

**Bursa Teknik Üniversitesi**  
**Bilgisayar Mühendisliği**

**Bilgisayar Ağları Dönem Projesi Rapor**

**2023-2024**

**Yunus Efe YILMAZ**

# İçindekiler

|   |    |
|---|----|
| 1. Özet.....  | 3  |
| 2. NS3 Kurulumu ve 5G-LENA Modülünün Entegrasyonu.....            | 4  |
| 3. Simülasyon Ortamının Hazırlanması .....                        | 6  |
| 4. Hareketlilik Verilerinin NS3 Formatına Çevrilmesi .....        | 8  |
| 5. NS3 Dosyamızı Çalıştırma ve Python ile Ortalama Hesaplama..... | 9  |
| 6. Karşılaştırma ve Sonuç.....                                    | 12 |

# 1.  zet

Ns3 kullanılarak 5G LENA mod l   zerinde SUMO sim lasyomu  zerinden elde ettiĐimiz hareketlilik verileri ile PDR ve gecikme verilerinin  l  m . TrafiĐin ve ara  hızına baĐlı olarak deĐiŐimlerin g zlenip karŐılaŐtırılması.

## 2. NS3 Kurulumu ve 5G-LENA Modülünün Entegrasyonu

NS3 kurmak için linux veya mac ortamına ihtiyacımız vardır. Bu yüzden windows da WSL üzerinden testlerimizi gerçekleştireceğiz.

---

```
$ wget https://www.nsnam.org/releases/ns-allinone-3.41.tar.bz2
$ tar xjf ns-allinone-3.41.tar.bz2
$ cd ns-allinone-3.41/ns-3.41
```

---

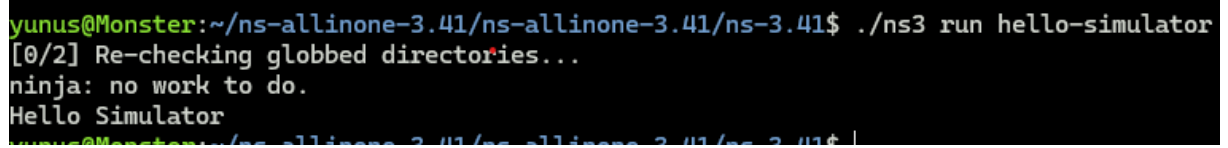
Bu kodları girdikten sonra NS3 ü indirmiş bulunuyoruz. Ama build edip test örneklerini çalıştıralım.

---

```
$ ./ns3 configure --enable-examples --enable-tests
$ ./ns3 build
```

---

Bu aşamayı tamamladıktan sonra örnek bir test çalıştıralım. Sonuç Şekil 1'deki gibi olmalıdır.



```
yunus@Monster:~/ns-allinone-3.41/ns-allinone-3.41/ns-3.41$ ./ns3 run hello-simulator
[0/2] Re-checking globbed directories...
ninja: no work to do.
Hello Simulator
yunus@Monster:~/ns-allinone-3.41/ns-allinone-3.41/ns-3.41$
```

Şekil 1

Şimdi 5G modülünü kuralım. **./ns3-41** dosyasının içindeyken alttaki kodları girelim.

---

```
$ cd contrib
$ git clone https://gitlab.com/cttc-lena/nr.git
$ cd nr
$ git checkout -b 5g-lena-v3.0.y origin/5g-lena-v3.0.y
```

---

5G-LENA modülünü kurmuş olduk şimdi aktif edelim ve test edelim. (Şekil 2)

```
$ ./ns3 configure --enable-examples --enable-tests  
$ ./ns3 build
```

```
yunus@Monster:~/ns-allinone-3.41/ns-allinone-3.41/ns-3.41$ ./ns3 run cttc-nr-demo  
[0/2] Re-checking globbed directories...  
ninja: no work to do.  
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP  
  Tx Packets: 6000  
  Tx Bytes:   768000  
  TxOffered: 10.240000 Mbps  
  Rx Bytes:   767744  
  Throughput: 10.236587 Mbps  
  Mean delay: 0.276044 ms  
  Mean jitter: 0.030032 ms  
  Rx Packets: 5998  
Flow 2 (1.0.0.2:49154 -> 7.0.0.3:1235) proto UDP  
  Tx Packets: 6000  
  Tx Bytes:   7680000  
  TxOffered: 102.400000 Mbps  
  Rx Bytes:   7667200  
  Throughput: 102.229333 Mbps  
  Mean delay: 0.900970 ms  
  Mean jitter: 0.119907 ms  
  Rx Packets: 5990  
  
Mean flow throughput: 56.232960  
Mean flow delay: 0.588507
```

Şekil 2

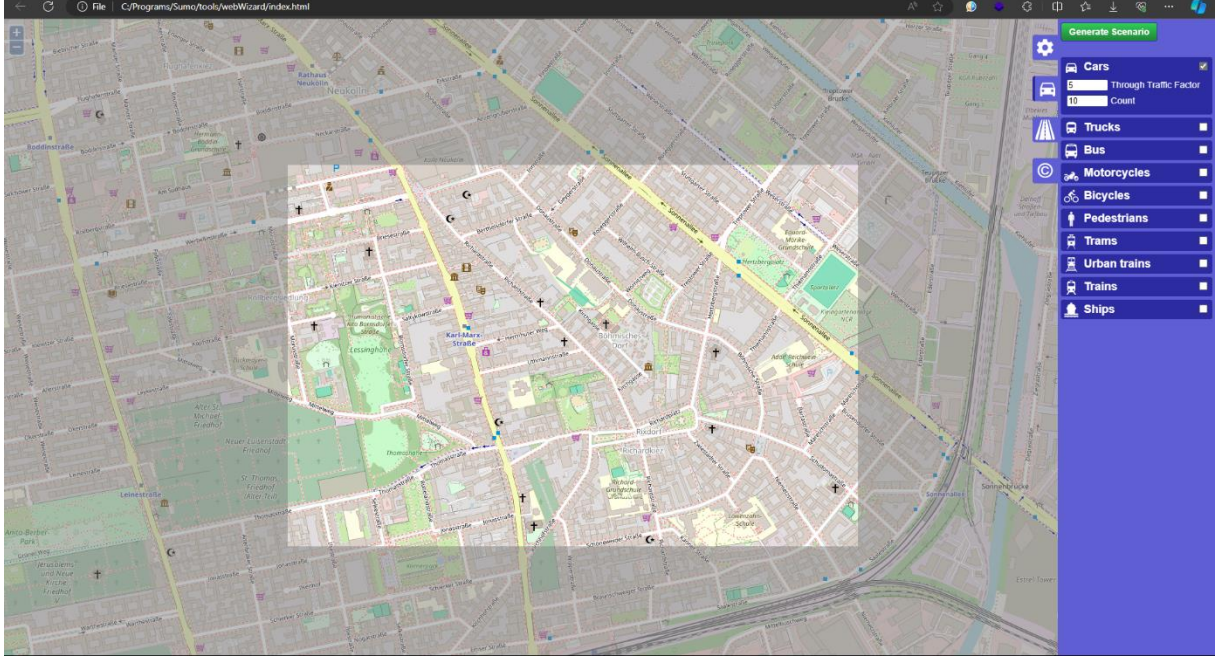
Bütün bu işlemleri yapmadan önce gerekli kütüphaneleri indirmek için bu adresleri kontrol edebilirsiniz.

- NS3: [2. Quick Start — Installation guide \(nsnam.org\)](https://nsnam.org/docs/tutorial/2_quick_start.html)
- 5G-LENA: [5G-LENA: 5G-LENA \(cttc-lena.gitlab.io\)](https://cttc-lena.gitlab.io/)

### 3. Simülasyon Ortamının Hazırlanması

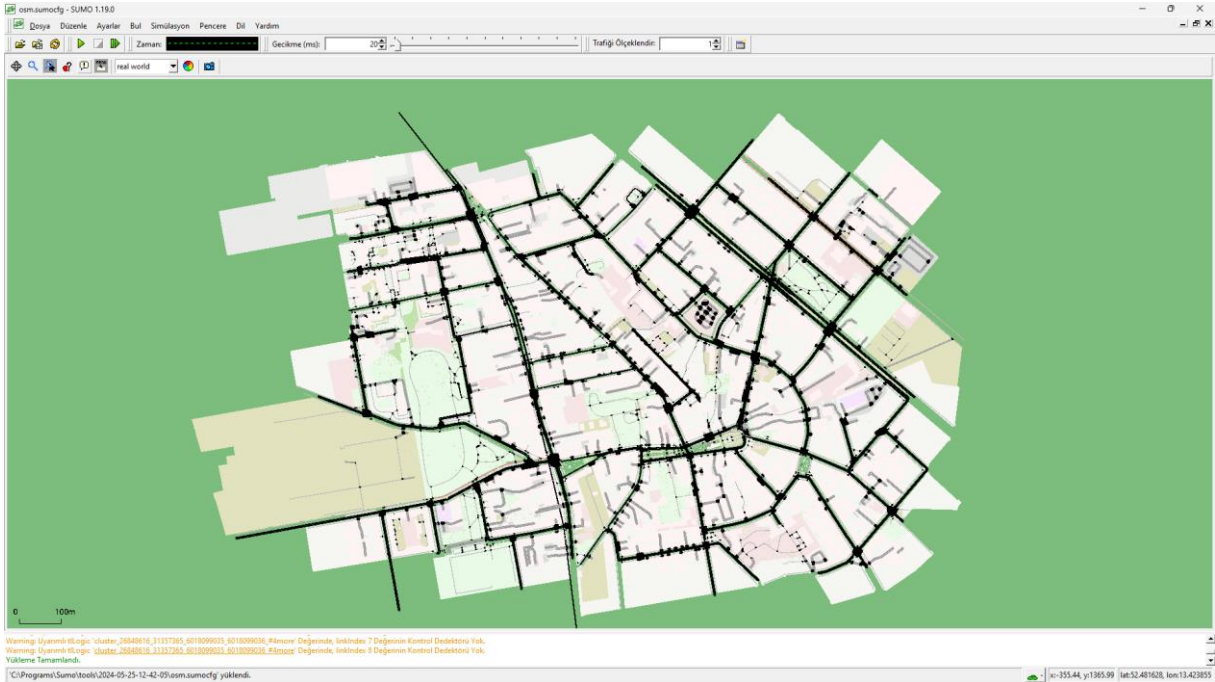
SUMO simülasyonunu [Eclipse SUMO - Simulation of Urban MOBility](http://eclipse.sumo.sourceforge.net/) adresinden indirip bilgisayarımıza kuruyoruz. Kurduktan sonra SUMO klasörü içerinden `\Sumo\tools\osmWebWizard.py` dosyasını çalıştırıp OpenStreetMap'ten (Şekil 3) istediğimiz alanın yol haritasını ve araç sayısını seçip SUMO'ya (Şekil 4) aktarıyoruz.

*OpenStreetMap:*



Şekil 3

**SUMO:**



Şekil 4

SUMO üzerinden simülasyonu çalıştırıp bitmesini bekliyoruz (**Şekil 5**). Bitince tarih ve saat adlarında klasörler oluşuyor ve bu dosyalar içinde hareketlilik verilerimiz bulunuyor şimdi bu verileri NS3 ün anlayacağı formata çevirmemiz lazım onu da bir sonraki başlık altında inceleyeceğiz.

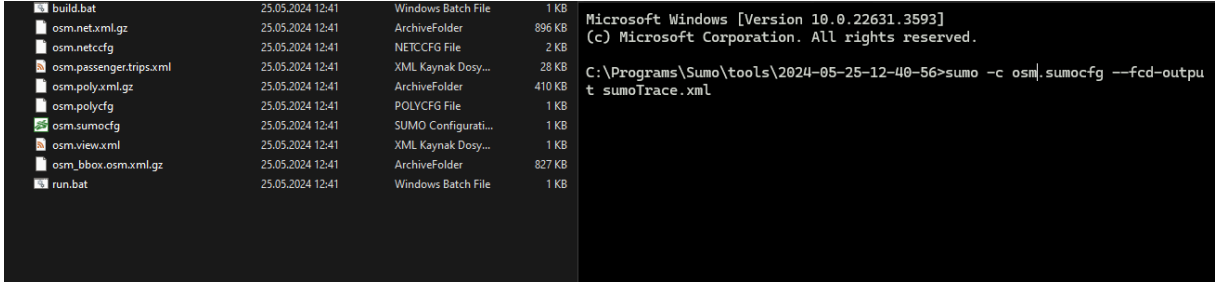
**SUMO:**



*Şekil 5*

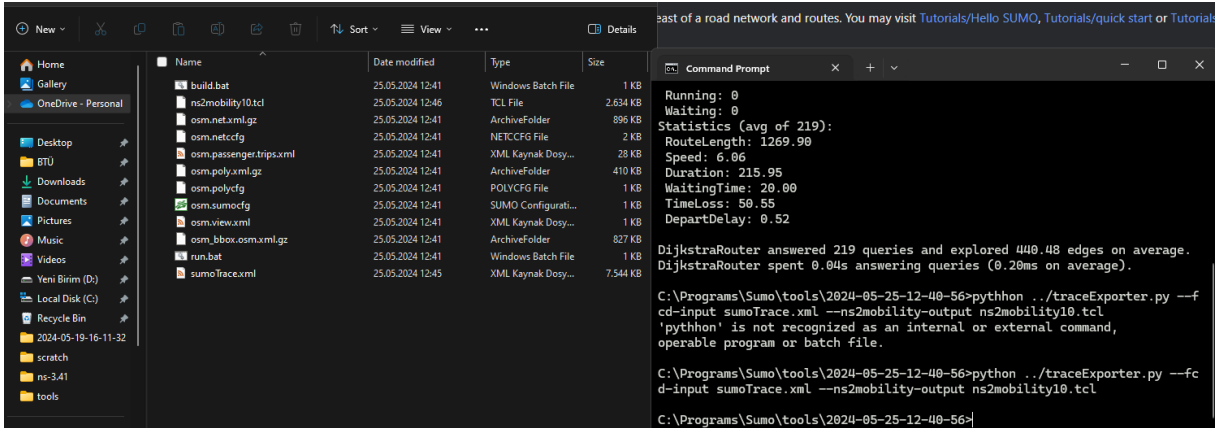
## 4. Hareketlilik Verilerinin NS3 Formatına Çevrilmesi

SUMO simülasyonundan elde ettiğimiz verileri NS3 ün anlayacağı formata çevirmek için alttaki kodu **osm.sumocfg** dosyasına uyguluyoruz. (Şekil 6)



Şekil 6

Elde ettiğimiz **sumoTrace.xml** dosyasını da alttaki kod ile **ns2mobility10.tcl** dosyasına çeviriyoruz. İstersek 100 araçlık bir simülasyon da oluşturabiliriz. (Şekil 7)



Şekil 7

Hareket dosyamızı NS3 ün anlayacağı formata çevirmiş olduk. Şimdi bu dosyayı ns3 klasörü altında **scratch** klasörü altına yolluyoruz.



## 5. NS3 Dosyamızı Çalıştırma ve Python ile Ortalama Hesaplama

Elde ettiğimiz hareketlilik dosyasını ns3 scriptimizin içine aktarıp çalıştırıyoruz. Burada daha verimli ve doğru sonuçlar için 30 defa rastgele çalıştırıp ortalamasını almamıza yardımcı olan başka bir bash script ile NS3 ü çalıştırıyoruz. Bu aşama biraz uzun sürebilir.

Kodlara ulaşmak için bu linki kullanabilirsiniz:

- [yunusefeyilmaz/bilgisayar-aglari-donem-projesi \(github.com\)](https://github.com/yunusefeyilmaz/bilgisayar-aglari-donem-projesi)

Hız 20m/s iken 2 araç, 5 araç ve 10 araç arasındaki araç etkisine bağlı iletim farkına bakalım. Bir de hızın etkisini ölçmek için 5 araçlı bir simülasyonda hızımız 20m/s ve 60m/s iken nasıl bir fark olucak ona bakalım.

Bash scriptimiz Şekil 8'deki gibidir. Otomatik olarak bütün simülasyonları bizim için çalıştıracaktır. Bütün değerler için hepsini çalıştırıp **experiments** klasörü altına değerleri kaydedecektir. (Şekil 10)

```
#!/ns3 run scratch/network-simulation -- --randomStream=2 --ueNumPergNb=10 --ueVelocity=40
script="scratch/network-simulation"

#2 car 20m/s
nDevices=2 # Change from 2 to 10
MaxSpeed=20 # Change from 20 to 80
folder="scratch/experiments/Test_$(MaxSpeed)_$(nDevices)/run"
echo -n "Running experiments: "
for r in `seq 1 30`;
do
    echo -n " $r"
    mkdir -p $folder${r}
    ./ns3 run "$script --randomStream=$r --ueNumPergNb=$nDevices ueVelocity=$MaxSpeed --outp
done

echo " END"

#5 car 20m/s
nDevices=5 # Change from 2 to 10
MaxSpeed=20 # Change from 20 to 80
folder="scratch/experiments/Test_$(MaxSpeed)_$(nDevices)/run"
echo -n "Running experiments: "
for r in `seq 1 30`;
do
    echo -n " $r"
    mkdir -p $folder${r}
    ./ns3 run "$script --randomStream=$r --ueNumPergNb=$nDevices ueVelocity=$MaxSpeed --outp
done

echo " END"

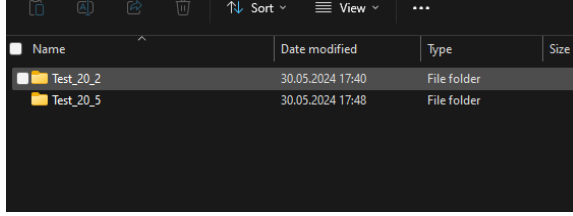
#5 car 60m/s
nDevices=5 # Change from 2 to 10
MaxSpeed=60 # Change from 20 to 80
folder="scratch/experiments/Test_$(MaxSpeed)_$(nDevices)/run"
echo -n "Running experiments: "
for r in `seq 1 30`;
do
    echo -n " $r"
```

Şekil 8

```
yunus@Monster:~/ns-allinone-3.41/ns-allinone-3.41/ns-3.41$ ./runsim.sh
Running experiments: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 END
Running experiments: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 END
Running experiments: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 END
Running experiments: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 END
```

Şekil 9

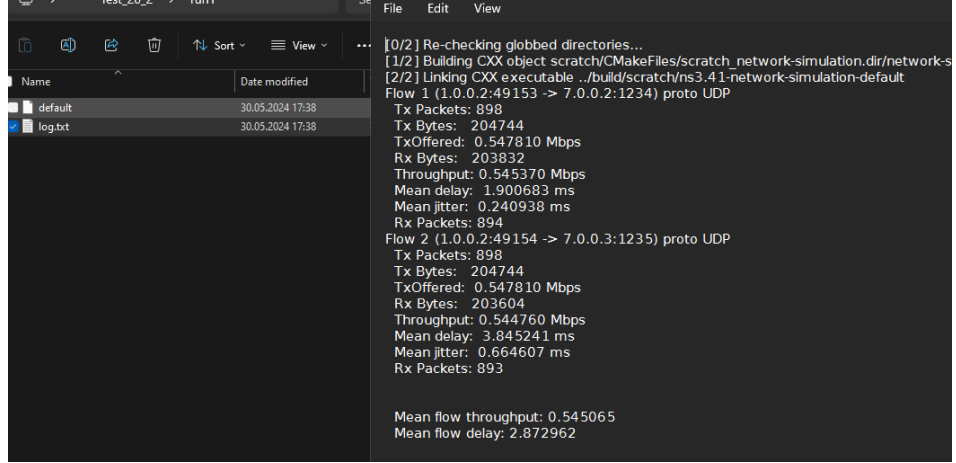
Şekil 9'deki gibi bütün simülasyon bitene kadar beklicez. Bitince Şekil 10 ve Şekil 11'deki gibi klasörler ve klasörler içinde Şekil 12'deki gibi değerler olacak.



Şekil 10

| Name  | Date modified   |
|-------|-----------------|
| run1  | 30.05.2024 17:3 |
| run2  | 30.05.2024 17:3 |
| run3  | 30.05.2024 17:3 |
| run4  | 30.05.2024 17:3 |
| run5  | 30.05.2024 17:3 |
| run6  | 30.05.2024 17:3 |
| run7  | 30.05.2024 17:3 |
| run8  | 30.05.2024 17:3 |
| run9  | 30.05.2024 17:3 |
| run10 | 30.05.2024 17:3 |
| run11 | 30.05.2024 17:3 |
| run12 | 30.05.2024 17:3 |
| run13 | 30.05.2024 17:3 |
| run14 | 30.05.2024 17:3 |
| run15 | 30.05.2024 17:3 |
| run16 | 30.05.2024 17:3 |
| run17 | 30.05.2024 17:4 |
| run18 | 30.05.2024 17:4 |
| run19 | 30.05.2024 17:4 |
| run20 | 30.05.2024 17:4 |
| run21 | 30.05.2024 17:4 |

Şekil 11



Şekil 12

Elde ettiğim farklı hızlardaki ve araç sayılarındaki verileri bir sonraki aşamada karşılaştırıp sonuçları karşılaştıracacağız. Şimdi sonuçları python yardımıyla okuyup ortalamaları hesaplayalım. (Şekil 13)

```
Users > Yunus > Documents > bilgisayar-aglari-donem-projesi > meancalc.py > process_experiment

def process_experiment(experiment_folder):
    if os.path.exists(log_file_path): # log.txt mevcutsa
        with open(log_file_path, 'r') as log_file:
            log_content = log_file.read()
            # Paket sayılarını ve gecikmeyi bul
            tx_packets = sum(map(int, re.findall(r'Tx Packets: (\d+)', log_content)))
            rx_packets = sum(map(int, re.findall(r'Rx Packets: (\d+)', log_content)))
            mean_delay_match = re.search(r' Mean flow delay: \s+([d.]+)', log_content)
            if mean_delay_match:
                mean_delay = float(mean_delay_match.group(1))
                # Toplam değerlere ekle
                total_tx_packets += tx_packets
                total_rx_packets += rx_packets
                total_mean_delay += mean_delay * (rx_packets / tx_packets) # Gecikme
                flow_count += 1

    # Ortalamaları hesapla
    if flow_count > 0:
        avg_tx_packets = total_tx_packets / flow_count
        avg_rx_packets = total_rx_packets / flow_count
        avg_mean_delay = total_mean_delay / flow_count
        print(f"D deney: {experiment_folder}")
        print(f"Ortalama Tx Paket Sayısı: {avg_tx_packets}")
        print(f"Ortalama Rx Paket Sayısı: {avg_rx_packets}")
        print(f"Ortalama Gecikme: {avg_mean_delay} ms")

        # Ulaşılan miktarı bul
        reached_amount = total_rx_packets / total_tx_packets
        print(f"Ulaşılan Miktar: {reached_amount}")

    else:
        print(f"D deney: {experiment_folder}")
        print("Hiç geçerli log dosyası bulunamadı.")

# Deney dosyalarını işle
experiment_folders = ['/Test_20_2/', '/Test_20_5/', '/Test_60_5/', '/Test_20_10/']
for experiment_folder in experiment_folders:
    process_experiment('./experiments' + experiment_folder)
```

Şekil 13

Burada bütün deneylerimizin içine girip her deney için o run1...run30 kadar bütün dosyaların içindeki **log.txt**'yi okutup TX, RX ve gecikme verilerini alıp ortalamasını hesaplıyoruz. Sonuçlar Şekil 14'deki gibi ekrana yazılacaktır. Bu aşamadan sonra diğer başlığa geçip karşılaştırma yapabiliriz.

```
python.exe c:/Users/Yunus/Documents/bilgisayar-aglari-donem-projesi/meancalc.py
Deney: ./experiments/Test_20_2/
Ortalama Tx Paket Sayısı: 1796.0
Ortalama Rx Paket Sayısı: 1787.7333333333333
Ortalama Gecikme: 2.033974884688196 ms
Ulaşılan Miktar: 0.9953971789161099
Deney: ./experiments/Test_20_5/
Ortalama Tx Paket Sayısı: 4490.0
Ortalama Rx Paket Sayısı: 4395.2666666666666
Ortalama Gecikme: 7.31157682043801 ms
Ulaşılan Miktar: 0.9789012620638455
Deney: ./experiments/Test_60_5/
Ortalama Tx Paket Sayısı: 4490.0
Ortalama Rx Paket Sayısı: 4395.2666666666666
Ortalama Gecikme: 7.31157682043801 ms
Ulaşılan Miktar: 0.9789012620638455
Deney: ./experiments/Test_20_10/
Ortalama Tx Paket Sayısı: 8980.0
Ortalama Rx Paket Sayısı: 8584.233333333334
Ortalama Gecikme: 17.25462741520787 ms
Ulaşılan Miktar: 0.9559279881217521
```

Şekil 14

## 6. Karşılaştırma ve Sonuç

Elde ettiğimiz veriler neticesinde hızdaki değişimin **Şekil 14 de** görüldüğü üzere paketlerin ulaşım miktarında değişim olmadığı gözlemlenmiştir. **5** araçlık simülasyonda hız **20m/s** iken **%97** paket ulaşım ve gecikme **7.31ms** iken hız **60m/s** çıkarıldığında **%97** paket ulaşım ve gecikme **7.31ms** olarak hesaplanmıştır.

Araba sayısının artmasıyla beraber **Şekil 14 de** görüldüğü üzere paketlerin ulaşım miktarını düşürdüğünü ve gecikme oranlarının arttığı gözlemlenmiştir. **2, 5 ve 10** araçlık simülasyonları incelersek sırasıyla **%99.5,%97,%95** ve gecikme ise sırasıyla **2ms, 7.3ms, 17.25ms** olarak hesaplanmıştır.

Bütün kodlara ve deney sonuçlarına ulaşmak için bu linki kullanabilirsiniz:

- [yunusefeyilmaz/bilgisayar-aglari-donem-projesi \(github.com\)](https://github.com/yunusefeyilmaz/bilgisayar-aglari-donem-projesi)