Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. 49 / Prog No. 10

PROBLEM STATEMENT: NS2 Simulatore

ALGORITHM & CODE:

of networks, including Wired, Wireless and mobile networks.

Introduction to NS2

1. Purpose and Use:

NS2 is designed to simulate and model network Protocols such as TCP, UDP, routing Protocols, Congestion control algorithms and more.

2. Simulation Types

- Wired Networks
- Wireless Networks
- Routing and Congestion Control.

3. Key feature

- Discrete Event Simulation
- Support for multiple Protocols
- Trace files
- Animation

4. Architecture:

- Simulator Core
- Tcl (Tool Command Language)
- C++ / OTcl Interface
- NAM (Network Animator)

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty
Date :

Signature of Student
Date :

	Subject :	Software :
		Hardware :
	Branch : / Semester :	Page No. 50 / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

5. Application

- Network Protocol Research
- Network Design and Optimization
- Educational Tool

6. Limitation

- Steep Learning Curve
- Outdated

7. NS2 vs NS3 :

NS3 is an evolution of NS2 and it is considered more modern and better suited for simulating current networking technologies.

NS2 is an effective and powerful tool for simulating and analyzing networks, helping researches and engineers evaluate the Performance of network protocols, routing algorithms, and more under various conditions.

Key parameters

1. Packet arrival rate (λ):
2. Transmission Capacity (C):
3. Buffer Size (B)
4. Simulation Time (T)
5. Packet Size :

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. 51 / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

Simulation Step:

1. Generate random Packet arrivals based on the Packets arrival rate.
2. If the buffer has space, store the incoming Packet.
3. If the buffer is full, drop the incoming Packet.
4. Track the number of ~~dpp~~ dropped Packets.

A Python code that Simulates Packet dropping based on above assumption.

```
import random
import numpy as np
def simulate_packet_drop(arrival_rate,
transmission_capacity, buffer_size, simulation_time):
    # initialize variables
    buffer = 0
    dropped_packet = 0
    total_packets = 0

    time_step = 1  # 1 second per time step

for t in range (0, simulation_time, time_step):
    Packet_arrivals = np.random.Poisson(arrival_rate)
    total_Packets += Packet_arrivals
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

LEM STATEMENT

ORITHM & CODE :

```
for _ in range (packet_arrivals):
    if buffer < buffer_size :
        buffer += 1
    else :
        dropped_packets += 1
packets_transmitted = min (buffer, transmission_capacity)
buffer -= packets_transmitted
return dropped_packets, total_packets

arrival_rate = 5
transmission_capacity = 3
buffer_size = 10
simulation_time = 100

dropped, total = simulate_packet_drop (
    arrival_rate, transmission_capacity, buffer_size,
simulation_time)
print (f" total Packets Dropped : {dropped}")
print (f" Total Packets Arrived : {total}")
print (f" Total Drop ratio : {dropped / total : 2f}")
```

Output

```
Total Packets Dropped : 234
Total Packets Arrived : 482
```

INPUT GIVEN

OUTPUT OBTAINED — Packet Drop ratio : 0.49

REMARKS

GRADE :

Signature of Faculty
Date :

Signature of Student
Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.

BLEM STATEMENT

GORITHM & CODE :

To Simulate the number of Packets dropped in a TCP / UDP Network environment, we need to model the transmission of Packets, loss events, and underlying protocol behaviour.

1. UDP Packet Dropping Simulation

UDP is a connectionless protocol and doesn't include built in reliability mechanisms. There UDP Packets dropping is mainly due to network congestion, buffer overflows, or transmission errors.

2. TCP Packet Dropping Simulation

TCP, on the other hand, is a reliable, connection oriented protocol. When Packet loss is detected TCP will Triggers retransmission.

Simulation Setup:

We'll simulate a network where Packets are transmitted and occasionally dropped due to network conditions. The Packet loss rate for each protocol (UDP or TCP) can be influenced by several factors, like network congestion, timeout, or buffer overflow. For simplicity we can model the

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

Key Parameters for the Simulation :

1. Number of Packets :
2. Packet loss Rate (UDP)
3. Timeout Threshold (TCP)
4. Retransmission mechanism (TCP)
5. Congestion level

Simulation steps

1. UDP Simulation
2. TCP Simulation

Python Code for Simulation.

```
import random
num_Packets = 1000
udp_loss_rate = 0.1
tcp_loss_rate = 0.05
tcp_retransmission_limit = 3

def simulate_udp_Packet_drops(num_Packets,
                              loss_rate);
    dropped_Packets = 0
    for _ in range (num_Packets):
        if random.random() < loss_rate:
            dropped_Packets += 1
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty	Signature of Student
Date :	Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
    return dropped_packets
def simulate_tcp_packets_drops(num_packets,
    lossrate, max_retransmissions):
    dropped_packets = 0
    retransmitted_packets = 0
    for _ in range(num_packets):
        attempts = 0
        while attempts < max_transmission:
            if random.random() < loss_rate:
                dropped_packets += 1
                attempts += 1
            else:
                break
        if attempts == max_retransmission:
            dropped_packets += 1
    return dropped_packets
udp_drops = simulate_udp_packet_drops
    (num_packets, udp_loss_rate)
print(f" UDP Packet Drops : {udp_drops}/{num_
packets} ({udp_drops/num_packets)*100}%loss)")
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty	Signature of Student
Date :	Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
tcp_drops = simulate_tcp_packet_drops
(num_packets, tcp_loss_rate, tcp_retransmission_limit)
Print (f" TCP Packet Drops : {tcp_drops}/{num_packets}
({(tcp_drops/num_packets)*100}% loss including
retransmission)")
```

Output :

UDP Packet Drops : 100/1000 (10.0% loss)
TCP Packet Drops : 45/1000 (4.5% loss including retransmission.

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :
	Signature of Student Date :