MAC Scheduler Performance Evaluation in 5G NR. Problem Description This Python code is intended to evaluate the performance of three MAC schedulers (Round Robin, Proportional Fair, Max Throughput) in a 5G NR network using NS3.42. The simulation involves 1 gNB and 10 UEs, with varying traffic loads (low, medium, high) to assess throughput, latency, and fairness across schedulers. The code should automate the simulation, collect metrics, plot results, and save them to a CSV file. However, it contains multiple errors—syntactical, logical, and NS3-specific—that prevent it from functioning correctly. Your task is to identify and fix these issues."

---

import ns3 import matplotlib.pyplot as plt import pandas as pd import os import random

Configuration

NS3_PATH = "/path/to/ns3.42" os.chdir(NS3_PATH)

Schedulers and traffic loads

schedulers = ["RoundRobin", "ProportionalFair", "MaxThroughput"] traffic_loads = {"low": 1e6, "medium": 5e6, "high": 10e6} # bps results = []

def run_simulation(scheduler, traffic_load):

Set scheduler and random seed

ns3.Config.SetDefault("ns3::NrMacSchedulerNs3::SchedulerType", scheduler) ns3.RngSeedManager.SetSeed(random.randint(1, 1000))

Create nodes

gnbNodes = ns3.NodeContainer() ueNodes = ns3.NodeContainer() gnbNodes.Create(1) ueNodes.Create(10)

Mobility setup
mobility = ns3.MobilityHelper() mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel") mobility.Install(gnbNodes) mobility.Install(ueNodes)

NR network setup

nrHelper = ns3.NrHelper() nrHelper.SetAttribute("Bandwidth", ns3.UintegerValue(20e6)) # 20 MHz gnbDevs = nrHelper.InstallGnbDevice(gnbNodes) ueDevs = nrHelper.InstallUeDevice(ueNodes) nrHelper.AttachToClosestGnb(ueDevs)

Internet stack and traffic

internet = ns3.InternetStackHelper() internet.Install(ueNodes) ipv4 = ns3.Ipv4AddressHelper() ipv4.SetBase("10.1.1.0", "255.255.255.0") interfaces = ipv4.Assign(ueDevs)

UDP traffic

sink = ns3.PacketSinkHelper("ns3::UdpSocketFactory", ns3.InetSocketAddress(ns3.Ipv4Address.GetAny(), 9)) sinkApps = sink.Install(ueNodes) sinkApps.Start(ns3.Seconds(1.0))

source = ns3.OnOffHelper("ns3::UdpSocketFactory", ns3.InetSocketAddress(interfaces.GetAddress(0), 9))

source.SetConstantRate(ns3.DataRate(traffic_load)) sourceApps = source.Install(gnbNodes) sourceApps.Start(ns3.Seconds(1.0)) sourceApps.Stop(ns3.Seconds(10.0))

Flow monitor

flowMonHelper = ns3.FlowMonitorHelper() monitor = flowMonHelper.InstallAll() ns3.Simulator.Run()

Metrics

total_throughput = 0 total_latency = 0 packets_received = [] for flowId, flowStats in monitor.GetFlowStats(): total_throughput += flowStats.rxBytes * 8 / 10 # bps total_latency += flowStats.delaySum.GetSeconds() packets_received.append(flowStats.rxPackets)

throughput = total_throughput / 1e6 # Mbps avg_latency = total_latency / len(packets_received) fairness = 1 - (max(packets_received) - min(packets_received)) / sum(packets_received)

return throughput, avg_latency, fairness

Run simulations

for scheduler in schedulers: for load_name, load_value in traffic_loads: throughput, latency, fairness = run_simulation(scheduler, load_value) results.append({"Scheduler": scheduler, "TrafficLoad": load_name, "Throughput": throughput, "Latency": latency, "Fairness": fairness})

Plotting

fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 8)) for scheduler in schedulers: sched_data = [r for r in results if r["Scheduler"] == scheduler] ax1.plot([r["TrafficLoad"] for r in sched_data], [r["Throughput"] for r in sched_data], label=scheduler) ax2.plot([r["TrafficLoad"] for r in sched_data], [r["Latency"] for r in sched_data], label=scheduler) ax3.plot([r["TrafficLoad"] for r in sched_data], [r["Fairness"] for r in sched_data], label=scheduler)

ax1.set_title("Throughput (Mbps)") ax2.set_title("Latency (s)") ax3.set_title("Fairness Index") plt.legend() plt.savefig("mac_scheduler_results.png") plt.show()

Save to CSV

results_df = pd.DataFrame(results) results_df.to_csv("mac_scheduler_results.csv")

Questions

1. **Syntax Errors: Why will the results.append() line fail, and how should the loop structure be corrected?**

ANS: The results.append() line itself doesn't contain a syntax error, but the loop structure and its context introduce logical and potential runtime issues:

- The traffic_loads variable is a dictionary, but the loop iterates over it incorrectly. The line for load_name, load_value in traffic_loads: should use .items() to properly unpack the dictionary's key-value pairs.
- Without .items(), Python will iterate only over the keys ("low", "medium", "high"), and load_value will be undefined, causing a NameError when passed to run_simulation().

FIX:

```
for scheduler in schedulers:
    for load_name, load_value in traffic_loads.items():  # Add .items()
        throughput, latency, fairness = run_simulation(scheduler, load_value)
        results.append({"Scheduler": scheduler, "TrafficLoad": load_name,
"Throughput": throughput,
                        "Latency": latency, "Fairness": fairness})
```

- ***.items() ensures that both the traffic load name (e.g., "low") and its value (e.g., 1e6) are accessible within the loop.***

2. **NS3 Configuration Errors: The scheduler configuration is incorrect. What's the proper way to set it using ns3.Config.SetDefault?**

ANS:

The line ns3.Config.SetDefault("ns3::NrMacSchedulerNs3::SchedulerType", scheduler) is incorrect because:

- ns3.Config.SetDefault expects a value wrapped in an NS-3 attribute type (e.g., ns3.StringValue), not a raw Python string.
- The attribute "SchedulerType" may not exist directly under ns3::NrMacSchedulerNs3. In NS-3's 5G NR module (nr), the scheduler is typically configured via the NrHelper object, not through a global default attribute like this.

FIX:

The scheduler type should be set using the NrHelper object's SetSchedulerType method, which is the proper way to configure the MAC scheduler in the NR module:

```
def run_simulation(scheduler, traffic_load):
    # Create NrHelper and set scheduler type
    nrHelper = ns3.NrHelper()
```

```
if scheduler == "RoundRobin":
    nrHelper.SetSchedulerType(ns3.NrMacSchedulerNs3.ROUND_ROBIN)
elif scheduler == "ProportionalFair":
    nrHelper.SetSchedulerType(ns3.NrMacSchedulerNs3.PROPORTIONAL_FAIR)
elif scheduler == "MaxThroughput":
    nrHelper.SetSchedulerType(ns3.NrMacSchedulerNs3.MAX_THROUGHPUT)
```

- **Remove the ns3.Config.SetDefault call entirely.**
- **Use the enumerated types provided by ns3.NrMacSchedulerNs3 (e.g., ROUND_ROBIN, PROPORTIONAL_FAIR, MAX_THROUGHPUT) instead of strings.**

3. Simulation Control Errors: Identify at least three issues with simulation timing and state management that could lead to crashes or incorrect results.

ANS:

Missing Simulator Initialization and Cleanup:

- The code calls ns3.Simulator.Run() without initializing the simulation time or stopping it explicitly. NS-3 requires a Simulator.Stop() call to define the simulation duration, and Simulator.Destroy() to clean up resources between runs. Without these, subsequent simulations in the loop may crash due to uncleared states.

No Reset of Random Seed Across Runs:

- ns3.RngSeedManager.SetSeed(random.randint(1, 1000)) sets a random seed, but it's not reset between simulations. This means all runs in the loop might use the same seed, leading to identical results across schedulers and loads, undermining the statistical validity of the comparison.

Overlapping Application Start/Stop Times:

- Both sinkApps and sourceApps start at ns3.Seconds(1.0), and sourceApps stops at ns3.Seconds(10.0), but the simulation duration isn't explicitly capped. If the simulator runs beyond 10 seconds, it may collect irrelevant data, skewing results.

FIX:

```
def run_simulation(scheduler, traffic_load):
    # Reset random seed for each simulation
```

```
ns3.RngSeedManager.SetSeed(random.randint(1, 1000))

# ... (node creation, mobility, NR setup, traffic setup) ...

# Define simulation duration

ns3.Simulator.Stop(ns3.Seconds(10.0))  # Stop at 10 seconds

ns3.Simulator.Run()

ns3.Simulator.Destroy()  # Clean up simulator state
```

- Add Simulator.Stop() to cap the simulation at 10 seconds.
- Call Simulator.Destroy() after Run() to reset the simulator state for the next iteration.
- Ensure the seed is set uniquely for each run.

4. **Network Setup Errors: The AttachToClosestGnb call is incomplete, and traffic setup has an issue. What's wrong, and how should they be fixed?**

   ANS:

   1. Incomplete AttachToClosestGnb:

      a. The line nrHelper.AttachToClosestGnb(ueDevs) is incomplete because it doesn't specify the gNB devices to attach to. The method requires both UE devices and gNB devices as arguments in the NR module.

   2. Traffic Setup Issue:

      a. The source (OnOffHelper) is installed on the gNB (gnbNodes), and the sink is on the UEs (ueNodes), but the destination address interfaces.GetAddress(0) assumes all UEs share the same IP, which is incorrect. Each UE should have a unique IP address, and traffic should target all UEs, not just one.

   FIX:

```
# NR network setup

nrHelper = ns3.NrHelper()

nrHelper.SetAttribute("Bandwidth", ns3.UintegerValue(20e6))  # 20 MHz

gnbDevs = nrHelper.InstallGnbDevice(gnbNodes)

ueDevs = nrHelper.InstallUeDevice(ueNodes)

nrHelper.AttachToClosestGnb(ueDevs, gnbDevs)  # Specify gNB devices
```

```
# Internet stack and traffic
internet = ns3.InternetStackHelper()
internet.Install(gnbNodes)  # Install on gNB too
internet.Install(ueNodes)
ipv4 = ns3.Ipv4AddressHelper()
ipv4.SetBase("10.1.1.0", "255.255.255.0")
interfaces = ipv4.Assign(ueDevs)  # IPs for UEs
gnbInterfaces = ipv4.Assign(gnbDevs)  # IPs for gNB

# UDP traffic to all UEs
sink = ns3.PacketSinkHelper("ns3::UdpSocketFactory",
ns3.InetSocketAddress(ns3.Ipv4Address.GetAny(), 9))
sinkApps = sink.Install(ueNodes)
sinkApps.Start(ns3.Seconds(1.0))
sinkApps.Stop(ns3.Seconds(10.0))

source = ns3.OnOffHelper("ns3::UdpSocketFactory",
ns3.InetSocketAddress(ns3.Ipv4Address("10.1.1.255"), 9))  # Broadcast to UEs
source.SetAttribute("DataRate", ns3.StringValue(str(traffic_load) + "bps"))
sourceApps = source.Install(gnbNodes)
sourceApps.Start(ns3.Seconds(1.0))
sourceApps.Stop(ns3.Seconds(10.0))
```

- ***Add gnbDevs to AttachToClosestGnb.***
- ***Install the internet stack on both gNB and UEs, and assign IPs to both.***
- ***Use a broadcast address (e.g., "10.1.1.255") for the OnOffHelper to send traffic to all UEs, or loop over UE addresses if unicast is intended.***

5. **Metrics Calculation Errors: The throughput, latency, and fairness calculations have flaws. Identify the problems and suggest fixes.**

   ANS:

   - **Throughput Calculation:**
     - total_throughput += flowStats.rxBytes * 8 / 10 assumes a 10-second duration hard-coded into the formula, which is brittle and incorrect if the simulation duration changes. It should use the actual simulation time.
   - **Latency Calculation:**
     - avg_latency = total_latency / len(packets_received) divides by the number of flows, not the total number of packets. This skews the average latency if flows have different packet counts. It should use the total rxPackets across all flows.
   - **Fairness Calculation:**
     - fairness = 1 - (max(packets_received) - min(packets_received)) / sum(packets_received) is a simplistic metric and may not align with standard fairness indices (e.g., Jain's Fairness Index). It also assumes packets_received is non-empty; otherwise, it will crash.

   FIX:

```
simulation_time = 9.0  # Seconds (10 - 1, since traffic starts at 1s)

total_throughput = 0

total_latency = 0

total_packets = 0

packets_received = []

monitor.CheckForLostPackets()  # Ensure stats are complete

for flowId, flowStats in monitor.GetFlowStats():

  total_throughput += (flowStats.rxBytes * 8) / simulation_time  # bps

  total_latency += flowStats.delaySum.GetSeconds()

  total_packets += flowStats.rxPackets

  packets_received.append(flowStats.rxPackets)

throughput = total_throughput / 1e6  # Mbps

avg_latency = total_latency / total_packets if total_packets > 0 else 0  # Per-packet latency

fairness = sum([x**2 for x in packets_received]) / (len(packets_received) * sum([x for x in packets_received])**2) if packets_received else 0  # Jain's Fairness Index

return throughput, avg_latency, fairness
```

     - **Compute throughput using the actual simulation duration (9.0 seconds of active traffic).**

- **Calculate average latency over total packets received, with a safeguard for zero packets.**
- **Use Jain's Fairness Index for a standard fairness metric, with a check for empty lists.**

6. **Plotting Errors: Why will the plot legend fail, and how should it be corrected?**

   ANS:

   The legend fails because plt.legend() is called after plotting all subplots, but the label argument is set only for the individual plot calls within each subplot (ax1.plot, ax2.plot, ax3.plot). Matplotlib requires legend() to be called on each subplot axis (ax1, ax2, ax3) separately to display legends correctly for each.

   FIX:

```
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 8))

for scheduler in schedulers:

  sched_data = [r for r in results if r["Scheduler"] == scheduler]

  ax1.plot([r["TrafficLoad"] for r in sched_data], [r["Throughput"] for r in sched_data], label=scheduler)

  ax2.plot([r["TrafficLoad"] for r in sched_data], [r["Latency"] for r in sched_data], label=scheduler)

  ax3.plot([r["TrafficLoad"] for r in sched_data], [r["Fairness"] for r in sched_data], label=scheduler)


ax1.set_title("Throughput (Mbps)")

ax1.legend()

ax2.set_title("Latency (s)")

ax2.legend()

ax3.set_title("Fairness Index")

ax3.legend()

plt.tight_layout()

plt.savefig("mac_scheduler_results.png")

plt.show()
```

   - **Call legend() on each axis (ax1.legend(), ax2.legend(), ax3.legend()) to ensure legends appear in all subplots.**
   - **Add plt.tight_layout() to prevent overlap**.