**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 is to create a simulator for *OLSRv2* (and NHDP) protocol. Investigate and analyze the behavior and performance of the protocol with different characteristics of the network.

## Main Goals

### MPR Selection Analysis

#### Minimal 1-Hop Set

#### All 1-Hop Set

### Data Send Mode Analysis

#### MPRs only

#### All neighbors

### Topology Analysis

#### Uniform

#### Cluster

## Output

### GUI

### SQL

#### Data Types (all the columns of the sql table)

#### Event Types (types of events that are written to the database)

# Topology Layout

The *OLSRv2* will illustrate a MANET network of nodes that move dynamically. The simulator will simulate both high and low dense areas. There will be no specific router that will route the messages but each node will calculate the routes by itself. Furthermore, all nodes can be spread uniformly over the area as well as in other formations e.g. several groups.

Figure 3-1: Uniform formation

Figure 3-2: Cluster Distribition

Figure 3-1: Group formation

Figure 3-2: Uniform Distribition

# Implementation

## Class Diagram



## GUI

The *GUI* will allow the user to enter different specification for the simulator and view a graphical presentation of the MANET and the nodes.

Map of nodes will be presented as circles (or dotes).

The user will be able to choose from a pre defined set of initial formations, the initial node distribution in the area. This will be the initial configuration for the *Topology Manager* and *Event Generator*.

Furthermore, the user will be able to view different measurements and analysis reports by clicking on appropriate buttons.

The *GUI* model will communicate with the *Log* model in order to get different information to display reports and graphs and with *Topology Manager* in order to display the MANET.

## 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.

Since data from all parts of the system concentrate here, the *Log* is the idle place to process the data and produce more meaningful numbers, like the different throughput calculations, and data messages failure rate.

Beside the regular data reports the *Log* will also hold Error reports, that contain information about system health, and also reports about data messages that have failed to reach their destination (points to protocol failure).

The main type of data the *Log* will collect is information about different messages that will spark from the different *OLSRv2 Layers* (Nodes). Each control message (*"Hello"* or *"TC"* message) as well as each data message that is being sent (passed to the *Tasks Queue*) will be logged for later processing.

## Event Generator

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

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* Job is to maintain information about the simulated physical attributes of nodes. The combined data of all the nodes attributes defines the network physical picture at a certain time. The attributes that will be stored per node within 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, simulating changes that might be induced because of different platforms or terrain conditions that real world station are expose to.

The *Topology Manager* may be asked by the *Dispatcher* to supply a list of nodes which *Coordinates* are within a certain node’s *Reception Radius*. That way the *Dispatcher* can decide which *OLSRv2 Layers* needs to be on the receiver end of a certain *“Hello”* or *“TC” message* task.

## 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.

### NHDP Layer

Implements the NHDP logic. Is responsible for some of the *“Hello” messages* tasks processing, and the invocation of all them.

### OLSRv2 Layer

Implements the *OLSRv2* logic. Is responsible for adding additional information to the *“Hello” messages* (such as MPR information) as detailed in the *OLSR* description (Section ‎1.3), as well as generating and processing *“TC” messages* and *Data messages*.

Passing messages to nodes in the *OLSRv2 Layer* node *Reception Radius* is done, by pushing the relevant events to the *Tasks Queue* with information about the message source. The *Dispatcher* in its turn will redirect that message to the appropriate *OLSRv2 Layers*.

Messages events which are produces as a byproduct of an earlier event will be tagged with a timestamp of the previous event’s timestamp in addition to a small delta which represent the network’s propagation delay as well as the nodes computation time.

*OLSRv2 Layer* objects are created/destroyed by the *Dispatcher* upon receiving a *Topology event* which corresponds to that action.

The *OLSRv2 Layer* alerts the *Log* about new messages that will be pushed to the *Tasks Queue* for later throughput calculations, as well as errors that might occur because of a protocol failure. E.g. when a *Data message* task is received which the destination is unknown to the *OLSRv2 Layer* (may have not been updated yet, with a new node’s appearance).

### OLSRv2 Protocol

# 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. The project team will try it best to accommodate the protocols attributes and description as explained in these documents, however modification may be necessary because of time and complexity constrains.
* The project team will produce more/different features/data as asked by the Project Guidance crew.
* As mention in the *Project Description* section (‎2), eventually some goals may not be implemented because of time issue constrains. The Project team will, of course, try it best to accommodate all of the described features and Goals.

### Requirements

* The simulator will be implemented in Java, so JVM is needed for execution.

## Future Work