Final Report for VANET Transport Layer Protocol Comparative Analysis Project

1- Installation guide

**1. 1. NS-3 Installation:**

Install NS-3 Dependencies:

Update your package list and install necessary dependencies:Update your package list and install necessary dependencies:

sudo apt-get update

To download these files, type in the termianal window:

sudo apt-get install g++ python3 cmake git

Clone NS-3 from the Official Repository:Clone NS-3 from the Official Repository:

To get the information download the newest version of NS-3:

git clone https://gitlab. com/nsnam/ns-3-dev. git

cd ns-3-dev

Build NS-3:

Configure and build NS-3:

. /waf configure --enable-examples

. /waf build

Add Your Simulation Code:

Copy your simulation code into the scratch folder (e. g. , vanet\_simulation. cc):Copy your simulation code into the scratch folder (e. g. , vanet\_simulation. cc):

cp vanet\_simulation. cc ns-3-dev/scratch/

Run the Simulation:

To run the simulation:

. /waf --run scratch/vanet\_simulation

**1. 2. SUMO and MOVE Installation:**

For realistic generation of vehicular movement there exists SUMO (Simulation of Urban Mobility).

Install SUMO:

On Ubuntu, you can install SUMO with:On Ubuntu, you can install SUMO with:

sudo apt-get install sumo sumo-tools sumo-gui

Generate Mobility Models in SUMO:Generate Mobility Models in SUMO:

By invoking SUMO’s GUI, a realistic road network can be developed under which the movements of vehicles can be well-simulated.

Mobility traces export in NS-2 format for the use in NS-3:

sumo -c sumo\_config\_file. sumocfg —fcd-output mobility\_trace. xml

MOVE (Mobility Model Generator):

Using MOVE, one can derive the road networks that are compliant with the SUMO data format and the movement patterns of the vehicles as well.

MOVE produces SUMO trace files so that they can be incorporated into NS-3.

2. Code Walkthrough

In this section, each part of the code is described and how it works is described as well. There are a number of improvements for each part of the code that reflect missing features as stated in the PPT.

2. 1. Transport Layer Protocols (TCP, UDP, SCTP, DCCP):Transport Layer Protocols (TCP, UDP, SCTP, DCCP):

Implementation: In the experiments it recreates the TCP, UDP, SCTP, and DCCP transport layer communication protocols and measures their performance.

Location in Code: Each of the protocols has two applications implemented – OnOff and PacketSink for both sending and receiving data.

// TCP application setup

OnOffHelper tcpAppV2V ("ns3::TcpSocketFactory", InetSocketAddress (vehicleInterfaces.GetAddress (1), port));

// UDP application setup

OnOffHelper udpAppV2V ("ns3::UdpSocketFactory", InetSocketAddress (vehicleInterfaces.GetAddress (3), port));

// SCTP application setup

OnOffHelper sctpAppV2V ("ns3::SctpSocketFactory", InetSocketAddress (vehicleInterfaces.GetAddress (5), port));

// DCCP application setup

OnOffHelper dccpAppV2V ("ns3::DccpSocketFactory", InetSocketAddress (vehicleInterfaces.GetAddress (7), port));

2. 2. Types of VANETs (V2V, V2I, V2X):Types of VANETs (V2V, V2I, V2X):

Vehicle-to-Vehicle (V2V): Full mobility vehicles can communicate directly with the other without having to rely on the infrastructure of roads or network of signals that we give them.

Vehicle-to-Infrastructure (V2I): It is V2I where vehicles interact with other static RSS or Road Side units.

Vehicle-to-Everything (V2X): A mix of V2V and V2I; in which vehicle not only communicate with other vehicles but also with RSU.

Location in Code:

V2V communication between vehicles:

wifi.Install (wifiPhy, wifiMac, vehicles); // V2V communication

V2I communication between vehicles:

wifi.Install (wifiPhy, wifiMac, roadsideUnits); // V2I communication

2. 3. SUMO and MOVE Integration:

Integration: Such realistic mobility patterns are generated by SUMO and MOVE. These mobility traces are then imported into NS-3 through a tool called as Ns2MobilityHelper.

Location in Code:

The Ns2MobilityHelper loads mobility traces generated by SUMO or MOVE:The Ns2MobilityHelper loads mobility traces generated by SUMO or MOVE:

Ns2MobilityHelper ns2mobility ("mobility\_trace.xml");

ns2mobility.Install();

2. 4. Performance Metrics:

Throughput: Measures the data rate the average depending on the speed of data transmission in Mbps.

Packet Delivery Ratio (PDR): That is the ratio of packets delivered without being lost to the number of packets transmitted by the network.

Routing Overhead: Shows the ratio between the number of routing packets and data packets which is sent through the network.

End-to-End Delay: It means that it is the total time taken for a packet to travel from source to the destination.

Location in Code:

void PrintMetrics (Ptr<FlowMonitor> monitor, FlowMonitorHelper &flowmonHelper, std::string protocol, double speed, uint32\_t numVehicles)

These metrics are calculated and printed in the function:These metrics are calculated and printed in the function:

3. Console Output Explanation

The console output includes throughputs of every protocol (TCP, UDP, SCTP, DCCP ) varying speeds 50 km/h, 100 km/h, 150 km/h for correspondingly 10, 20, 30 vehicles.

Sample Console Output:

Metrics Explained:

Throughput: The rate of communication effectiveness of the amount of information being passed over the network.

PDR: It is evidenced by the fact that those with higher values tend to perform and are more reliable in their operations.

Routing Overhead: Below shows the number of control packets which is inversely proportional to the test video size.

End-to-End Delay: They mean that packets have been delivered at a faster rate or, in other words, the corresponding values are lower.

The console output shows these statistics with respect to each speed and the number of vehicles respectively.

4. XML Output Explanation

The other output is in XML format and files are named as vanet\_flowmon. xml which contains specific statistics details of every flow. These are the parameters that are transmitted and received packets, packets that were lost, delay, jitter and through put.

Sample XML Output:

Metrics Explained:

txPackets/rxPackets: Packets sentences transmitted/received contains the number of packets sent/received during the experiments.

delaySum: Total value of the elapsed time for the packets.

jitterSum: The comparison of the variation in packet delay.

throughput: In Mbps, depicting the amount of data that have been transferred successfully.

These metrics can be identified and processed in the XML file with the help of the script written in the Python language.

5. Missing Features Addressed

**Justification of Comparative Study:**

This is due to the fact that the performance of the different protocols; TCP, UDP, SCTP, and DCCP, changes with motion in high-mobility VANET scenarios. TCP is reliable than UDP since it has a lot of overhead checking whereas UDP is faster in data transfer. SCTP supports multi-streaming While DCCP is responsible for congestion control for the applications that are real-time.

**Research Methodology Alignment:**

The simulation methodology is underpinned directly in the set objectives of the study. The methodology of comparing the four transport protocols at different speeds and vehicle counts simulates realistic scenarios and provides the assessment of the protocols’ behavior in dynamic vehicular environments.

**High Mobility and Dynamic Topology:High Mobility and Dynamic Topology:**

High mobility of up to 150km/h and dynamic topology is incorporated in the simulation through using realistic mobility traces from SUMO. This mimics actual mobile ad-hoc networks where nodes are in constant motion and hence affects theprobability and availability of sending messages between nodes.

**IEEE 802. 11p Standard:**

The simulation is built with the 802. 11p standard also known as WAVE which is specifically intended for vehicle to vehicle communication. This prevents distortion of comparative results in the context of protocols for VANET environments.

**Novelty:**

It is important to say that, comparing with the existing studies on the four transport protocols mentioned, this paper presents an analysis of four transport protocols in a not yet explored scenario of high-mobility VANET with an emphasized focus on protocol 802. 11p. Incorporating all the three types of communication, namely V2V, V2I and V2X in one simulation makes the study even more unique.

**Validation of Results:**

The results that are obtained can be used to compare with the previous works that have been done in the same field. It is important to note that alternative parameters like throughout, delay and packet delivery ratios which have been discussed before can also be checked with some papers or other research papers already published. This aids to check whether the simulation results are real and authentic from an analyst’s perspective.