**The Computer Communication Lab (236340)**

**Winter 2009-2010**

OLSRv2 protocol simulation and analysis

### Final Report

# Submitted by:

###### Assaf Israel

###### Eli Nazarov

###### Asi Bross

### Table of contents:

[1 Introduction 3](#_Toc244784421)

[1.1 OLSR introduction 3](#_Toc244784422)

[1.2 Neighborhood Discovery Protocol (NHDP) 3](#_Toc244784423)

[1.3 Optimized Link State Routing Ver. 2 (OLSRv2) 4](#_Toc244784424)

[2 Project Description 6](#_Toc244784425)

[2.1 Purpose 6](#_Toc244784426)

[2.2 Main Goals 6](#_Toc244784427)

[2.2.1 Utilization Analysis 6](#_Toc244784428)

[2.2.2 Network Reliability 6](#_Toc244784429)

[2.2.3 Topology 6](#_Toc244784430)

[2.2.4 Scalability 6](#_Toc244784431)

[2.3 Stretched Goals 7](#_Toc244784432)

[2.3.1 Topology mobility 7](#_Toc244784433)

[2.3.2 Secondary MPR sets 7](#_Toc244784434)

[2.4 Output 7](#_Toc244784435)

[3 General Layout 8](#_Toc244784436)

[4 Main Modules 8](#_Toc244784437)

[4.1 GUI 9](#_Toc244784438)

[4.2 Log 9](#_Toc244784439)

[4.3 Tasks Queue 10](#_Toc244784440)

[4.4 Event Generator 10](#_Toc244784441)

[4.5 Dispatcher 10](#_Toc244784442)

[4.6 Topology Manager 11](#_Toc244784443)

[4.7 OLSRv2 Layer 11](#_Toc244784444)

[5 Main Algorithm 12](#_Toc244784445)

[5.1 Event Generation 12](#_Toc244784446)

[5.2 Protocol implementation 12](#_Toc244784447)

[5.3 Dispatcher 14](#_Toc244784448)

[5.4 Topology Manager 14](#_Toc244784449)

[6 Appendixes 16](#_Toc244784450)

[6.1 Appendix 1: Estimated project timetable 16](#_Toc244784451)

[6.2 Appendix 2: Requirements and Assumptions 16](#_Toc244784452)

[6.2.1 Assumptions 16](#_Toc244784453)

[6.2.2 Requirements 16](#_Toc244784454)

# Introduction

## OLSR introduction

The Optimized Link State Routing Protocol (OLSR) is developed for mobile ad hoc networks. It operates as a proactive protocol (table driven) which exchanges topology information with other nodes of the network regularly. It was developed to work independently from other protocols. Likewise it makes no assumptions about the underlying link-layer. The protocol inherits the stability of a link state algorithm and has the advantage of having routes immediately available when needed due to its proactive nature. OLSR stack location is beneath the transport layer as supplement to the network layer. Each node can operate as a router to different implementations of local networks.

OLSR is well suited to larger and denser mobile networks then other naive routing options. The larger and more dense a network, the more optimization can be achieved. OLSR uses hop-by-hop routing, i.e., each node uses its local information to route packets.

OLSR is designed to work in a completely distributed manner and does not depend on any central entity. The protocol does not require reliable transmission of control messages since each node sends control messages periodically, and can therefore sustain a reasonable loss of some messages.

## Neighborhood Discovery Protocol (NHDP)

The NHDP objective is to maintain a 1-hop and 2-hop neighbors set for each node in a Mobile Ad-hoc Network (MANET). This enables the node, to use this information later, when implementing the OLSR protocol. NHDP is a table driven protocol, meaning maintenance messages are used in order to maintain the nodes databases.

NHDP uses only one type of messages, called *"Hello" messages*. These messages are sent between each node and every node that is in its broadcast radius (1-hop neighbors), but not beyond. Meaning, these messages are not forwarded by any receiver. Their role is passing information about a node's 1-hop neighbors to its 1-hop neighbors. A node, which receives a message, can add the sender to its 1-hop neighbors set. If the receiver is stated in the senders 1-hop neighbors set, he can state that the link is bi-directional (i.e. messages are send and received by both parties). The receiver construct a 2-hop neighbors set by joining all the received sender's 1-hop neighbors sets and subtracting its own 1-hop neighbors set. Meaning, node C is a 2-hop neighbor of node A, if it's a 1-hop neighbor of node B that is a 1-hop neighbor of node A, and C is not a 1-hop neighbor of node A.

Figure ‎1‑1

*"Hello" messages* are sent by every participating node, and by every node's *network interface*, in a manner that will extend the network connectivity as much as possible. E.g. – node C can be a 2-hop neighbor of node A, via node B but not share the same *network interface* with node B that B shares with A. This behavior allows the protocol to build the most comprehensive and flexible 1-hop & 2-hop topology map, so future *data messages* could "switch" *network interfaces* in order to achieve the best and fastest possible connectivity to their destination.

*"Hello" messages* are sent periodically according to an adjustable *Refresh Interval*. This means that if a message from a 1-hop neighbor is not received by a node, then this connection is considered lost, and the neighbor is removed from the 1-hop neighbors set. At the next *"Hello" message* that will be sent from the nodes that noticed their neighbor's disappearance, the change will be noted so other neighbors could be aware of the change as quickly as possible.

A node does not have to wait until the next scheduled *"Hello" message* transmission, in order to alert the change (new/lost/changed status link). He can issue a dynamically scheduled *"Hello" message* after a certain *Minimum Interval* from the last message transmission.

## Optimized Link State Routing Ver. 2 (OLSRv2)

Optimized Link State Routing Ver.2 (*OLSRv2*) is a proactive protocol (table driven) that uses NHDP protocol described in section 1.2. *OLSRv2* is optimized routing protocol for MANET networks that can handle dynamic topology.

The main concept of the protocol is the use of Multipoint Relays (MPRs). MPRs are used in the OLSR protocol to minimize the overhead of flooding messages in the network by reducing redundant retransmissions in the same region. Each node in the network selects a set of nodes in its symmetric 1-hop neighborhood (see ‎1.2) that may retransmit its messages. This set of selected neighbor nodes is called the "Multipoint Relay" (MPR) set of that node. The neighbors of node X which are not in its MPR set, receive and process broadcast messages but do not retransmit broadcast messages received from node X. In route calculation, the MPRs are used to form the route from a given node to any destination in the network.

The set of MPRs should cover all the 2-hop (see ‎1.2) neighbors of the node, meaning that the union of all neighbors of MPRs gives the group of all nodes until the 2-hop degree of the MPRs selector.

Basically, the route between two nodes in the network is a sequence of hops through MPRs. The last MPR is the target node or, the target is a 1-hop neighbor of the last MPR.

A node’s request from another to be an MPR for it, is passed via attaching new information to *“Hello” messages* generated by the underlying *NHDP*.

*OLSRv2* defines only one type of message, *Topology Control (TC).*

*“Hello” messages* are being modified by *OLSRv2* to contain the selected MPRs set of the source node. Upon receiving *“Hello” messages*, if the target node was selected as MPR, it will update its status to be an MPR for the source node. *“Hello” messages* are never forwarded by the *OLSRv2* protocol.

*“TC” messages* include the set of all nodes that selected the source node as a MPR. Also, this message may contain relevant information regarding the different interfaces of the node and the network (if such exist) that are connected to this node. *“TC” messages* are always forwarded, unless this message was already sent by this station, and are broadcasted to all nodes in the network.

*OLSRv2* keeps different information tables, Topology Information Base, that hold information regarding the MPR selectors of this node, network topology information and information regarding routes to all nodes in the network.

Furthermore, *OLSRv2* keeps information regarding all *“TC” messages* that this station receives processes or forwards – Processing and Forwarding Information Base. This information helps to lower the amount of *“TC” messages* that are being sent by each node.

*“TC” messages* are being generated and transmitted periodically by the node once in a specific time, or due to some changes in the topology of the node. Topology change is basically a change in contents of Topology Information Base.

When a packet needs to be transmitted from node A to node B, node A will send it to the first MPR in the current route known to node A. Each MPR will forward the packet to the next MPR in the route.

Figure ‎1‑2

# Project Description

## Purpose

The main purpose of this project was to create a simulator for the *OLSRv2* protocol. Investigate and analyze the behavior and performance of the protocol by setting different network, protocol and layout characteristics. The simulator was created with a wide range of adjustable parameters that can produce a wide spectrum of analytic measurements.

## Main Goals

The simulator that we created can produce a wide range of analytic measurement; hence we decided to focus our research on the following topics.

### MPR Selection Analysis

MPR selection can have a significant impact on the network utilization, a minimal selection of 1-hop MPRs that cover the entire 2-hop neighborhood of a single node may result in a better network utilization in comparison to a selection of the entire 1-hop neighborhood as MPRs.

#### Minimal 1-Hop Set

#### All 1-Hop Set

#### Results

### Data Send Mode Analysis

We decided to compare between the propagation of data packets through the MPRs vs. the propagation of data packets through the entire 1-hop neighborhood.

#### MPRs only

#### All neighbors

#### Results

### Topology Analysis

The simulator we created supports setting the topology layout in one of two modes, either UNIFORM or CLUSTER. We decided to investigate the throughput received in each of these topology layouts.

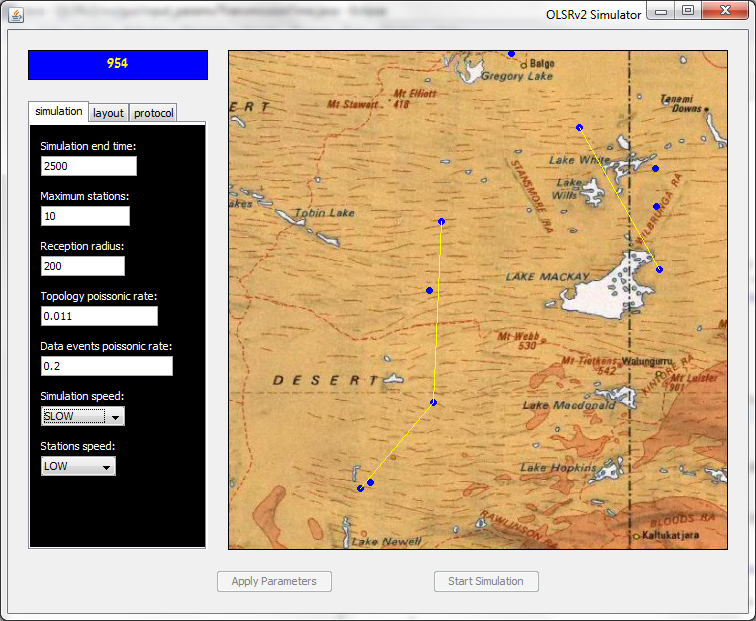
#### Uniform

#### Cluster

## Output

### GUI

The output generated by the GUI shows the state of the stations and the data packets transmissions while they occur, the size attributes of the simulated world can be altered through the "World size" entry in the Layout tab. The speed in which the graphical simulation is presented can be changed during the simulation run by altering the "Simulation speed" entry in the Simulation tab.



### SQL

The simulator we created simulates the OLSRv2 protocol under various conditions, in order to analyze it's output we use a logger module (see 4.3) that logs all of the different events (see 2.3.2.2) while the simulation is running, when the simulation ends, all of the logged events are recorded into a MySQL database (in addition to an excel spread sheet). Since all of the events that occurred during the run are saved into the SQL database when the simulation ends, different queries can be performed on the database table, and analyzed data can be pulled out.

We used a MySQL eclipse plug-in in order to create and modify the SQL table, an SQL proxy can be configured to save the data generated on a local host or on a designated remote host.

#### SQL Table Structure

The SQL table consists of the following columns.

**INDX** - A progressive index of the table rows.

**VIRTUAL\_TIME** - The virtual time the event occurred.

**NODE\_ID** – A UUID that represents the station that caused the event to be logged.

**EVENT\_TYPE** - The type of the event (see 2.3.2.2).

**GLOBAL\_SOURCE** - The initiator source of the event (i.e. the source of the Data Message).

**LOCAL\_SOURCE** - The UUID of the station from which the message was received.

**LOCAL\_TARGET** - The UUID of the station that received the message.

**GLOBAL\_TARGET** - The final recipient of the event (i.e. the target of the Data Message).

**X\_COOR** – The x coordinate of the station in the plane.

**Y\_COOR** - The y coordinate of the station in the plane.

**RADIUS** - The reception/transmission radius of the station.

**LOST** – A Boolean indicator that states if the event was lost (i.e. in message events).

**ERROR** - Boolean flag stating if some error occurred.

**DETAILS** - More details regarding this event.

#### Event Types

The followings are the main types of events that can be logged into the database under the EVENT\_TYPE column.

**DATA\_SENT\_EVENT** – Processing the data message.

**BUSSY\_MSG\_IGNORED** – The station is busy, thus ignored the received message.

**DATA\_SENT\_FROM\_SOURCE** – Message was sent from the source.

**DATA\_REACHED\_2\_TARGET** – Message reached its destination.

**DATA\_NOT\_SENT\_NO\_ROUTE** – Message was not sent because the source couldn't find a route.

**DATA\_LOSS\_NO\_ROUTE** – Relay station couldn't find a route, thus the message was lost.

**DATA\_IGNORED\_AT\_RELAY** – Station received a message that was not addressed to her.

**DATA\_SENT\_FROM\_RELAY** – Relay station sent the message to the next station in the route.

**DATA\_REACHED\_2\_RELAY** – Relay station on the route received data message.

**LOCAL\_TARGET\_NOT\_PHYSIBLE** – There was a message sent to a station that doesn’t exist in its 1-hop neighborhood (the station didn’t learn yet that the station is no longer there).

**TTL\_LIMIT\_REACHED** – TTL of the data message is over (the message is dropped).

# Topology Layout

The *OLSRv2* illustrates a MANET network of nodes that moves dynamically. The simulator simulates both high and low dense areas. We have implemented variations for the topology of the stations. Uniform topology is a topology where the stations are spread uniformly over the plane. The Cluster topology is when the stations are spread over the plane in groups. Using this topology we can check when the messages are routed through only few nodes in order to reach stations in other clusters (“bottle-neck” stations).

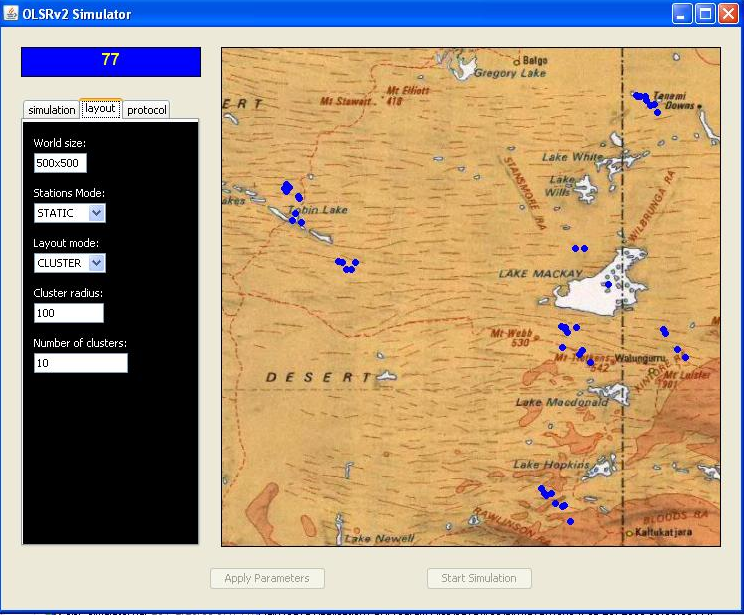
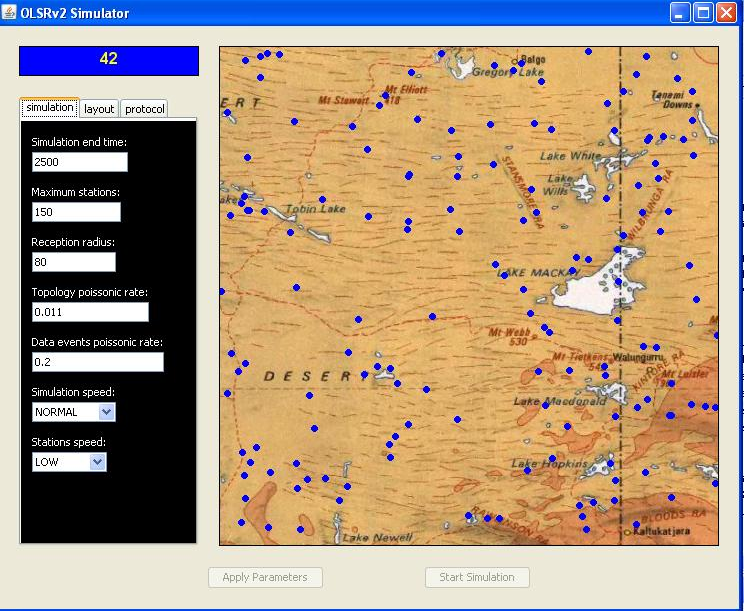


Figure 3-2: Cluster Distribition

Figure 3-1: Uniform formation

# Implementation

## Class Diagram



## GUI

The GUI allows the user to set different specification for the simulator execution it enables the user to see the current state of the MANET by painting a graphical presentation of the nodes and the transmission of data messages.

The GUI is designed using swing (a graphical library in java). The world state is dynamically updated using a dedicated thread the pulls the topology events from a dedicated queue and updates the graphical state of the world, the events are inserted into the dedicated queue by the dispatcher module during the simulation execution. The implementation of the GUI maintains its own simulation ticks, hence makes it possible to the user to set the simulation speed.

### Input Parameters

#### Simulation parameters

* Simulation end time – The amount of logical time units the simulation execution should run.
* Maximum stations – The maximum number of stations that can exist simultaneously. In static mode this represent the number of stations (No topology changes occur), in dynamic mode it represent the initial and maximum number of stations.
* Reception radius – Represents the radius of reception of each station.
* Topology poissonic rate – The expected number of topology events that can occur in a single time unit.
* Data events poissonic rate - The expected number of data messages events that can occur in a single time unit.
* Simulation speed – The current speed of the simulation graphic display, it sets the rate at which a single simulation tick occurs, this parameter can be changed during the simulation run, since this only effect the GUI representation it does not have any effect on the simulation results. The values for this parameter are SLOW, NORMAL, FAST, REAL TIME – the "REAL TIME" value displays the current simulation state while the others may have a certain delay in the graphic presentation.
* Stations speed – In dynamic mode it set the amount of time the station moves between two points.

#### Layout parameters

* World size – Sets the size of the area in which stations can exist, the input value is in the format of <width>x<height>, in the GUI representation it is changed dynamically and covers a pixel area in the given size, the boundaries are between 100x100 and 500x500.
* Stations mode – Set the nature of the stations, STATIC value means that all stations are created in the beginning of the simulations and they don’t change their position, DYNAMIC value means that stations are constantly moving in space and MIXED means that Stations will decide to move every once in a while (in a non-deterministic fashion), it also means that new stations can be created and existing stations can be destroyed.
* Layout mode – Set the topology layout of the stations, the values are UNIFORM and CLUSTER, CLUSTER means that the stations are organized in groups.
* Cluster radius – The radius of a single cluster, all the stations are inside this radius. Applicable only if layout mode is CLUSTER.
* Number of clusters – The total number of clusters in the topology. Applicable only if layout mode is CLUSTER.

#### Protocol parameters

* Entry validity period – The amount of unit time a table entry is valid in the station's control tables.
* Hello message interval – The amount of unit time between consecutive hello message transmissions of the same station.
* TC message interval - The amount of unit time between consecutive TC message transmissions of the same station.
* Transmission time – The amount of unit time it takes for a transmission of a packet.
* Protocol mode – The mode in which control packets are being transferred, NORMAL mean through all 1-hop neighbors, ALL MPRs mean through all the MPRs of a station.
* Protocol data send mode – The mode in which data packets are being transferred, NORMAL means through any 1-hop neighbor, MPRs mean through MPRs only.

### World Topology

The stations location are represented as circles on a <width>x<height> panel that represents the world in which the stations reside. The stations distribution can be defined through the Layout mode entry which enables a uniform or cluster distribution. Data packets that are sent between stations are represented as a line that connects the two participating stations and is visible for a period of "Transmission time" as specified by the user.

The graphical updated state of the stations are handled using a dedicated thread that maintains a correct presentation of the simulation current state. The dedicated thread keeps the state updated by pulling events from a dedicated queue, the events are analyzed and the topology changes occur, in addition the queue holds events of data packets that are being sent by the stations, the dedicated thread analyzes these events and alerts the GUI to draw a line between the two participating stations, this line represents the transmission of a data packet. Control messages (HELLO/TC) are not represented in the GUI as packets being, though they are.

## Log

This module is responsible for all the data handling of the system. It acts as a hub for different components to log their data in.

The log module saves the different events into an excel spread sheet while the simulation is running, and once the simulation is over it dumps all the gathered data into a predefined SQL server, hence in the end of the simulation run, the user receives both an excel sheet and an SQL database that hold all of the events that occurred during the simulation.

The Log module is implemented as a singleton; hence it can be accessed easily from all the participating components. The Log module uses an SQL Writer module that encapsulates all the interactions with the SQL server. The Log module saves all the data into the SQL database in the end of the simulation run; hence keeps the simulation execution free from any database write delays.

## Event Generator [Assaf]

The *Event Generator* is responsible for the creation of two types of events:

* *Topology events* – Creation/Migration/Destruction of stations.
* *Data messages* – These events simulate data, that’s being received by the OLSR Layer at a certain node, and is designated to another node in the network.

In order to safely create these events, the Event Generator needs to hold a set of the nodes labels and coordination. It will query the set prior to any event generated in order to maintain consistency.

For example, upon creating a new station the *Event Generator* should check that a station does not already exists, with the same parameters (id, location). Another example is when creating *Data event*, the *Event Generator* should check if the source or target stations exists.

For more information about the specific algorithms that are used to handle these events, see section ‎5.1.

## Dispatcher [Assaf]

The *Dispatcher*’s name basically reveals its role. It’s designed to retrieve the top task from the *Tasks Queue* (The next task in a chronological order), inspect it, and redirect it to the relevant objects which in turn process it.

The *Dispatcher* specific behavior depends on the type of task it has retrieved from the *Tasks Queue*:

* *Topology events* – are passed directly to the *Topology Manager*, who in turn updates its records based on the information given.
* *Data messages* – are passed to the *OLSRv2 Layers* which identifies as the source nodes of the messages.
* *“Hello”* or *“TC” messages* – are passed to all of the *OLSRv2 Layers* that represents nodes which are within the *Reception Radius* of the originator node location. The list of nodes which answer this criterion is given to the *Dispatcher* by requesting the *Topology Manager* which has the knowledge regarding the nodes physical whereabouts in space.

Whenever the *Dispatcher* notices that an event destination is missing (e.g. *Data message* is sent to a non-existing node) it will log this event at the *Log* and will discard the event.

## Topology Manager

The *Topology Manager*'s job is to maintain information about the simulated physical attributes of stations. The combined data of all the stations attributes defines the network physical picture at a certain time. The attributes that are stored per station in the *Topology Manager* are:

* *Coordinates* in space – the simulated physical location at the defined 2D space. These Coordinates can be changed by the *Topology Manager* upon receiving a *Topology event* from the *Dispatcher* which instructs it to migrate a certain node to a new location (considering that the new location is still within the defined space boundaries)
* *Reception Radius –* each node might have a different reception radius, in this version of the simulator the option isn't given to the user although it is well supported, thus all stations uses the default assigned reception radius.

The topology manager supplies a wide range of options regarding the current topology state, it allows:

1. Creating a new station.
2. Destroying an existing station.
3. Changing the position of a given station.
4. Receiving a station's full 1-hop neighborhood. (inside its reception radius)
5. Checking if a station exists, either by its location or by its ID.

## Protocol [ELI]

This module implements the *OLSRv2* logic of a single node it a MANET environment. This module is split into two Three-layers:

* *Protocol Layer*
* *NHDP Layer*
* *OLSR Layer*

### Protocol Layer

This is a logical layer that makes the connection between the different parts of the simulator and the protocol layers. Due to the fact that the simulator is event driven all messages are being sent as events to the dispatcher and all the intervals (Hello/TC) are measured by special events, thus when the dispatcher processes an event it invokes an appropriate method in the protocol layer of the station. This method translates the event received from the dispatcher to an appropriate message of the protocol or an event related to the work flow, e.g. Hello/TC interval ended, and passes it for process in the protocol layers. Also this layer is responsible for generating, processing and routing Data messages according to different topology data sets that are maintained by the OLSRv2 and NHDP layers.

### NHDP Layer

Implements the NHDP layer. NHDP layer is responsible for processing and generating *Hello messages*. Upon receiving *Hello messages* this layer updates the different data sets it maintains and generates *Hello messages* if any change happened in the 1-hop topology of the station.

### OLSRv2 Layer

Implements the *OLSRv2* logic. This layer is responsible for adding additional information to the generated *Hello messages* (MPR information) as detailed in the *OLSR* description (Section ‎1.3), as well as generating and processing *“TC” messages* and *Data messages*. Upon receiving a *TC message* the layer updates different data sets that hold information regarding the global topology of the network, re-calculates MPRs for this station and calculates routes to all the stations it learned.

# Appendixes

## Requirements and Assumptions

### Assumptions

* Low level implementation details as described in **draft- ietf-manet-olsrv2-10** and in **draft-ietf-manet-nhdp-10** like packet formation or jitter aspects are not in the scope of this simulation. Requirements
* The simulator is implemented in Java, so JVM is needed for execution.
* MySQL should be installed in order to get SQL support.

## Future Work