Mesh Network Optimizer Proje Raporu

Proje kapsamında çalışmalara öncelikle bir literatür araştırması yapılarak başlanmıştır. Bu anlamda Mesh ağ yapıları, Routing algoritmaları, Node/Link ve Path metrikleri, Graph Kavramı, Feature Engineering, Supervised ve Reinforcement Learning kavramları üzerinde araştırmalar yapılmış ve konu detaylıca araştırılmıştır. Ardından projenin gerçekleştirilmesinde kullanılacak simülasyon ortamı olarak ns-3 seçilmiş ve bu doğrultuda ns-3.46 sürümünün kurulum çalışmalarına başlanmıştır.

Projede amaç şudur: ns-3 üzerinde çok hoplamalı bir mesh ağ topolojisi kurularak düğümler arası RSSI, gecikme, enerji ve yük gibi metrikler simüle edilir. Bu metrikler kullanılarak her düğümde bir ML/RL tabanlı yönlendirme modeli eğitilir veya çalıştırılır ve paketler dinamik olarak en uygun next-hop üzerinden iletilir. Elde edilen performans, statik routing algoritmaları ile gecikme, enerji tüketimi ve paket başarı oranı açısından karşılaştırılır.

Projede ilk aşamada üzerinde kafa yorulan konular, ns-3 ortamının kurulumunda yaşanılabilecek zorluklar hangi ns-3 versiyonunun kurulmasının en doğru olduğu, LoPy4 cihazlarının projeye nasıl entegre edilebileceği, kullanılacak kod yapılarının nereye kaydedileceği (örneğin ns-3'nin tar dosyası dışarı aktarılınca oluşacak klasörün vsc ortamında mı düzenleneceği yoksa ayrı bir proje klasörü mü oluşturulacağı) üzerinde olmuştur. Bu anlamda yapılan araştırmalarda projenin iki katmanlı olduğu ve ns-3 ve LoPy4'ün bu iki katmanı temsil ettiği, LoPy4'ün ns-3 üzerine gömülemeyeceği öğrenilmiştir. Ardından ns-3'nin versiyonları arasından hangi versiyonun kullanılmasının proje açısından daha verimli olacağı LLM kullanımı ile araştırılmış ve ns-3.39 sürümünün kullanılmasının daha verimli olacağı cevabı alınmıştır. Projenin görsel sunumu için ise ns-3'nin NetAnim özelliğinin kullanılmasına karar verilmiştir.

Tüm bu literatür ve kavram araştırmalarından sonra sıra proje açıklama metninde Hafta 1-2 başlığı altında yer alan Altyapı ve Veri Toplama görevlerinin icrasına gelmiştir bu anlamda yapılan işlemler aşağıdaki paragraflarda izah edilmiştir.

Projenin ilk aşamasını ifade eden bu kısımda ML ve RL yapılarından ziyade çalışan bir mesh ağ + ölçüm yapan bir routing altyapısı oluşturulmaya çalışılmıştır. Bu işlem için öncelikle "https://www.nsnam.org/releases/ns-3-39/" internet adresinden ilgili versiyonun tar dosyası indirilmiş ve dışarı aktarılarak İndirilenler klasörünün içerisine kaydedilmiştir. Ardından ns-3 dosyası örnekler ve testler dışarıda tutularak yapılandırılmış ve bu şekilde derlenmiştir derleme işlemi tamamlandıktan sonra ise test komutu ile ns-3 ile birlikte gelen testler çalıştırılmış ve 700 civarı testin tamamının başarıyla geçtiği görülmüştür. Bununla birlikte ilk simülasyon da çalıştırılmıştır.

Ns-3 simülatörünün kurulumu başarıyla tamamlandıktan sonra ns-3'nin ana dizinine geçilmiş ve scratch klasörünün içine proje topolojisinin net olarak görülebilmesi için

gerekli xml dosyasının oluşturulmasını sağlayan bir mesh-optimizer.cc dosyası oluşturulmuştur. Oluşturulan bu dosyada 5 düğüm vardır ve düğümler arasındaki mesafe 50 metre olarak belirlenmiştir. Bu işlemin ardından ns3 'build' komutuyla yeniden derlenmiş ve ardından oluşturulan bu dosya 'run' komutu ile çalıştırılmıştır. Bu işlemin ardından ns-3.39 dizininde mesh-sim.xml adında bir dosya oluşturulmuştur.

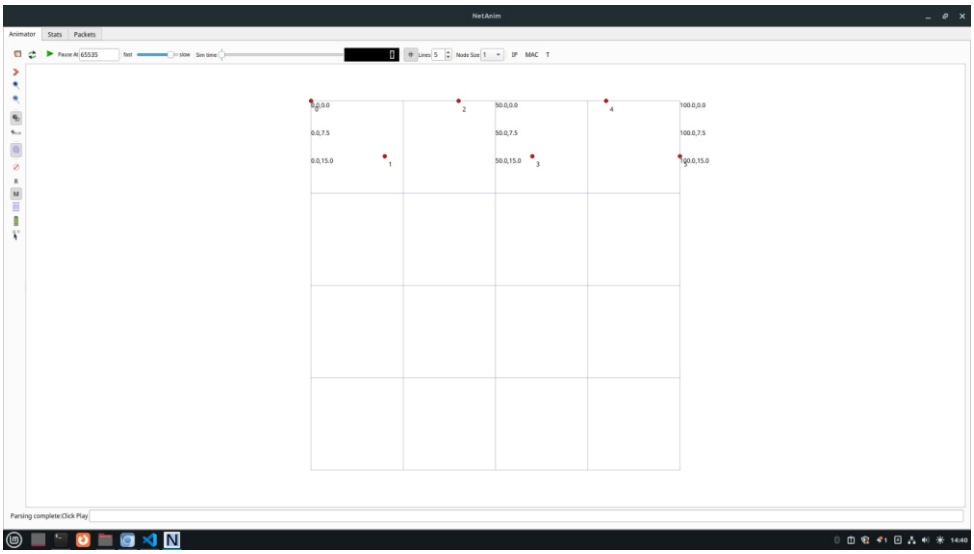
Bu işlemden sonra sıra simülasyon ortamının görsel sunumu için kullanılacak olan NetAnim'in kurulmasına gelmiştir. Bu işlem için öncelikle qmake komutunun ve qt6 paketlerinin yüklenmesi gerekmektedir aşağıda yer alan komutlar ile bu işlem gerçekleştirilmiştir.

```
sudo apt update
sudo apt install qt6-base-dev qt6-multimedia-dev qt6-base-dev-tools -y
```

Bu işlemin ardından ns-3 bir daha yapılandırılmış ve derlenmiştir. İşlem tamamlandıktan sonra build klasörünün içerisinde NetAnim dosyası oluşmuştur. Oluşan bu dosyanın başarı ile açıldığı farkedildikten sonra VS Code ortamında mesh_optimizer.cc dosyasına tekrar geçilmiş ve bu dosyada öncelikle include komutu ile netanim modülü dahil edilmiştir ardından dosyanın sonundaki "Simulator::Run();" satırından hemen önce aşağıdaki komut yazılarak projenin xml dosyası kaydedilmiştir.

```
AnimationInterface anim ("mesh-projesi.xml");
```

Bu işlem de tamamlandıktan sonra tekrar ana dizine dönülerek run komutu ile mesh_optimizer dosyası çalıştırılmış ve oluşan xml dosyası açılarak mesh ağı görsel olarak aşağıdaki gibi görülmeye başlanmıştır.



Proje çalışmaları kapsamında ilk hafta bu şekilde noktalanmıştır. Çalışmalar kapsamında işlemlere sonraki hafta mesh-optimizer.cc dosyasında yapılacak düzenlemeler ile RSSI değerlerinin okunması ve bu değerlerin daha sonra bir yapay zeka modeli ile yorumlanabilmesi için csv uzantılı bir dosyaya kaydedilmesi ile devam edilmiştir. Bu doğrultuda yapılan işlemler aşağıdaki paragraflarda yer almaktadır.

İkinci haftada çalışmalara geçen hafta oluşturulan mesh-optimizer.cc dosyasında düzenlemeler gerçekleştirilerek terminal üzerinden RSSI değerlerinin okunması ve sonrasında bu verilerin ileride bir yapay zeka ajanı ile yorumlanarak en sağlıklı rota üzerinden iletime devam edilmesi için csv uzantılı bir dosyaya kaydedilmesine yönelik çalışmalar gerçekleştirilerek devam edilmiştir.

Bu doğrultuda mesh_optimizer.cc dosyasında birtakım düzenlemeler yapılmış ancak ilk aşamada MeshHelper bileşeninin bulunamaması ve ns3'ün trace (takip) mekanizmasındaki imza uyuşmazlıkları gibi sorunlardan kaynaklı hatalarla karşılaşılmıştır. Ardından ise hatalara yönelik çözümlemeler yapılmış ve son durumda mesh-optimizer.cc dosyası aşağıdaki görsellerdeki gibi düzenlenmiştir.

```
#include "ns3/core-module.h"
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/netanim-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

// HATA BURADAYDI: Parametre listesine 'double txPowerW' eklendi.
// ns-3.39 PhyTxBegin izlemesi hem paket hem de güç değeri (Watt) gönderir.
void MeshPointTrace (Ptr<const Packet> packet, double txPowerW)
{
    std::cout << "Paket Gonderildi - Zaman: " << Simulator::Now ().GetSeconds ()
              << "s | Guc: " << txPowerW << " W" << std::endl;
}

int main (int argc, char *argv[])
{
    uint32_t nNodes = 5;
    Time simTime = Seconds (10.0);

    NodeContainer nodes;
    nodes.Create (nNodes);

    // Mobilite ayarları
    MobilityHelper mobility;
    mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                   "MinX", DoubleValue (0.0),
                                   "MinY", DoubleValue (0.0),
                                   "DeltaX", DoubleValue (20.0),
                                   "DeltaY", DoubleValue (20.0),
                                   "GridWidth", UintegerValue (3),
                                   "LayoutType", StringValue ("RowFirst"));
    mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
    mobility.Install (nodes);

    // WiFi ve Mesh ayarları
    YansWifiPhyHelper phy;
    YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
    phy.SetChannel (channel.Create ());

    MeshHelper mesh = MeshHelper::Default ();
    mesh.SetStackInstaller ("ns3::Dot11sStack");
```

```
    NetDeviceContainer devices = mesh.Install (phy, nodes);

    // IP yığını ayarları
    InternetStackHelper internet;
    internet.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (devices);

    // UDP Trafiği oluşturma (Düğüm 0 -> Düğüm 4)
    uint16_t port = 9;
    UdpEchoServerHelper echoServer (port);
    ApplicationContainer serverApps = echoServer.Install (nodes.Get (nNodes - 1));
    serverApps.Start (Seconds (1.0));
    serverApps.Stop (simTime);

    UdpEchoClientHelper echoClient (interfaces.GetAddress (nNodes - 1), port);
    echoClient.SetAttribute ("MaxPackets", UintegerValue (100));
    echoClient.SetAttribute ("Interval", TimeValue (Seconds (0.1)));
    echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
    clientApps.Start (Seconds (2.0));
    clientApps.Stop (simTime);

    // Trace bağlantısı (Buradaki imza artık MeshPointTrace ile uyumlu)
    Config::ConnectWithoutContext ("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/Phy/PhyTxBegin",
                                   MakeCallback (&MeshPointTrace));

    // NetAnim Çıktısı
    AnimationInterface anim ("mesh-anim.xml");

    std::cout << "Simulasyon basliyor... Paketler gonderiliyor!" << std::endl;

    Simulator::Stop (simTime);
    Simulator::Run ();

    std::cout << "Simulasyon bitti. Veriler kaydedildi." << std::endl;

    Simulator::Destroy ();

    return 0;
}
```

Bu düzenlemenin ardından ‘./ns3 run scratch/mesh-optimizer’ komutu ile dosya çalıştırılmış ve RSSI değerlerinin terminalden okunabildiği aşağıdaki gibi görülmeye başlanmıştır.

```
19.9013,1102,-74.9707
19.9013,1102,-79.4862
19.9013,14,-74.9707
19.9013,14,-74.9707
19.9013,14,-74.9707
19.9013,14,-74.9707
19.9013,14,-79.4862
19.9013,14,-79.4862
19.9013,14,-79.4862
19.9013,14,-79.4862
19.9019,1102,-74.9707
19.9019,1102,-74.9707
19.9019,1102,-74.9707
19.9019,1102,-74.9707
19.9019,1102,-79.4862
19.9019,1102,-79.4862
19.9019,1102,-79.4862
19.9019,1102,-79.4862
19.9019,14,-74.9707
19.9019,14,-74.9707
19.9019,14,-79.4862
19.9958,85,-74.9707
19.9958,85,-74.9707
19.9958,85,-74.9707
19.9958,85,-79.4862
19.9958,85,-79.4862
kagan@debian:~/İndirilenle
```

Verilerin alınabildiği görüldükten sonra sıra proje açıklanmasında 2. Hafta başlığının altında söylendiği gibi bu verilerin sonrasında işlenmek üzere kaydedilmesine gelmiştir. Bu işlem için mesh-optimizer.cc dosyası içerisinde ufak bir düzenleme yapılarak 'mesh_performance_data.csv' isminde bir dosya oluşturulmuş ve kod çalıştırıldıktan sonra cat komutu ile bu dosyanın içeriğine bakılarak ilgili verilerinin bu dosyaya işlendiği görülmüştür.

```
Paket Gonderildi - Zaman: 9.5s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.5002s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.50027s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.50047s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.6s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.6002s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.60027s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.60047s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.7s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.7002s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.70028s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.70048s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.78213s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.8s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.8002s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.80028s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.80048s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.80539s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.84919s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.85374s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.90001s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.90021s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.90028s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.90048s | Guc: 0.04 W
Paket Gonderildi - Zaman: 9.99568s | Guc: 0.04 W
Simulasyon bitti. Veriler kaydedildi.
kagan@debian:~/İndirilenler/ns-allinone-3.39/ns-3.39$
```

Fakat yukarıda da görüleceği üzere şu anda iletilen veriler arasında RSSI verileri bulunmamaktadır. Bu sebeple sonraki aşamada RSSI verilerinin yakalanması için kodda bir düzenleme daha yapılmıştır. Bu aşamada ise WifiNetDevice modülünün kullanılması durumunda mesh ağlarda cihazların birer MeshPointDevice olarak tanımlanmasından kaynaklı olarak enerji modelinin doğrudan bir Mesh Point üzerine kurulamayıp bunun yerine onun altındaki fiziksel arayüzlere kurulmasının gerekmesinden kaynaklı hatalarla karşılaşılmıştır. Son durumda mesh-optimizer.cc dosyasına RSSI değerlerinin yakalanmasını sağlayan aşağıdaki fonksiyon eklenmiştir.

```
// RSSI verisini yakalayan fonksiyon
void TraceRssi (std::string context, Ptr<const Packet> packet, uint16_t channelFreqMhz,
                WifiTxVector txVector, MpduInfo mpduInfo, SignalNoiseDbm signalNoise, uint16_t staId)
{
    if (g_csvFile.is_open()) {
        g_csvFile << Simulator::Now ().GetSeconds () << ","
                  << packet->GetSize () << ","
                  << signalNoise.signal << std::endl;
    }
}
```

Bu işlemin ardından mesh-optimizer.cc dosyası çalıştırılıp, mesh_rssi_dataset.csv dosyası 'cat' komutu ile okunduğunda içinde "19.9009,1102,-74.9707" gibi pek çok satırın olduğu görülmüştür.

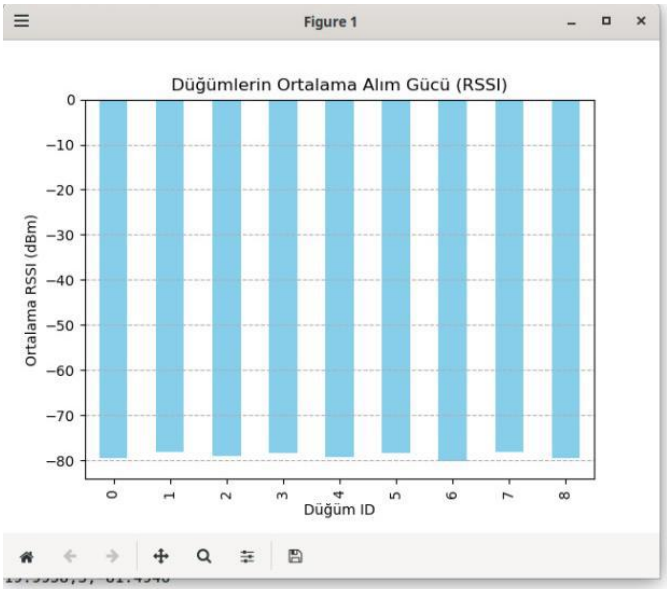
RSSI verilerinin başarıyla kaydedilmesinin ardından sıra sistemde bulunan he bir düğüme belirli bir düğüm numarası verilmesine gelmiştir. Bu sayede oluşturacağımız yapay zeka ajanı hangi düğümün daha verimli olduğuna karar verdikten sonra ilgili düğüm numarasını kolaylıkla iletebilecektir. Bu işlem için az önce paylaşılan TraceRssi fonksiyonu şu şekilde düzenlenmiştir:

```
void TraceRssi (std::string context, Ptr<const Packet> packet, uint16_t channelFreqMhz,
                WifiTxVector txVector, MpduInfo mpduInfo, SignalNoiseDbm signalNoise,
uint16_t staId)
{
    // Context içinden düğüm ID'sini ayıkla (Örn: /NodeList/5/...)
    std::string sub = context.substr (10);
    uint32_t pos = sub.find ("/");
    std::string nodeID = sub.substr (0, pos);

    if (g_csvFile.is_open()) {
        g_csvFile << Simulator::Now ().GetSeconds () << ","
                  << nodeID << ","
                  << signalNoise.signal << std::endl;
    }
}
```

Bu düzenlemenin ardından ise verinin sıra verinin bulunduğu CSV uzantılı dosyayı bir Python scripti ile basitçe okuyup veri akışının doğru sağlanıp sağlanmadığının kontrolüne gelmiştir. Bu işlem için öncelikle ns-3.39 dizininde analyze_mesh.py adında bir dosya oluşturulmuş ve içine şu kod eklenmiştir:

```
import pandas as pd
import matplotlib.pyplot as plt

# CSV dosyasını oku
data = pd.read_csv('mesh_rssi_dataset.csv')

# Düğüm bazlı ortalama RSSI değerlerini hesapla
node_stats = data.groupby('Receiver_Node_ID')['RSSI_dBm'].mean()

print("--- Düğüm Bazlı Sinyal Analizi ---")
print(node_stats)

# Grafiğe dök
node_stats.plot(kind='bar', color='skyblue')
plt.title('Düğümlerin Ortalama Alım Gücü (RSSI)')
plt.xlabel('Düğüm ID')
plt.ylabel('Ortalama RSSI (dBm)')
plt.grid(axis='y', linestyle='--')
plt.show()
```

Bu işlemin ardından ‘python3 analyze_mesh.py’ komutu ile terminal üzerinden dosya çalıştırıldığında aşağıdaki görsel elde edilmiştir.




```
19.9958,5,-81.4946
kagan@debian:~/İndirilenler/ns-allinone-3.39/ns-3.39$ python3 analyze_mesh.py
--- Düğüm Bazlı Sinyal Analizi ---
Receiver_Node_ID
0   -79.407154
1   -78.240337
2   -79.028660
3   -78.289401
4   -79.325142
5   -78.268284
6   -80.105215
7   -78.249297
8   -79.491635
Name: RSSI_dBm, dtype: float64
```

Bu görsel ve görselde görülen veriler bize verinin homojen olduğunu göstermektedir. Yani tüm düğümler birbirlerini yaklaşık olarak aynı kalitede duyabilmektedir. Ancak proje kapsamına göre ağda bazı zorlukların olması ve hangi düğümün daha iyi bir iletime sahip olduğunun bir yapay zeka ajanı tarafından seçilmesi istendiğinden son hafta mesh-optimizer.cc dosyasında birtakım düzenlemeler daha yapılarak düğümler arası mesafe değiştirilmiş ve bu sayede RSSI değerlerinin bu kadar yakın olmaması sağlanmaya çalışılmıştır. Ayrıca projeye yapay zeka entegrasyonu ile ilgili işlemler de son hafta gerçekleştirilmiştir. Son hafta yapılan işlemler ise aşağıdaki paragraflarda özetlenmiştir.

Proje kapsamında son hafta yapılan çalışmalar ile proje büyük oranda bitirilmiştir. Öncelikle geçen hafta terminal üzerinden görülen ve csv uzantılı bir dosyaya kaydedilen RSSI verilerinin yorumlanması için bir yapay zeka modeli geliştirilmeye çalışılmıştır. Bu kapsamda ilk seçenek olarak ns3-ai modülünün kullanılmasına karar verilmiştir.

Ns3-ai modülü resmi github sitesi üzerinden zip dosyası olarak indirilmiş ve içindeki ns-3 main klasörü ayıklanarak ns3-ai olarak yeniden isimlendirilmiş ve projede içe aktarılarak kullanılmak üzere ns-3.39 dizini içerisinde yer alan contrib dizininim içerisine yapıştırılmıştır. Ardından ise mesh-optimizer.cc dosyasında içe aktarma işlemini gerçekleştirecek komutla beraber gelen RSSI değerlerinin farklılaşması için ilgili kodlar da yazılmış ve run komutuyla dosya çalıştırılmıştır. Bu noktada terminalde karşımıza ns3-ai

ile ilgili birtakım kütüphane eksikliklerine dayalı hatalar çıkmıştır bunlardan kısaca bahsetmek gerekirse ilgili kütüphaneler şunlardır: Boost, pybind11, Protobuf. Bu kütüphaneleri de sisteme dahil edip ns3'ü derlemeye çalıştığımızda ise ns3-ai modülü sürekli olarak derlenemeyecek modüller listesinde yer almaya devam etmiştir dolayısıyla verilerin yorumlanabilmesi için ns3-ai modülünden ziyade Python içerisinde yer alan socket kütüphanesinin kullanılmasına zorunlu olarak karar verilmiştir.

Bu işlemde mantık şudur: Proje iki farklı terminal üzerinden çalıştırılacaktır. Bu terminallerden biri ns-3 terminali diğeri ise Python terminalidir. Python terminali ajan terminalidir. Yani kısaca gelen verilerin alınıp yorumlanacağı terminaldir. Dolayısıyla öncelikle 'python3' komutu ile Python terminali çalıştırılır ve ajan dinlemeye geçer. Ardından ise ns-3 terminali çalıştırılır ve veri akışı sağlanır.

Bu işlem kapsamında ns-3.39 ana dizininde Python ajanının yer alacağı rl_bridge.py isimli bir Python dosyası oluşturulur. Ve ardından proje sunumunda yer alacak olan mesh-optimizer.cc ve rl_bridge.py dosyaları şu şekilde doldurulur:

Mesh_optimizer.cc Dosyası:

```
#include "ns3/core-module.h"
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/netanim-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string>

using namespace ns3;

int g_sock = 0;
struct sockaddr_in g_serv_addr;

void SetupExternalSocket() {
    g_sock = socket(AF_INET, SOCK_DGRAM, 0);
    g_serv_addr.sin_family = AF_INET;
    g_serv_addr.sin_port = htons(5555);
    inet_pton(AF_INET, "127.0.0.1", &g_serv_addr.sin_addr);
}

void TracePhy (std::string context, Ptr<const Packet> packet, uint16_t channelFreqMhz,
               WifiTxVector txVector, MpduInfo mpduInfo, SignalNoiseDbm signalNoise, uint16_t staId)
{
    std::string sub = context.substr (10);
    size_t pos = sub.find ("/");
    std::string nodeID = sub.substr (0, pos);

    std::string msg = nodeID + "," + std::to_string(signalNoise.signal);
    sendto(g_sock, msg.c_str(), msg.length(), 0, (struct sockaddr *)&g_serv_addr, sizeof(g_serv_addr));
}

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

    uint32_t nNodes = 9;
    Time simTime = Seconds (20.0);

    NodeContainer nodes;
    nodes.Create (nNodes);
```

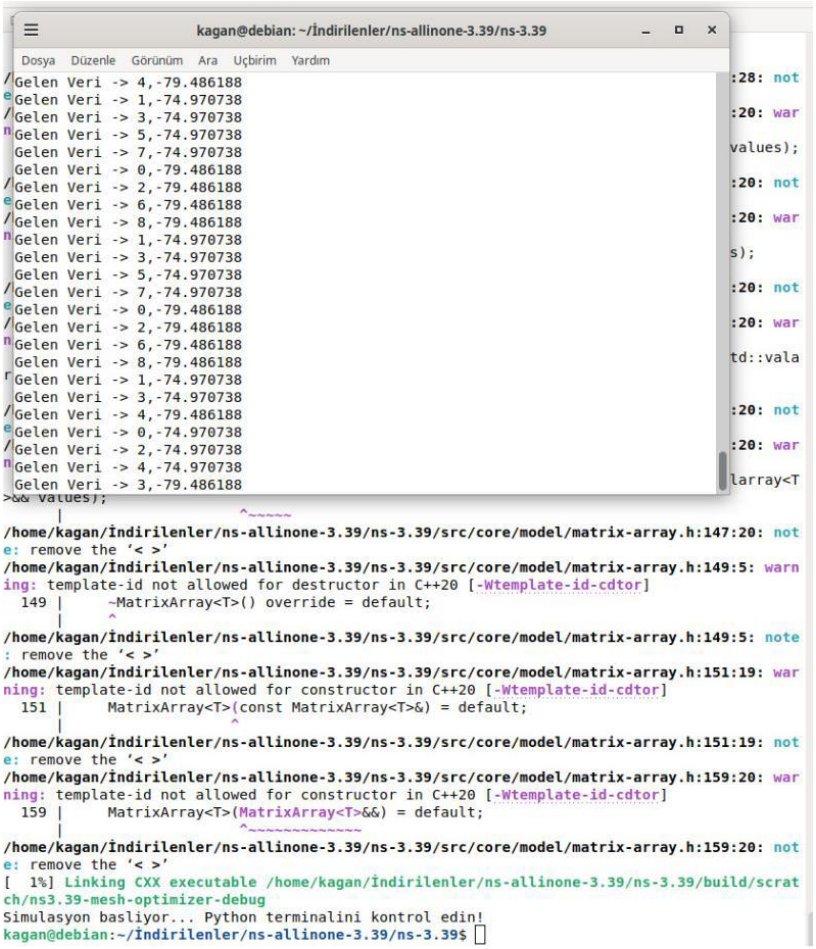
```
    MobilityHelper mobility;
    mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                   "DeltaX", DoubleValue (20.0),
                                   "DeltaY", DoubleValue (20.0),
                                   "GridWidth", UintegerValue (3),
                                   "LayoutType", StringValue ("RowFirst"));
    mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
    mobility.Install (nodes);

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

    MeshHelper mesh = MeshHelper::Default ();
    mesh.SetStackInstaller ("ns3::Dot11sStack");
    NetDeviceContainer meshDevices = mesh.Install (phy, nodes);

    InternetStackHelper internet;
    internet.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (meshDevices);

    UdpEchoServerHelper echoServer (9);
    ApplicationContainer serverApps = echoServer.Install (nodes.Get (8));
    serverApps.Start (Seconds (1.0));

    UdpEchoClientHelper echoClient (interfaces.GetAddress (8), 9);
    echoClient.SetAttribute ("MaxPackets", UintegerValue (100000));
    echoClient.SetAttribute ("Interval", TimeValue (Seconds (0.01)));
    echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
    clientApps.Start (Seconds (2.0));

    Config::Connect ("/NodeList//DeviceList//$ns3::WifiNetDevice/Phy/MonitorSnifferRx",
                     MakeCallback (&TracePhy));

    std::cout << "Simulasyon Basliyor (Mesafe 20m)..." << std::endl;
    Simulator::Stop (simTime);
    Simulator::Run ();

    if (g_sock != 0) close(g_sock);
    Simulator::Destroy ();
    return 0;
}
```

Rl_bridge.py Dosyası:

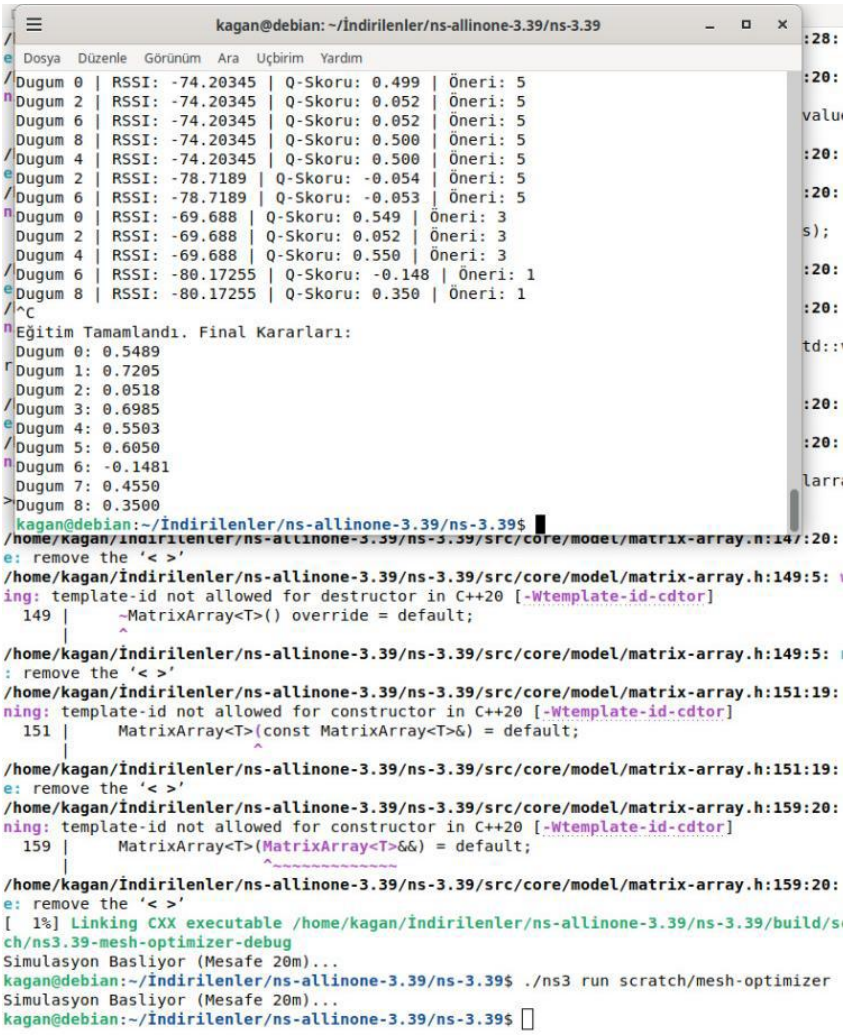
```
import socket
import numpy as np

UDP_IP = "127.0.0.1"
UDP_PORT = 5555
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock.bind((UDP_IP, UDP_PORT))

q_values = {str(i): 0.5 for i in range(9)}
learning_rate = 0.1  # Yeni bilginin ne kadar hızlı öğrenileceği
alpha = 0.2          # Ödül katsayısı

print(f"--- MESH AI AGENT AKTIF ---")
print("Dinamik rota analizi yapılıyor...")

try:
    while True:
        data, addr = sock.recvfrom(1024)
        msg = data.decode('utf-8').split(',')
        node_id, rssi = msg[0], float(msg[1])

        # 1. ÖDÜL (REWARD) FONKSİYONU
        # RSSI ne kadar yüksekse (0'a yakınsa) o kadar yüksek ödül
        # -75 dBm altı (örn -80) riskli bölge olarak kabul edilir
        if rssi > -72:
            reward = 1.0  # Harika bağlantı
        elif rssi > -78:
            reward = 0.5  # Kabul edilebilir
        else:
            reward = -1.0 # Kötü bağlantı (Cezalandır)

        old_q = q_values[node_id]
        q_values[node_id] = old_q + learning_rate * (reward - old_q)

        best_node = max(q_values, key=q_values.get)

        if int(node_id) % 2 == 0: # Ekran kirliliğini önlemek için seçici yazdırıyoruz
            print(f"Dugum {node_id} | RSSI: {rssi} | Q-Skoru: {q_values[node_id]:.3f} | Öneri: {best_node}")

except KeyboardInterrupt:
    print("\nEğitim Tamamlandı. Final Kararları:")
    for n, q in q_values.items():
        print(f"Dugum {n}: {q:.4f}")
```

Dosyalar bu şekilde ayarlandıktan sonra iki adet terminal açılır. Bunlardan biri Python terminali diğeri ise ns-3 terminalidir.

Terminaller açıldıktan sonra ilk olarak Python terminalinden “python3 rl_bridge.py” komutu ile ajan aktif hale getirilir. Ardından ise ns-3 terminaline geçilir ve “./ns3 run mesh-optimizer” komutu ile veri gönderimi gerçekleştirilir bu işlemlerin ardından verinin Python tarafından alınmasına ilişkin iki terminale ait ekran görüntüsü aşağıda yer almaktadır.



Verilerin Python terminaline ulaşması bu şekilde gerçekleşmiştir ancak bu aşamada Python tarafı sadece dinleme aşamasındadır. Ajanı daha zeki bir karar verici konumuna getirmek için kodlarda birtakım daha düzenlemeler yapılmış ve Python tarafına Q-learning tabanlı yeni bir kod eklenmiştir Aynı zamanda gelene verilerin RSSI değerlerinin birbirine çok yakın olmaması ve ajanın daha sağlıklı bir seçim yapabilmesi için ns-3 tarafındaki mesh-optimizer.cc dosyasında da birtakım düzenlemeler yapılmıştır. Son durumda terminalde ajanın verdiği karara ilişkin ekran görüntüsü bir sonraki sayfada yer almaktadır.



Yukarıdaki ekran görüntüsünde görüleceği üzere Python tarafında ajan sistemde yer alan toplam 9 düğüm üzerinden hangi düğümün daha verimli olduğuna ilişkin bir değerlendirme yapabilmektedir ve buna ilişkin bir öneri sunabilmektedir.

Bu doğrultuda projenin son adımına geçilmiştir bu adımda amaç ajanın verdiği kararın ns-3 tarafına iletilmesi ve sonrasında ns-3 tarafının Python tarafında verilen en iyi düğüm kararına göre rotasını anlık olarak güncellemesidir.

Bu amaç doğrultusunda mesh-optimizer.cc ve rl_bridge.py dosyalarında son olarak yapılan düzenlemeler ile elde edilen nihai dosyalar aşağıdaki gibidir.

Mesh-optimizer.cc Dosyası:

```
#include "ns3/core-module.h"
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/netanim-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h> // HATA ÇÖZÜMÜ: fcntl ve O_NONBLOCK için gerekli
#include <string>

using namespace ns3;

// Global soket değişkenleri
int g_outSock = 0;
int g_inSock = 0;
struct sockaddr_in g_sendAddr, g_recvAddr;

// Python ile çift yönlü haberleşme kurulumu
void SetupSockets() {
    // Gönderici Soket (ns-3 -> Python | Port: 5555)
    g_outSock = socket(AF_INET, SOCK_DGRAM, 0);
    g_sendAddr.sin_family = AF_INET;
    g_sendAddr.sin_port = htons(5555);
    inet_pton(AF_INET, "127.0.0.1", &g_sendAddr.sin_addr);

    // Alıcı Soket (Python -> ns-3 | Port: 5556)
    g_inSock = socket(AF_INET, SOCK_DGRAM, 0);
    g_recvAddr.sin_family = AF_INET;
    g_recvAddr.sin_addr.s_addr = INADDR_ANY;
    g_recvAddr.sin_port = htons(5556);
    bind(g_inSock, (struct sockaddr *)&g_recvAddr, sizeof(g_recvAddr));

    // Alıcıyı beklemesiz (non-blocking) moda al
    fcntl(g_inSock, F_SETFL, O_NONBLOCK);
}

void TracePhy (std::string context, Ptr<const Packet> packet, uint16_t channelFreqMhz,
               WifiTxVector txVector, MpduInfo mpduInfo, SignalNoiseDbm signalNoise, uint16_t staId)
{
    // Düğüm ID ayıklama
    std::string sub = context.substr (10);
    size_t pos = sub.find ("/");
```

```
    std::string nodeID = sub.substr (0, pos);

    // 1. Python'a Durumu Bildir
    std::string msg = nodeID + "," + std::to_string(signalNoise.signal);
    sendto(g_outSock, msg.c_str(), msg.length(), 0, (struct sockaddr *)&g_sendAddr, sizeof(g_sendAddr));

    // 2. Python'dan Gelen Kararı Dinle
    char buffer[1024];
    int len = recv(g_inSock, buffer, sizeof(buffer) - 1, 0);
    if (len > 0) {
        buffer[len] = '\0';
        std::cout << "--- [AI KARARI] Hedef Dugum: " << buffer << " uzerinden rota guncellendi! ---" << std::endl;
    }
}

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

    uint32_t nNodes = 9;
    NodeContainer nodes;
    nodes.Create (nNodes);

    MobilityHelper mobility;
    mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                   "DeltaX", DoubleValue (30.0),
                                   "DeltaY", DoubleValue (30.0),
                                   "GridWidth", UintegerValue (3),
                                   "LayoutType", StringValue ("RowFirst"));
    mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
    mobility.Install (nodes);

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

    MeshHelper mesh = MeshHelper::Default ();
    mesh.SetStackInstaller ("ns3::Dot11sStack");
    NetDeviceContainer meshDevices = mesh.Install (phy, nodes);

    InternetStackHelper internet;
    internet.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (meshDevices);
```

```
    UdpEchoServerHelper echoServer (9);
    ApplicationContainer serverApps = echoServer.Install (nodes.Get (8));
    serverApps.Start (Seconds (1.0));

    UdpEchoClientHelper echoClient (interfaces.GetAddress (8), 9);
    echoClient.SetAttribute ("MaxPackets", UintegerValue (5000));
    echoClient.SetAttribute ("Interval", TimeValue (Seconds (0.05)));
    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
    clientApps.Start (Seconds (2.0));

    Config::Connect ("/NodeList//DeviceList//$ns3::WifiNetDevice/Phy/MonitorSnifferRx",
                     MakeCallback (&TracePhy));

    std::cout << "Kapali Dongu RL Simulasyonu Basliyor..." << std::endl;
    Simulator::Stop (Seconds (20.0));
    Simulator::Run ();

    close(g_outSock);
    close(g_inSock);
    Simulator::Destroy ();
    return 0;
}
```

Rl_bridge.py Dosyası:

```
import socket

# Gelen veri kanalı (ns-3 -> Python)
IN_IP, IN_PORT = "127.0.0.1", 5555
in_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
in_sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
in_sock.bind((IN_IP, IN_PORT))

# Karar gönderim kanalı (Python -> ns-3)
OUT_IP, OUT_PORT = "127.0.0.1", 5556
out_sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# RL Hafızası
q_values = {str(i): 0.5 for i in range(9)}
alpha = 0.1

print(f"--- [AI AJANI] Karar Merkezi Baslatildi ---")

try:
    while True:
        data, addr = in_sock.recvfrom(1024)
        msg = data.decode('utf-8').split(',')
        node_id, rssi = msg[0], float(msg[1])

        # Ödül/Ceza Mantığı
        reward = 1.0 if rssi > -72 else (-1.0 if rssi < -78 else 0.1)

        # Q-Skoru Güncelleme
        q_values[node_id] += alpha * (reward - q_values[node_id])

        # Karar: En yüksek puana sahip düğümü ns-3'e fırlat
        best_node = max(q_values, key=q_values.get)
        out_sock.sendto(best_node.encode(), (OUT_IP, OUT_PORT))

        # Her 10 pakette bir durumu yazdır
        if int(time.time() * 10) % 5 == 0:
            print(f"Analiz: Dugum {node_id} RSSI: {rssi} | Karar: {best_node} uzerinden git.")

except KeyboardInterrupt:
    print("\nAjan durduruldu.")
except Exception as e:
    print(f"Hata: {e}")
```

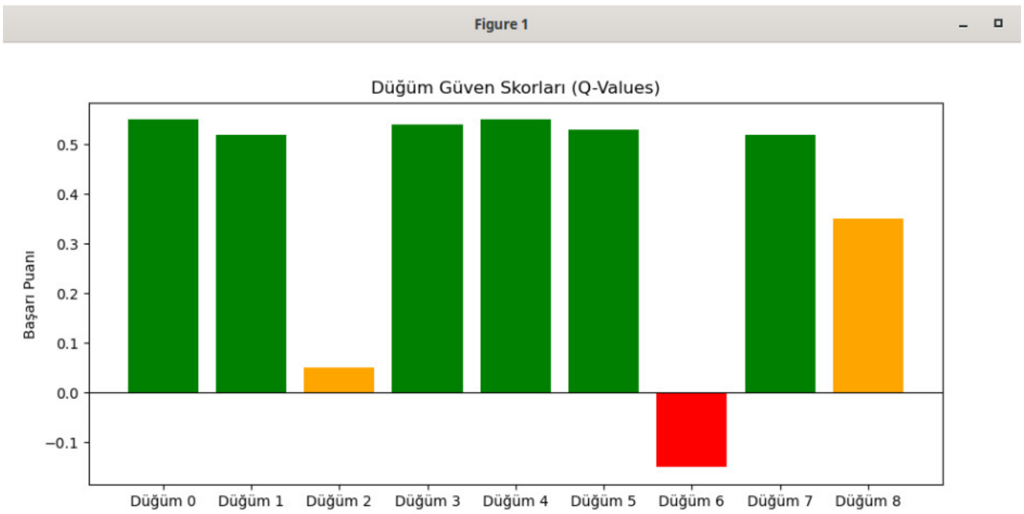
Her iki dosyanın nihai durumdaki kodları bu şekildedir. Son durumda projenin çalışmasına ve rotanın anlık olarak güncellenmesine ilişkin terminal çıktılarına da aşağıdaki görsel üzerinden ulaşılabilir.

Aşağıdaki görselde üst tarafta yer alan terminal Python terminali alt tarafta yer alan terminal ise ns-3 terminalidir. Gelen verilere göre Python terminalinde en iyi düğüm belirlenerek ns-3 terminaline gönderilir sonrasında ns-3 terminali bu bilgiyi kullanarak rotasını anlık olarak günceller.

```
Analiz: Dugum 7 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 0 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 2 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 6 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 8 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 5 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 7 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 4 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 5 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 7 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 4 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 1 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 3 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 5 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 7 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 0 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 2 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 6 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 8 RSSI: -79.486188 | Karar: 3 uzerinden git.
Analiz: Dugum 1 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 3 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 5 RSSI: -74.970738 | Karar: 3 uzerinden git.
Analiz: Dugum 7 RSSI: -74.970738 | Karar: 3 uzerinden git.
```

```
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
--- [AI KARARI] Hedef Dugum: 1 uzerinden rota guncellendi! ---
kagan@debian:~/İndirilenler/ns-allinone-3.39/ns-3.39$
```

Projede yer alan toplam 9 düğüme ilişkin RSSI değerleri dikkate alınarak üretilen güven skorları grafiği ise aşağıdaki gibidir.



Kablosuz Ağlar Projesi Grup 4:

Kağan Emre Meral – Emre Kale – Muhammet Sönmezalp