Create());

WifiHelper wifi;
wifi.SetRemoteStationManager("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid("ns-3-ssid");
mac.SetType("ns3::StaWifiMac",
            "Ssid", SsidValue(ssid),
            "ActiveProbing", BooleanValue(false));

NetDeviceContainer staDevices = wifi.Install(phy, mac, wifiStaNodes);

mac.SetType("ns3::ApWifiMac",
            "Ssid", SsidValue(ssid));
NetDeviceContainer apDevices = wifi.Install(phy, mac, wifiApNode);
```

Step 4: Setup Mobility for WiFi Nodes
Configure mobility for WiFi stations and set a fixed position for the WiFi AP:
```cpp
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
                          "Bounds", RectangleValue(Rectangle(-50, 50, -50, 50)));
mobility.Install(wifiStaNodes);

mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiApNode);
```

Step 5: Install Internet Stack and Configure IP

Install the Internet stack and assign IP addresses to both networks:

```cpp
InternetStackHelper stack;
stack.Install(csmaNodes);
stack.Install(wifiStaNodes);

Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces = address.Assign(csmaDevices);
address.NewNetwork();
Ipv4InterfaceContainer wifiInterfaces = address.Assign(staDevices);
address.Assign(apDevices);
```

Step 6: Bridge the CSMA and WiFi Interfaces at the AP

To allow seamless communication between wired and wireless devices, bridge the CSMA and WiFi interfaces on the AP node:

```cpp
BridgeHelper bridge;
NetDeviceContainer bridgeDev;
bridgeDev.Add(csmaDevices.Get(2)); // Assuming the AP node's CSMA device is the third one
bridgeDev.Add(apDevices.Get(0)); // AP node's WiFi device
bridge.Install(wifiApNode.Get(0), bridgeDev);
```

Step 7: Run the Simulation

```cpp
Simulator::Run();
Simulator::Destroy();
```

This setup uses a bridge at the access point to link the wired and wireless segments directly at Layer 2, enabling devices on both networks to communicate as if they were part of the same collision domain. This configuration is typical in scenarios where an access point is connected to a wired network, and you want to ensure that all nodes, regardless of how they are connected (wired or wireless), can communicate with each

HOW TO USE MOBILITYMODEL FOR CSMA :
Setting up mobility for CSMA nodes in an NS-3 simulation typically isn't common practice since CSMA (Carrier Sense Multiple Access) is often used to simulate wired networks, and nodes in such networks are usually static. However, in some scenarios—like simulating mobile car networks or robots in a wired setting—you might need to assign mobility models to CSMA nodes. Here's how you can set up mobility for CSMA nodes in NS-3:

Step 1: Include Necessary Headers

Ensure you have included the headers for the core, network, CSMA modules, and importantly, the mobility module:

```cpp
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/mobility-module.h"
```

Step 2: Create Node Containers

Create a node container for the CSMA nodes that will have mobility:

```cpp
NodeContainer csmaNodes;
csmaNodes.Create(3); // Example: Creating 3 CSMA nodes that will have mobility
```

Step 3: Setup CSMA Network

Set up the CSMA network. Even though these nodes will be "mobile", they are still part of a typical CSMA network:

```cpp
CsmaHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csma.SetChannelAttribute("Delay", TimeValue(MilliSeconds(2)));

NetDeviceContainer csmaDevices = csma.Install(csmaNodes);
```

Step 4: Setup Mobility Models

Configure the mobility models for these CSMA nodes. You can choose from various mobility models based on your simulation needs. For example, you might use `RandomWalk2dMobilityModel` for nodes moving within a 2D space with random speeds and directions:

```cpp
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::RandomRectanglePositionAllocator",
                              "X", PointerValue(CreateObjectWithAttributes<UniformRandomVariable>("Min", DoubleValue(0), "Max", DoubleValue(100))),
```

"Y", PointerValue(CreateObjectWithAttributes<UniformRandomVariable>("Min", DoubleValue(0), "Max", DoubleValue(100))));

mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",

"Bounds", RectangleValue(Rectangle(0, 100, 0, 100)),

"Distance", DoubleValue(50.0), // Distance to change direction

"Speed", StringValue("ns3::UniformRandomVariable[Min=0.0|Max=2.0]")); // Random speed between 0 and 2 m/s

mobility.Install(csmaNodes);

```

### Step 5: Install Internet Stack and Configure IP

Install the Internet stack and assign IP addresses. While not directly related to mobility, this step is necessary for any network communication:

```cpp

InternetStackHelper stack;

stack.Install(csmaNodes);

Ipv4AddressHelper address;

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

Ipv4InterfaceContainer csmaInterfaces = address.Assign(csmaDevices);

```

Step 6: Run the Simulation

Now, run your simulation:

```cpp

Simulator::Run();

Simulator::Destroy();

```

Additional Considerations

- **Mobility Impact**: In real-world scenarios, CSMA nodes are typically stationary because they are connected via cables. Simulating mobility for such nodes might be conceptually incorrect unless justified by your specific simulation scenario (e.g., moving robots connected by a tether).

- **Model Choice**: Choose your mobility model based on the scenario. For example, `ConstantPositionMobilityModel` can be used if nodes only need to be placed at specific positions but don’t move, or `RandomWaypointMobilityModel` for more dynamic scenarios.

This setup allows you to simulate "mobile" CSMA nodes, which can be useful in studying the network behavior in dynamic environments, such as nodes on moving vehicles in a controlled area.
```