Clustering in VANETs

Software: NetSim Standard v13.3 (64 bit), Visual Studio 2022, MATLAB R2016 or higher

Project Download Link:

https://github.com/NetSim-TETCOS/Clustering_in_VANETs_v13.3/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and set up the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

Clustering in VANETs

Clustering in Vehicular Ad-Hoc Networks (VANETs) is a fundamental technique that involves grouping vehicles into clusters to enhance network efficiency, resource utilization, and communication management. In the context of VANETs, where vehicles are highly mobile and dynamically changing their positions, clustering becomes a critical strategy to organize the network and facilitate various applications.

Dynamic Clustering in VANETs

In the context of mobile vehicular networks, clustering cannot be static due to the dynamic nature of vehicle movements. Vehicles constantly change their positions and connections, requiring clusters to adapt accordingly. Dynamic clustering involves the formation, dissolution, and reformation of clusters based on real-time information about vehicle positions and network conditions.

Dynamic clustering addresses challenges such as frequent network topology changes, varying vehicle densities, and the need for efficient data dissemination. It ensures that clusters remain relevant and effective despite the constant mobility of vehicles.

Clustering in NetSim with MATLAB interfacing

Dynamic Clustering in VANETs is implemented through NetSim's interface with MATLAB. MATLAB receives vehicle and RSU coordinates as input data.

Cluster heads, represented by RSUs, are identified in the network. For instance, if the network configuration requires three clusters, three RSUs are strategically placed in the network.

Vehicles become members of clusters led by RSUs. The clustering process involves assessing the distance between each vehicle and the available RSUs. The vehicle associates itself with the closest RSU, which becomes the cluster head of its cluster.

These steps occur periodically and can be tailored as needed. The cluster members, i.e., vehicles, are decided based on their current positions, making the process adaptable. However, cluster heads (RSUs) remain fixed since there is no mobility involved in their placement.

This implementation enhances network efficiency by dynamically forming clusters and optimizing communication between vehicles and RSUs.

The codes required for the mathematical calculations done in MATLAB are written to a clustering.m file and this file is available in the MATLAB folder under bin_x64 of clustering_in_VANET_Workspace_v13.3.

Implementation and code modifications

The modifications are done to DSR project, where we are calling MATLAB with the inputs required (vehicle and RSU coordinates, number of vehicles and RSUs).

And in return we'll get the cluster and the vehicles in a particular cluster.

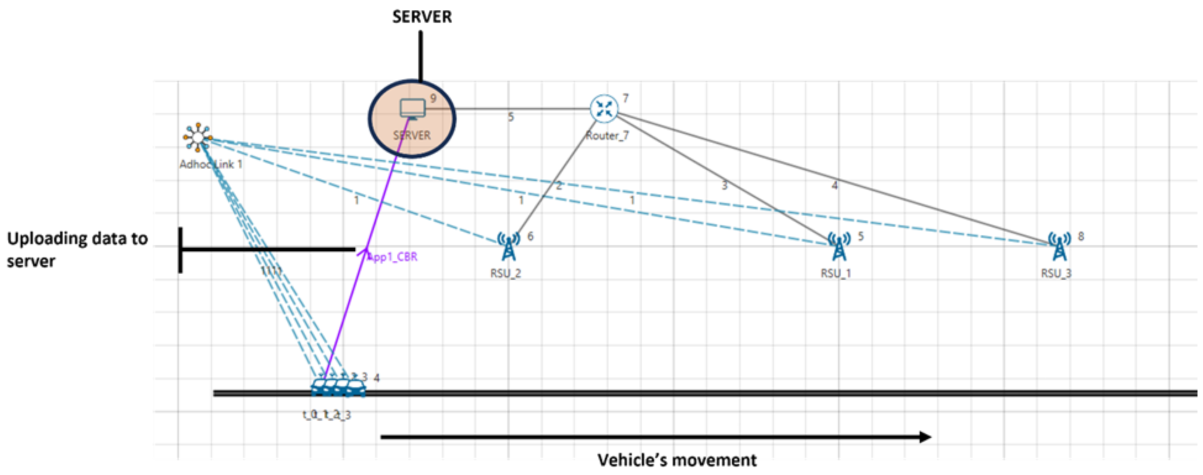
Once the cluster is formed, we are modifying the NetSim code to route the data traffic from the source (vehicles) to the cluster Heads (RSUs) and to destination from CHs

A Clustering.c file is added to the DSR project which contains the following functions:

- **fn_NetSim_dynamic_clustering_CheckDestination()** //This function is used to determine whether the current device is the destination.
- **fn_NetSim_dynamic_clustering_GetNextHop()** //This function statically defines the routes within the cluster and from the cluster to the sink node. It returns to the next hop based on the static routing that is defined.
- **fn_NetSim_dynamic_clustering_IdentifyCluster()** //This function returns the cluster id of the cluster to which a sensor belongs.
- **fn_NetSim_dynamic_clustering_run()** //This function makes a call to MATLAB interfacing function and passes the inputs from NetSim (i.e.) the num of vehicles and RSUs along with its coordinates.
- **fn_netsim_dynamic_form_clusters()**//This function assigns each vehicle to its respective clusters based on the cluster IDs obtained from MATLAB.
- **fn_netsim_assign_cluster_heads()**//This function assigns the cluster heads for each cluster based on the cluster head IDs obtained from MATLAB.
- **fn_NetSim_Dynamic_Clustering_Init()**//This function initializes all parameter values.

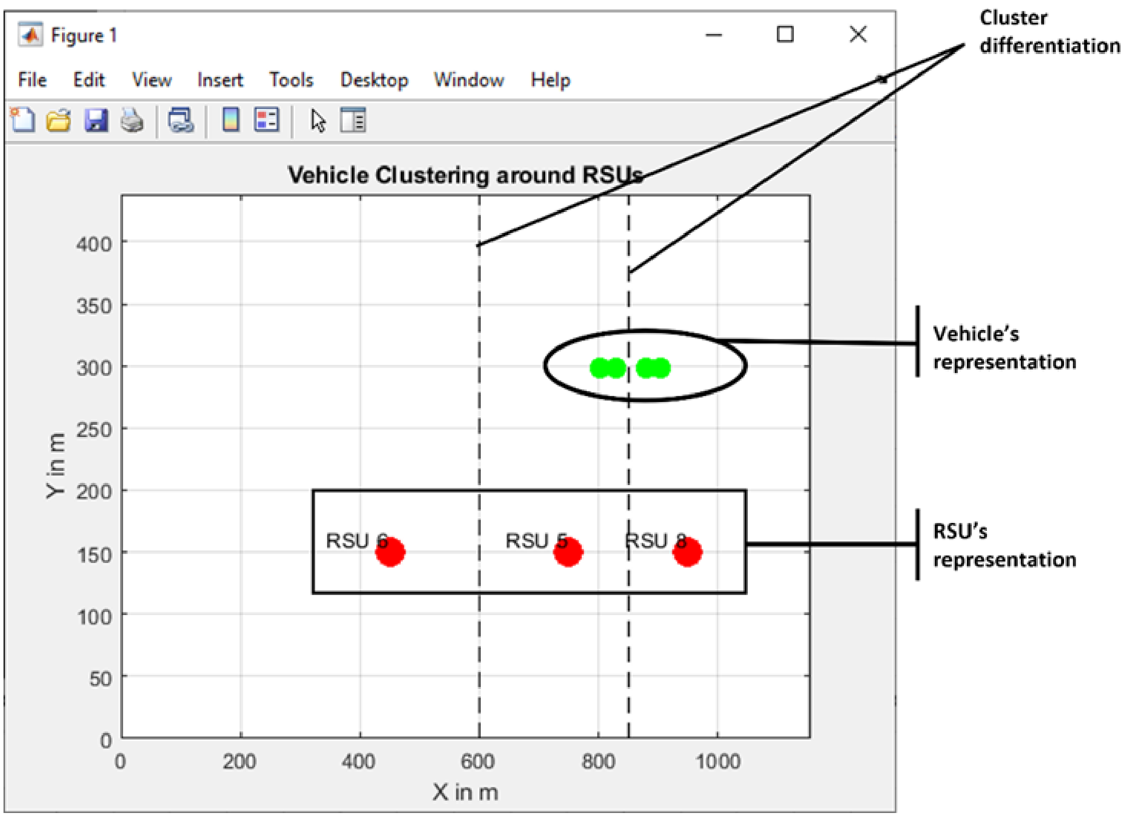
Example

1. Run NetSim in administrative mode.
2. Dynamic_Clustering_Workspace_v13.2 comes with a sample network configuration that is already saved. To open this example, go to Your work on the home screen of NetSim and click on the clustering_example_1 from the list of experiments.
3. The saved network scenario consists of 4 vehicles along with 3 RSUs. Traffic is configured from vehicle1 to SERVER.

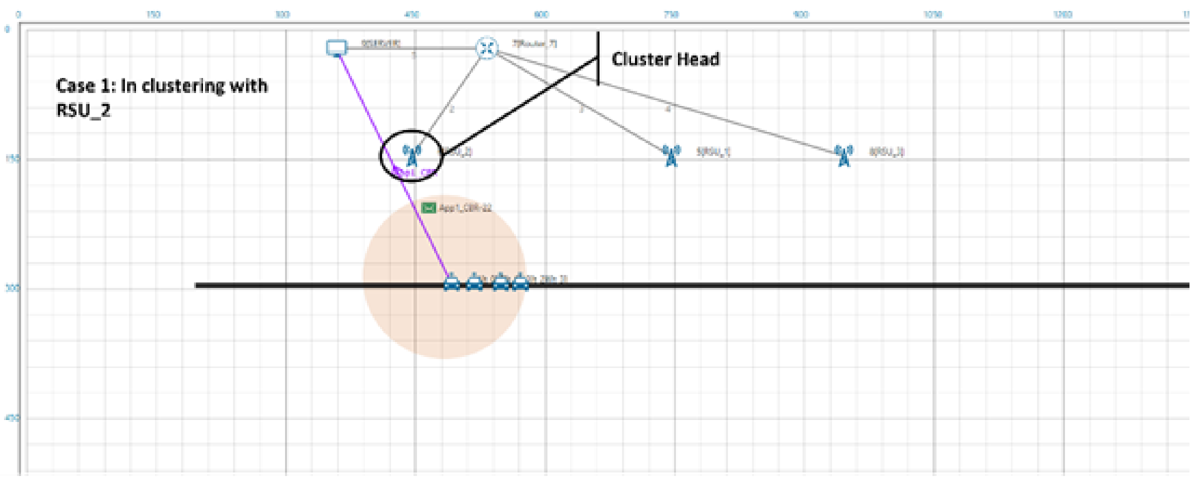


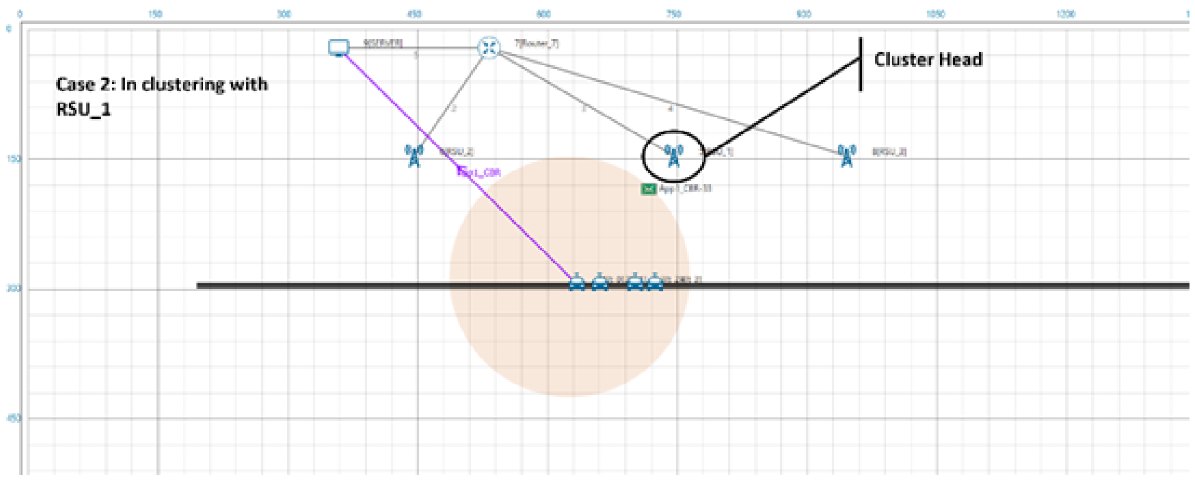
4. Run the simulation and press any key to continue. NetSim simulation console will show the following message in the console “Waiting for NetSim MATLAB Interface to connect...”. NetSim will automatically open the MatlabInterface.exe console window.
5. It will open the MatlabInterface.exe console window. You will observe that as the simulation starts in NetSim, along with SUMO. MATLAB gets initialized and the graph that showcases the clustering is plotted during runtime.

Results and Discussion



In this scenario, vehicles keep moving and switch clusters whenever they get closer to a different RSU. This dynamic process is shown visually in the animation window. Vehicles adjust their clusters based on their distance from nearby RSUs as they move through the network.





The impact of changing clusters on data transmission can be observed through the packet trace.

This MATLAB-based clustering approach for VANETs is applicable to scenarios involving mobile vehicles. However, there are certain limitations:

- Vehicles must exhibit mobility.
- RSUs are essential and serve as cluster heads (CH) in the network.
- The number of clusters corresponds to the number of RSUs in the network.

Note: Include RSU's in the VANET network that you create, for running simulations with this modified workspace.