

BOTSWANA INTERNATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY



Faculty of Sciences

Department of Computer Science and Information Systems

A Multi-Greedy Forwarding Approach in Geographic Packet Routing

by

Gbadebo Oladeji-Atanda

(ID: 17100097)

BSc Computer Science & Geography (Unam), PGDE (Unam),
MSc Computer Science (UB), BSc Hons in Information Systems (Unisa)

A Thesis Submitted to the Faculty of Sciences in Partial Fulfilment of the Requirements for the
Award of the Degree of Doctor of Philosophy in Computer Science and Software Engineering of
BIUST

Supervisor: Dimane Mpoeleng PhD
Department of Computer Science and Information Systems
Faculty of Science, BIUST
mpoelengd@biust.ac.bw

July 2023

Table of Contents

Table of Contents	ii
List of Appendices	v
List of Figures	vi
List of Tables	viii
List of Equations	ix
List of Abbreviations	xi
Definition of Terms.....	xii
Declaration Regarding the Work and Copyright	xiv
Acknowledgements.....	xv
Dedication.....	xvi
Abstract.....	xvii
CHAPTER 1	1
1 INTRODUCTION.....	2
1.1 Context and topology instability in highly dynamic MANET environments	3
1.2 Geographic progress and greedy forwarding.....	4
1.3 Problem statement	7
1.4 The rationale for multi-greedy packet forwarding.....	8
1.5 Research objective	10
1.6 Research question	11
1.7 Evaluation method	11
1.8 Contribution to knowledge.....	12
1.9 Summary.....	13
CHAPTER 2	15
2 LITERATURE REVIEW.....	16
2.1 MANET routing environments and hybrid context-aware routing protocols	16
2.1.1 The MANET zoning and hierarchical hybrid routing protocols	17
2.1.2 Context-aware routing improvement methods and frameworks	18
2.1.3 The DTN routing in partitioned or intermittently connected environments.....	21
2.1.4 Summary.....	23

2.2	The geographic routing method.....	24
2.2.1	Position-based routing	24
2.2.2	The geographic greedy forwarding procedure.....	25
2.2.3	Classic geographic greedy forwarding techniques.....	27
2.2.4	Greedy failure recovery methods	32
2.2.5	Summary.....	35
2.3	Supplementary variants of the geographic greedy forwarding method	36
2.3.1	A*-metric-like forwarding methods	36
2.3.2	The k -hop reach in greedy forwarding.....	37
2.3.3	The 3-D greedy forwarding.....	38
2.3.4	Randomized greedy forwarding	39
2.3.5	Beacon-less and opportunistic greedy forwarding.....	41
2.3.6	The composite greedy metrics.....	41
2.3.7	Summary.....	43
2.4	Context-aware geographic routing.....	44
2.4.1	Cost-aware geographic greedy metrics	44
2.4.2	Topology awareness in geographical packets forwarding.....	46
2.4.3	Summary.....	47
2.5	Impact of MANET topologies on protocol performance.....	48
2.5.1	Effect of node density and spatial distribution on packet routing and forwarding	48
2.5.2	The effect of node speed and mobility patterns on packet routing and forwarding ...	50
2.5.3	The effect of wireless transmission on packet routing and forwarding	53
2.5.4	Summary.....	55
2.6	Perceiving network contexts in routing and forwarding.....	57
2.6.1	Context knowledge acquisition for packet routing and greedy forwarding	57
2.6.2	Context knowledge syntheses for greedy forwarding and packet routing.....	58
2.6.3	The modelling of MANET topology in packet routing and forwarding	60
2.6.4	Summary.....	70
CHAPTER 3		71
3	THE MULTI-GREEDY FORWARDING DEFINITION	72
3.1	The multi-greedy packet forwarding model.....	72
3.2	Multi-greedy forwarding designs	76

3.3	Evaluation environment	78
CHAPTER 4		81
4	MULTI-GREEDY FORWARDING DESIGNS AND DEMONSTRATIONS.....	82
4.1	Multi-greedy forwarding based on network packet flow reports.....	82
4.1.1	Background: IPFIX-based packet flow reporting	82
4.1.2	Methodology	83
4.1.3	Simulation parameters	85
4.1.4	Results and discussion	87
4.2	Multi-greedy forwarding using packet forwarding distance dissimilarity measures	93
4.2.1	Background: topology characterizing and dissimilarity indications.....	93
4.2.2	Methodology	95
4.2.3	Simulation parameters	100
4.2.4	Results and discussion	101
CHAPTER 5.....		112
5	CONCLUSIONS	113
5.1	Multi-greedy forwarding efficiency over highly dynamic MANET topologies	113
5.2	Benefits and constraints of the multi-greedy scheme.....	114
5.3	Future works	116
REFERENCES.....		120
APPENDICES		137

List of Appendices

Appendix A: Use case: Geographic packet routing and forwarding in MANET	138
Appendix B: Multi-greedy packet forwarding evaluation design using NS-3 [235]	139
Appendix C: Parts of the C++ Codes Modified in the NS-3 Simulator	140

List of Figures

Figure 1.1: Mobile ad-hoc network types [5]	2
Figure 1.2: Variants of the geographic greedy forwarding method (adapted from [35]).....	5
Figure 1.3 An illustration of packet forwarding using multiple algorithms on varying topologies.....	7
Figure 1.4: Packet delivery performance of the GREEDY and the S* methods [5].....	9
Figure 1.5: Superimposed charts of Figures 1.4a (GREEDY) and 1.4b (S*) respectively [5]	10
Figure 2.1: CARAVAN Context Framework [14].....	18
Figure 2.2: An adaptive decision-making model [83]	19
Figure 2.3: Components of a DAWN node [25].....	20
Figure 2.4: Adaptive routing framework [26].....	21
Figure 2.5: A scenario requiring DTN routing between the partitioned segments S_1 and S_2	22
Figure 2.6: A dynamic MANET with sparse and dense sub-networks [22].	23
Figure 2.7: Controlled packets flooding of the LAR and DREAM methods [94]	25
Figure 2.8: A greedy forwarding step	26
Figure 2.9: Illustration of the MFR technique's next-hop choice-making.....	28
Figure 2.10: Illustration of the NFP technique's next-hop choice-making.....	29
Figure 2.11: Illustration of the GREEDY technique's next-hop choice-making.....	30
Figure 2.12: Illustration of the CR technique's next-hop choice-making.....	31
Figure 2.13: Illustration of the NC technique's next-hop choice-making.....	31
Figure 2.14: An illustration of a greedy failure or void situation.	32
Figure 2.15: GeoDTN+Nav architecture [31].....	34
Figure 2.16: Path selections with different greedy algorithms [42].....	35
Figure 2.17: The A* algorithm next-hop selection metric for routing path determination.....	36
Figure 2.18: Illustrations of greedy forwarding in 3-D. [111]	38
Figure 2.19 (a) – (d): Illustration of the FARINSECTOR random method [133].	40
Figure 2.20: An instance of the <i>GREEDY-COMPASS</i> hybrid approach [102].....	42
Figure 2.21: Example <i>FCPRvi</i> cost-aware evaluation for a greedy next-hop selection [145].	45
Figure 2.22: An illustration of node density variations.....	49
Figure 2.23: The IMPORTANT framework [24].	51
Figure 2.24: Data packet delivery ratio vs speed for different mobility models [177].	52
Figure 2.25: Effect of transmission power level on network connectivity (Adapted [178]).....	53
Figure 2.26: Interference and connectivity effects on MFR, MVR and NFP [43]	54
Figure 2.27: Metrics of MANET packet routing (adapted [24]).....	55
Figure 2.28: Conceptual performance vs knowledge trade-off [84]	58
Figure 2.29: Example graph with 5 vertices and 4 edges	61
Figure 2.30: Example topology matrices of the graph in Figure 2.29	65
Figure 2.31: Eigenvector (column 2) and Fiedler vector (column 3) for the graph of Figure 2.29.	66
Figure 3.1: Multi-greedy packet forwarding model with context-aware measuring units.....	73
Figure 4.1: The 200s (371 flow reports) performances of the greedy and multi-greedy methods.....	89
Figure 4.2: The 600s (886 flow reports) performances of the greedy and multi-greedy methods.....	89

Figure 4.3: The 1000s (1047 flows) performances of the greedy and multi-greedy methods.	90
Figure 4.4: The GGCE multi-greedy method performance for different forwarding durations	91
Figure 4.5: The GGC multi-greedy method performance for different forwarding durations.....	91
Figure 4.6: The GE multi-greedy method performance for different forwarding durations	92
Figure 4.7: The multi-greedy methods' PDR optimal performance: expected versus actual.	93
Figure 4.8: GREEDY (v_1) and ELLIPSOID (v_2) methods' next-hop choice-making.	98
Figure 4.9: Packet forwarding distance averages by node size (VANET case).....	102
Figure 4.10: Packet forwarding distance averages with trendlines (MANET case).	103
Figure 4.11: Packet forwarding distance dissimilarity distribution and PDR averages (VANET)	105
Figure 4.12: Average hopcount (VANET case).....	106
Figure 4.13: Average delay (VANET case).....	107
Figure 4.14: Packet forwarding distance dissimilarity distribution and PDR averages (MANET).....	109
Figure 4.15: Average hopcount (MANET case).....	110
Figure 4.16: Average delay (MANET case)	111

List of Tables

Table 4.1: Simulation parameters for the IPFIX flow-based multi-greedy packet forwarding	86
Table 4.2: Simulation Parameters for Topology Characterizing	101
Table 4.3: Topology character of the VANET environment	104
Table 4.4: Topology character of the MANET environment.....	108

List of Equations

$unext - relay(MFR) \leftarrow vi \mid maxdistuvi'$	(2.1).....	28
$unext - relay(NFP) \leftarrow vi \mid mindistuvi'$	(2.2).....	29
$unext - relay(GREEDY) \leftarrow vi \mid mindistvid$	(2.3).....	30
$unext - relay(CR) \leftarrow vi \mid minangle\angle viud$	(2.4).....	31
$unext - relay(NC) \leftarrow vi \mid mindistuvi$	(2.5).....	32
$unext - relay(A *) \leftarrow vi \mid mindistuvi + vid.$	(2.6).....	36
$FCPRvi = Cost(uvi)ud - vid ,$	(2.7).....	44
$NADVvi = ADV(vi)Costvi = Ds - D(vi)Cost(vi)$	(2.8).....	45
$= 1FCPRvi$	(2.9).....	46
$U(x1, x2, \dots xn) = i = 1nUi(xi).$	(2.10).....	59
$max\{f(U(x1, x2, \dots xn)) = i = 1nai(xi) Uivi(xi)\}.$	(2.11).....	59
$U = i = 1N((fs, ici)wi$	(2.12).....	59
$DFDx1, x2, \dots xn = DFDmax. I = 1Nwixi. DFD(xi)$	(2.13).....	60
$V = vi$	(2.14).....	61
while $E = ei, j \mid vi, vj \in V; i \neq j$	(2.15).....	61
$Nu = vi \mid u, vi \leq r$	(2.16).....	61
$d(v) = N(v) $	(2.17).....	61
$= 1Vi = 1VNvi$	(2.18).....	61
d or $density = n/A$	(2.19).....	62
$PG = n2m - 1$	(2.20).....	62
$PG = pm1 - pn2 - m$	(2.21).....	62
$h := j = 1rw1 - jh(2j)$	(2.22).....	63
$Ct = vivj \mid vi, vj \in V, pi, j \neq 0$	(2.23).....	64
$Ect = ei, j \mid vi, vj \in C(t)$	(2.24).....	64
$C(t)k = 2 Ect Ct (Ct - 1)$	(2.25).....	64
$cvi = 1\lambda vj \in NviAij cvj, \quad \forall v \in C \subseteq G$	(2.26).....	65
$Ann_x = \lambda x$	(2.27).....	65
$Lx = \lambda x$	(2.28).....	66
$dist(u, v) = ux - vx^2 + uy - vy^2$	(2.29).....	67
$v = \Delta d \Delta t$	(2.30).....	67
$relveliu, v = diu, v - di - 1u, vti - ti - 1$	(2.31).....	67
$eu, v = \{u, v, t, \Delta t\}$	(2.32).....	67
$= R \times velmax \times density$	(2.33).....	67
$PER = i = 1mAERi2$	(2.34).....	68
$ETX = 1rfrr$	(2.35).....	68
$ETXu, v = erelveliu, v\beta rfrr$	(2.36).....	68
$cdc(u)(t) = Nu(t - T) \cup Nu(t) - Nu(t - T) \cap Nu(t)Nu(t - T) \cup Nu(t)$	(2.37).....	68
$MF(u) = 1 - Nut \Delta Nu(t - THELLO)Nu(t) \cup Nu(t - THELLO)$	(2.38).....	69
(2.39).....		70

$= f_{x1, \dots, xn}. \quad = V_{1nxj} \quad (3.1)$	75
$f_{x1, \dots, xn} = \bigwedge_{1 \leq h \leq V_{1nxj}} (3.2)$	75
$PDR = rs \quad (3.3)$	79
$= 1_{nxj} IPFIX_{flowi} \quad (4.1)$	83
$= 1_{p1nxj} IPFIX_{flowi} \quad (4.2)$	84
$= 1_{p1nxj} \max PDR_{IPFIX_{flowi}} \quad (4.3)$	84
$= i = 1_{pGREEDY_ELLIPSOID_GREEDY_COMPASS} \max PDR_{IPFIX_{flowi}} \quad (4.4)$	84
$unext - relay(ELLIPSOID) \leftarrow vi \mid \min_{distu} vi + vid \quad (4.5)$	85
$unext - relay(GREEDY - COMPASS) \leftarrow vi \mid \min_{distviaCRd, vi(b)(CR)d} \quad (4.6)$	85
$dGF_{eu, vi}(t) = dist_{u, vit}. \quad (4.7)$	95
$DGF = dGF_i(t) \mid i = 1, 2, 3, \dots, DGF \quad (4.8)$	96
$dGF = 1_{ DGF } i = 1_{ DGF } dGF_i. \quad (4.9)$	96
$D(ai, aj) = k = 1_{naik - ajk}^2. \quad (4.10)$	96
$D(ai, aj) = ai - aj \quad (4.11)$	96
$dfb(k), b(m) = b(m)b(k) \quad (4.12)$	97
$= dGF(ELLIPSOID)(k) - dGF(GREEDY)(k) \quad (4.13)$	99
$= dGF(ELLIPSOID)(m) - dGF(GREEDY)(m) dGF(ELLIPSOID)(k) - dGF(GREEDY)(k) \quad (4.14)$	99
$= V_{1nxj} \max PDR_{b(k)} \quad (4.15)$	100

List of Abbreviations

AER	Average Encounter Rate
CR	Compass Routing
ETX	Expected Transmission Count
FANET	Flying Ad Hoc Network
GPS	Ground Positioning System
GPSR	Greedy Perimeter State Routing
IP	Internet Protocol
MANET	Mobile Ad Hoc Network
MFR	Most Forward within Range (or Radius)
MVR	Most Forward with Variable Radius
VANET	Vehicular Ad Hoc Network
NC	Nearest Closer
NFP	Nearest with Forward Progress
PDR	Packets Delivery Rate
RPGM	Reference Point Group Model
RWP	Random Way Point
SGT	Spectral Graph Theory
UAV	Unmanned Aerial Vehicle
WSN	Wireless Sensor Network

Definition of Terms

Connectedness	State of a link's existence between adjacent network nodes
Connectivity	A chain existence of links between a pair of network nodes which enables multi-hop communication exchanges
Delay	The amount of time it takes for a packet piece to be delivered
Hopcount	Number of intermediate nodes over which packets have travelled
Hop	A step in multi-hop packet transmissions to communicate from a source to a destination
Link	The line representing direct packets' transmission capability between two nodes
Load	Level of activity or involvement of a node in packet transmissions
Local maxima	A point when a packet forwarding node is unable to continue greedy forwarding, although it can route non-greedily. Void occurrence.
Local minima	A point when a packet forwarding node is closer to destination than any of its neighbours and as such is unable to forward packet greedily. Void occurrence.
Next-hop	A neighbour node that is chosen to receive data packets for subsequent forwarding in multi-hop routing
Node	A device that is involved with packets reception and transmission in a network
Node degree	Number of directly linked neighbours to a node
Progress region	A node's packet transmission range which contains neighbours that are closer to a target destination
Overhead	The costs incurred in the process of data packets transmission and routing

Stretch factor	Ratio showing hops that packets travel compared to shortest path
Throughput	The rate at which data bits can be moved in a network
Void	A divide across which a node is unable to perform progress-based greedy packet forwarding

Declaration Regarding the Work and Copyright



POSTGRADUATE SCHOOL

Declaration Regarding the Work and Copyright

DECLARATION REGARDING THE WORK AND COPYRIGHT

Candidate: GBADEBO OLADEJI-ATANDA

ID: 17100097

Thesis Title: A MULTI-GREEDY FORWARDING APPROACH IN GEOGRAPHIC
PACKET ROUTING

I, the **Candidate**, certify that the Thesis is all my own original work and that I have not obtained a degree in this University or elsewhere on the basis of any of this work.

This dissertation/thesis is copyright material protected under the Berne Convention, the Copyright and Neighbouring Rights Act, Act. No. 8 of 2000 and other international and national enactments, in that behalf, on intellectual property. It must not be reproduced by any means, in full or in part, except for short extracts in fair dealing; for researcher private study, critical scholarly review or discourse with an acknowledgement, without the written permission of the office of the Postgraduate School, on behalf of both the author and the BIUST.

Signed:

Date: 31 July 2023

Primary Supervisor (please write in caps or type) DIMANE MPOELENGB PhD

I, the Candidate's **Primary Supervisor**, hereby confirm that I have inspected the above titled thesis and, to the best of my knowledge, it is based on the original work of the candidate.

Signed:

Date: 31-July-2023

Acknowledgements

I thank God and the Lord Jesus Christ for enabling me to perform this PhD study despite all odds. I thank Dr Dimane Mpoeleng who supervised my PhD work. I acknowledge his kind invitation for me to come over to BIUST after we both differently left the University of Botswana (UB) where this study was initially proposed. I feel highly grateful for his rolling supportive gestures and academic support. Professor Francis J Ogwu, formerly of UB, stimulated my interest in networks. He supervised my MSc study and encouraged my embarking on the PhD research. The research work was performed in the laboratory office of Mr Andrew Ramarumo, the Computer Science and Information Systems (CSIS) Dept IT Technician. Apart from sharing with me his office space, Andrew was quite handy and helpful in technical issues and in companionable ways. The BIUST research initiation grant (project code S00122) enabled me to attend the workshop on ns-3, WNS3 2019, at the University of Florence, Italy. The ns-3 network simulator is the major tool I used for the research. The following postgraduate fellows offered inspirations on the road that we travelled together: Dr Mrs Oarabile Sebubi, Ms Kaloso Topias, Ms Marang Mbaakanyi, Mr M Marope, Ms Fatima Watlhaga, and Ms Bonani Thela. Tapela and Tshepo were fellow Research Assistants on the "drones for health project" supervised by Dr Mpoeleng. I appreciate the academic staff of the CSIS, especially Dr H Hlomani who was nearly always there at the progress presentations. I am thankful to those I worked with as a Teaching Assistant at the Centre for Business Management, Entrepreneurship and General Education, which contributed profoundly my academic writing skills. Mr M Moremi, Dr Mrs G Buru, Mr B Mogotsi, Dr F Bughio, Ms A Macha, Dr Mrs T Mokgwathi, and especially Mr Kelebonye Bagai who I warmly worked with for an extended period. Cheers to Beatrice Omenebele Oladeji-Atanda, my wife, who allowed my protracted absence to embark on this programme. So also to my children: Simon Laolu, Abigail Oluwagbemisola, and Mary Ebunoluwa.

Dedication

This work is dedicated to the memories of my parents and grandparents, all of whom I had enjoyed growing up with: Emmanuel Oladeji Atanda (father) who regularly took me to the house of God as a little child; Mopelola Abike Atanda (mother) who exhibited a profound godly character; James Atanda (paternal grandfather) who was firm in Christian faith; Maria Adejoke Atanda (paternal grandmother) who has a characteristic way of pronouncing Jesus whenever praying; Emmanuel Adeniyi (maternal grandfather) whose antique Yoruba language Bible I had wished to inherit; Fayoyin Anike Adeniyi (maternal grandmother) who was quite endearing and open-handed. Also dedicated to my dear aunt Rhoda Bolajoko Fateru (nee Atanda) and her husband Samson Adebawale Fateru who over several years gave me active living support.

Abstract

This thesis presents a *multi-greedy* approach to efficient *progress-based* forwarding in geographic packet routing over fluctuating and varying mobile ad hoc network (MANET) topologies. Several factors cause MANET topologies to vary, including node mobility, node size fluctuations, and wireless link vagaries. The pertinent packet routing environments have demanded specialized, context-aware, and hybrid approaches. The distributed geographic routing method is found most suitable for MANETs of extremely varying topologies, such as the emergent vehicular ad hoc networks (VANETs) and flying ad hoc networks (FANETs). In the progress-based geographic routing, a greedy forwarding algorithm at each intermediate node computes the *next-hop* to be a neighbour that is *closer* in Euclidean distance to the packets' final destination. The choice is also determined by the greedy algorithm's supplementary criteria, e.g., the Compass Routing's selection of a neighbour that also has the minimum angle wedged between the next-hop and the destination. Additionally, the underlying network topology influences the efficacy of greedy algorithms. Thus, each greedy forwarding algorithm matches specific topologies or requirements best. However, literature shows that the existing geographic routing protocols perform progress-based forwarding using a sole greedy method, which is inefficient over topologically varying environments. That is, the existing geographic routing protocols do not switch the greedy computation methods for efficient adaptation to topology changes in progress-based forwarding. Accordingly, a *multi-greedy model* is defined in this thesis to address packet forwarding efficiency in geographic routing over fluctuating and varying MANET topologies. A multi-greedy model consists of selected existing greedy metrics that are each assignable and re-assignable to a routing task for optimized forwarding performances over varying topologies. Based on the defined model, *context-aware* multi-greedy designs can be specified. Two example designs with sample multi-

greedy formulations are evaluated. The first design is a multi-greedy forwarding approach using the ‘IP Flow Information Export (IPFIX)’ packet delivery performance reports. The IPFIX-derived multi-greedy formulations, especially GREEDY~GREEDY-COMPASS~ELLIPSOID (GGCE), showed improved forwarding efficiency in comparison to its greedy constituents over a simulated VANET environment. For example, a packet delivery rate of 66.96% was achieved by the GGCE method in comparison to 59.51, 61.61, and 62.58% of its three constituents including GREEDY, ELLIPSOID, and GREEDY-COMPASS respectively. The second design is a topology-aware demonstration of multi-greedy forwarding efficiency using average *packet forwarding distance* dissimilarities. Accordingly, the *dissimilarity* indices of a MANET’s topology by node size can be matched to selected forwarding metrics for adaptive multi-greedy performance. The simulation results show that multi-greedy forwarding would be beneficial in the examined VANET case, but vaguely so for a less dynamic MANET case. With its simple ‘if-else’ condition checking from coupled context-aware schemes, the multi-greedy forwarding mechanism can replace the conventional method in most existing geographic routing protocols.

Keywords: mobile ad hoc networks, position-based protocols, geographic greedy packet routing, context-awareness, multi-greedy forwarding, topology characterizing.

CHAPTER 1

1 INTRODUCTION

A *multi-greedy* approach to improving progress-based forwarding in geographic packet routing over dynamic mobile ad hoc networks (MANETs) is defined and evaluated in this thesis. Progress forwarding reduces a packet's travel distance continually toward a routing destination. A MANET consists of geographically dispersed and arbitrarily located utility devices that can communicate wirelessly (Figure 1.1) and form *links*. The nodes thus cooperatively self-configure the network and inter-operate peer-to-peer without a central control. Moreover, the nodes act as intermediate routers in *multi-hop* data packet relaying from one endpoint to another. However, node mobility, dynamic changes or *churn* in node size [1][2], and wireless channel factors such as attenuation, shading, interference, etc., affect MANET links [3][4] which cause variations in the underlying topologies. Extreme incidence of the context factors leads to a rapidly metamorphosing topology that has required the geographic or position-based method in packet routing. High mobility environments, such as the vehicular ad hoc network (VANET) and the flying ad hoc network (FANET), exhibit endemic topology variations [3][4][1].

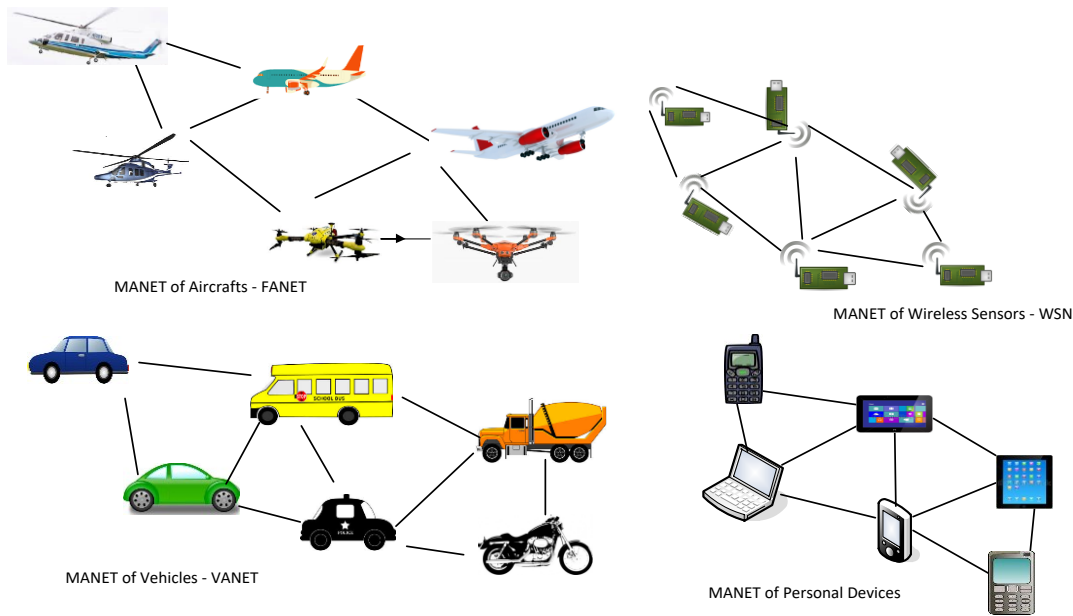


Figure 1.1: Mobile ad-hoc network types [5]

VANET provides the structure for seamless inter-vehicular situation awareness, collision-prevention, and traffic management, as well as delivering infotainment services to commuters, e.g. video streaming, advertisements, etc. [6][7][8][9]. The Intelligent Transport Systems community view the VANET scheme as a much cheaper and eco-friendly technology compared to conventional roadside installations [10][11]. VANET is also important in the implementation of the nascent autonomous driverless vehicles. On its part, FANET supports relevant airborne data transmission relays such as are required in Unmanned Aerial Vehicles (UAV) flight formation and communication, e.g., during natural disasters, military manoeuvres, telematics, etc. [6][7][12][13]. This thesis addresses the efficacy of progress-based packet forwarding in geographic routing over varying and fluctuating MANET topologies and provides a multi-greedy adaptive approach to performance improvement.

1.1 Context and topology instability in highly dynamic MANET environments

MANET context factors relate to the attributes of nodes' dynamic positional coordinates, wireless channel vagaries, and QoS requirements [14]. It is well known that the effectiveness of routing protocols in highly dynamic MANET is influenced or impeded by sporadic context variations [14][15][16]. The routing challenges of the multifarious environments of MANET have thus spurred several protocol designs for addressing specific context issues [17][18][19][9][20]. It is claimed that there is no single routing protocol [14][21][22][3] or a path-finding algorithm [15][20][23] that performs consistently well in all MANET environments. For example, the Optimized Link State Routing (OLSR) protocol is shown to perform well in high-density environments, while Destination Sequenced Distance Vector (DSDV) is recommended for the opposite case [19]. In [24], the protocols of Dynamic Source Routing (DSR), Ad-hoc On-Demand Vector routing (AODV) and the DSDV are found to exhibit performance characteristics of varied

kinds under different MANET mobility models, including Random Waypoint, Group Mobility, Freeway, and Manhattan. In order to address routing efficiency over mixed environments, researchers shifted towards context-aware adaptive and hybrid protocol designs [14][21][22][25][26][27]. The Zone Routing Protocol (ZRP) [27][28] is a classic example of a hybrid routing scheme, which incorporates IP-based *proactive* and *reactive* routing methods. The proactive method incurs high table maintenance costs but has routing maps readily available at the network nodes, but the opposite is the case with the reactive method. The ZRP performs intra-cluster routing proactively but inter-cluster reactively. The Context-AwaRe Architecture for VANET (CARAVAN) [14] defines a framework for context knowledge acquisition to support adaptiveness to the IP-based and Delay Tolerant Network (DTN) routing methods. The DTN approach involves packets' buffering or 'store-and-carry' across network partitions. Beyond the IP-based approaches, position-based or geographic routing protocols emerged, including hybrid types, to address highly dynamic MANET environments [9][29] such as VANET and FANET. Examples of geographic protocols are the classic Greedy Perimeter Stateless Routing (GPSR) [30], GeoDTN+Nav [31], and the Hybrid Geographic Routing (HGR) protocol [32]. GPSR performs greedy packet forwarding and whenever a failure occurs its perimeter routing method is switched on for recovery. Appendix A shows a use-case model of MANET geographic packet routing with highlight on progress-based greedy forwarding.

1.2 Geographic progress and greedy forwarding

Geographic packet routing [33][18][34] over dynamic MANETs is enabled by greedy forwarding algorithms that comprise localized computations in routing decisions, which scales quite well to large and unstable topologies [35][36][37][38][30]. Generally, the class of greedy algorithms are applicable to problems that exhibit optimal substructures, such that series of intermediate localized

choices effectively lead to an approximate global optimum solution [39]. The ease of maintaining neighbour tables containing positions' data at the network nodes makes the greedy method to be resource-efficient, in processing and storage, compared to the routing table methods [40][18]. A geographic routing node that has packets to forward requires the knowledge of (i) its location, (ii) its immediate neighbours' positions, and (iii) the packet's destination [35]. The nodes' location knowledge can be determined through services such as GPS, beacon messaging, cellular-sensing, etc. [14][41]. The location coordinates of the nodes are used in greedy computations and forwarding which generally aims at following minimum multi-hop traversal distance to destinations. When a node has packets to forward, its copy of the geographic greedy algorithm selects a *next-hop* to be a neighbour that is *closer* to the target destination. In addition, the algorithm makes the next-hop choice based on its secondary criteria. For example, the Nearest Closer (NC) method chooses a next-hop that is also closest to the currently forwarding node [35][36]. Other classic greedy algorithms include, Most Forward within Range (MFR), GREEDY, Nearest with Forward Progress (NFP), and Compass Routing (CR) [35][36][38][42]. Figure 1.2 depicts a currently forwarding node u , with the transmission range r , which must select a next-hop towards destination d . The node u shall select the neighbour that is marked NC as next-hop if that method is employed. Every intermediate node in a network must perform similarly in *progress-based* greedy forwarding until the destination is reached [35][36].

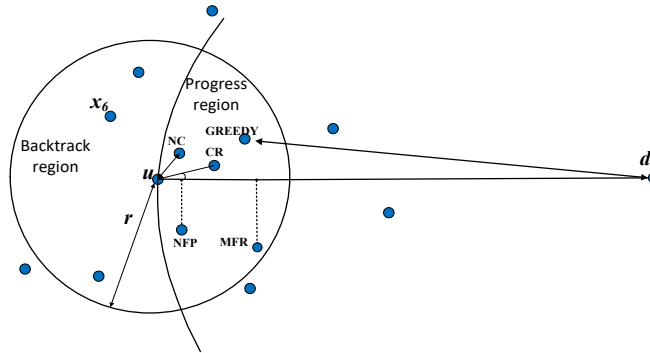


Figure 1.2: Variants of the geographic greedy forwarding method (adapted from [35])

The GREEDY forwarding technique (see Figure 1.2) is popular in geographic routing protocol designs [37][9] due to its secondary characteristic of choosing next-hops having the least Euclidean distance to the target destination and its implementation simplicity [29]. In the Euclidean space, distance between two points is determined by the straight line that connects them, ignoring any obstacle and earth's curvature.. Both the NC and the NFP techniques are employable in designs that aim for minimizing transmission power costs [35]. NFP is further described as good at minimizing collisions in environments with packet traffic congestions [38][43]. The MFR and the GREEDY methods are good at moving packets faster but often select next-hops at transmission range fringes where link connectedness is unstable [44][45]. The durability of links is an important issue in dynamic MANET packet routing [46][47]. For example, to minimize the fringe effect, the Context-aware Adaptive Opportunistic Routing (CAOR) protocol operates greedy forwarding with the transmitter range bounds of $[0, 2/3r]$. Among the classic greedy methods shown in Figure 1.2, it is only the CR method that is not strictly progress-based. The nodes in the backtrack region of the transmission range, as illustrated in the figure, are also potential choices for the CR method if none exists in the progress region. Greedy failure, aka *void*, occurs when no neighbour is found in the progress region. The Angular Routing Protocol (ARP) [48] is a hybrid that uses the GREEDY technique to forward packets normally in progress-based performances; but when a void is encountered it switches to the CR technique, which engages reverse forwarding for recovery. Nevertheless, the geographic routing protocols, including ARP, CAOR, GPSR [30], GeoDTN+Nav [31], etc., are not designed for the adaptive switching of different greedy algorithms in efficient progress forwarding. This thesis considers improvement capabilities to normal progress-based forwarding through adaptive switching over varying topologies.

1.3 Problem statement

The existing geographic packet routing protocols are typically inefficient in progress greedy forwarding over highly dynamic MANETs, due to non-adaptivity to varying and fluctuating underlying topologies. Network topology states influence the geographic routing protocol's performance outcomes [43][36]. A greedy forwarding algorithm yields sub-optimal outcomes when its attributes do not quite match a topological terrain.

Therefore, it is necessary that greedy forwarding be adaptive to changing topological structures in the performance of packet routing tasks. Figure 1.3 illustrates how progress forwarding could be improved in packet delivery efficiency by a scheme that employs multiple greedy methods to adaptively perform as the node size changes along the routing path. The geographic packet routing method could employ the most suitable greedy algorithm over each of the varying environments going from a dense to a sparse topology. The different greedy metrics, designated in the Figure 1.3 legend as AlgGreedy_x, could each be suitably employed in the dense, less dense, and sparse segments for more efficient protocol performance. Note that a greedy algorithm may have a performance edge in more than one topological condition. Metrics that describe topology states, such as the average node degree and the average node velocity, are found in [49][50][51][52][53][54].

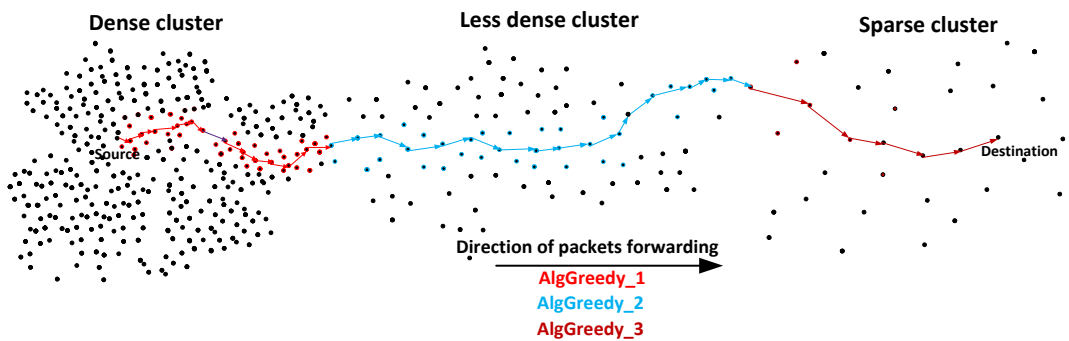


Figure 1.3 An illustration of packet forwarding using multiple algorithms on varying topologies.

1.4 The rationale for multi-greedy packet forwarding

The dynamic MANET environments of heterogeneous and varying topologies have demanded the designs of context-aware adaptive and hybrid multi-hop packet routing strategies, including the geographic kinds, for more efficient performances [14][21][22][25][26][27]. The IETF MANET Working Group's objective postulates that the design of routing protocols should 'provide for effective operation over a wide range of mobile networking contexts' and 'react efficiently to topological changes' [55]. Adaptive routing that engages multiple greedy forwarding metrics for progress-based performance provides a means for efficiency improvement in geographic packet routing over topologically changing MANET environments. This rationale is supported by the following standpoints.

- **Performance differences among the greedy methods**

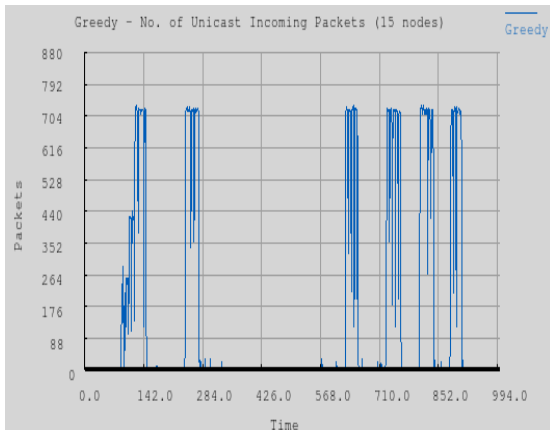
The existing greedy forwarding techniques, i.e., MFR, GREEDY, etc., each have distinct performance characteristics that are also influenced by the underlying topology of a network [5][18][34][35][36][37][38][43][56]. Thus, the effectiveness of each method in an environment can be ranked and serve as the basis for adaptive assignments over varying topologies.

- **Functional similarities of the greedy methods**

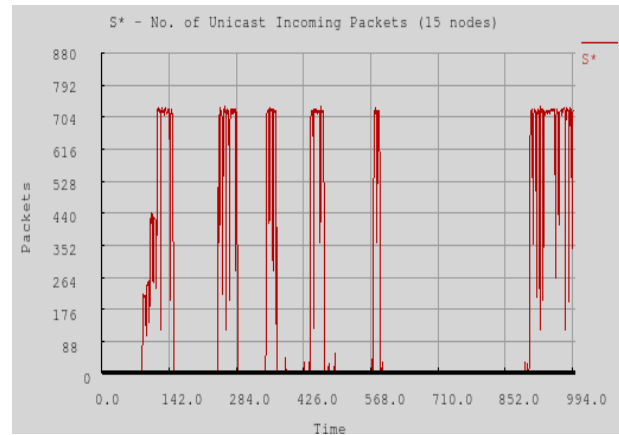
The geometric computations of determining or choosing a next-hop neighbour based on location coordinates are similar for all geographic greedy methods. Moreover, regardless of the method, the same neighbour-table data is accessed in the greedy processing procedure at a forwarding node. These commonalities make the runtime adaptive switching of the methods robust in a multi-greedy hybrid scheme. Besides, the switching overhead is negligible.

- **Example performance pattern of two greedy methods**

The study in [5] showed results that exhibit packet delivery performance patterns of two greedy forwarding methods which suggests leveragability for efficient performance in an adaptive scheme. Figure 1.4 shows the separate packet delivery performance charts of the GREEDY and the S* (aka ELLIPSOID [57][58]) methods over the same VANET environment. Generally, both methods exhibit instability in packet delivery performance as shown by the fluctuating chart patterns. Similar instability phenomena are described in [49][51][52][53][54]. However, Figure 1.5 shows the outcome of the statistical superimposition of the two charts in 1.4a and 1.4b. Scatter chart symbols are used for the clarity of the superimposition. The resulting chart (Figure 1.5) depicts minimized instability of the packet delivery pattern, which suggests derived improved performance through an appropriate alternating application of the methods.



(a): Packets delivery using GREEDY method



(b): Packets delivery using S* method

Figure 1.4: Packet delivery performance of the GREEDY and the S* methods [5]

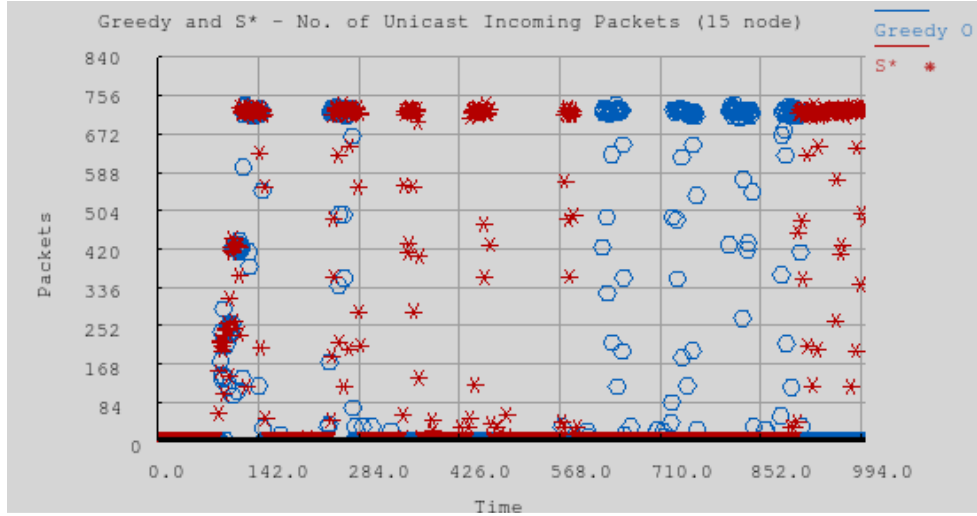


Figure 1.5: Superimposed charts of Figures 1.4a (GREEDY) and 1.4b (S*) respectively [5]

1.5 Research objective

The main objective of the study is to define a multi-greedy model for progress packet forwarding which encompasses multiple greedy methods, with each assignable to perform optimally over a specific MANET topological context in self-adaptive geographic routing. The aim is to specify a hybrid greedy multimethod packet forwarding scheme for adaptive efficient progress-based performance over varying and fluctuating topologies.

The specific objectives of the study are:

- (i). To examine performance differences of greedy algorithms over classes of topological contexts and ascertain the potential for multi-greedy progress forwarding.
- (ii). To define a multi-greedy forwarding model that comprises switching multiple metrics for progress greedy task performances in context-aware adaptive geographic packet routing.
- (iii). To demonstrate example multi-greedy forwarding designs in MANET geographic packet routing, premised on applicable context-awareness strategies, and evaluate same in successful packet delivery measures.

The objective postulates that the conventional geographic routing protocols, e.g., the GPSR [30], that operates with a greedy forwarding metric should rather be equipped with a set of metrics that are switchable for efficient adaptive performance to improve packet delivery success rates.

1.6 Research question

The question is that can multiple greedy methods be hybridized, with each characterized by its optimum neighbour selection attribute, for adaptive geographic routing to improve progress-based packet forwarding efficiency over varying and fluctuating MANET topologies?

Specific research questions are:

- (i). How could the MANET's varied topologies be classified in association with the different greedy techniques' neighbour selection and packet forwarding performance attributes?
- (ii). How could a multi-greedy scheme be defined to address the optimization requirement of engaging multiple greedy forwarding methods in progress-based geographic packet routing tasks?
- (iii). In conjunction with conditional testing of MANET contexts, how could specific designs of adaptive multi-greedy packet forwarding be achieved?

1.7 Evaluation method

Evaluation of the multi-greedy packet forwarding proposal was performed with the NS-3 [59] network simulator, which also incorporates the GPSR [30] geographic packet routing protocol module. Appendix B illustrates the evaluation process using NS-3. The GPSR's void recovery module was disabled in the simulation tests since the focus of this study is on progress-based

packet forwarding. The geographic routing protocol's primary performance indicators, including PDR, hopcount, and delay, were used to evaluate the efficiency of the multi-greedy formulations in comparison to any related constituent parts. The multi-greedy simulation experiments examined the unicast end-to-end packet routing and forwarding efficiency as influenced by fluctuating MANET topologies. Any packet routing improvement scheme should maintain acceptable levels of extant protocol performance measures, such as memory requirements, and service continuity [21] [60] [26] [61]. All multi-greedy forwarding designs should maintain similar or acceptable overhead costs in comparison to the related constituent greedy base types.

1.8 Contribution to knowledge

The study on multi-greedy forwarding contributes a solution to the quest for hybrid protocol designs that aim to improve packet routing efficiency in highly dynamic MANETs [14][21][22][25][26][27]. The specific contributions of this work to geographic packet routing knowledge are:

- (i) The definition of a multi-greedy model for the progress-based forwarding performances in geographic packet routing over varying and fluctuating MANET topologies. Although various hybrid approaches are described in the geographic routing literature, none is known to engage different greedy types for progress-based forwarding efficiency through adaptive turn-taking of multiple greedy algorithms.
- (ii) The design of a multi-greedy forwarding approach that is based on context-aware measures deriving from the IP Flow Information Export (IPFIX) [62][63] protocol's packet delivery reports. This design is exploration-exploitation oriented for repetitive routing task performances over a dynamic environment. Nevertheless, the geographic routing of per-hop forwarding is inherently repetitive.

- (iii) The design of a multi-greedy forwarding approach using greedy *packet forwarding distance* dissimilarity indices. The study shows that a greedy method's packet delivery performance chart does not generally exhibit a linear relationship with network node size, whereas the packet forwarding distance distribution does. Thus, these counteracting measures are considered leverageable in formulating the multi-greedy scheme that improves the PDR, etc.
- (iv) The identification of the 'packet forwarding distance' as a metric for *learning* and characterizing a network's topology by node size. The intra- and inter-greedy methods' comparisons show the packet forwarding distance measure as a function of node size. The packet forwarding distance measures of a MANET can be used to fix estimates in node size.

1.9 Summary

This thesis presents a multi-greedy forwarding scheme for geographic packet routing in heterogeneous and dynamically varying MANET topologies. Generally, the conventional IP-based MANET protocols, e.g. AODV [64] [19] and ZRP [27][28], are not suitable for routing over rapidly changing topologies. The method of position-based or geographic greedy packet routing [35][36][37][38] is found more suitable for highly dynamic MANETs that are characterized by rapidity in node movements, such as the VANET [9][31] and FANET [65][66][18]. The multi-greedy approach aims at improving efficiency in progress-based packet forwarding over dynamic topologies through context-awareness strategies and adaptive performance.

The next section, i.e. Chapter 2, sets out the literature position on MANET hybrid packet routing protocols, context awareness schemes, and geographic greedy forwarding. In Chapter 3, the multi-greedy model and the requisite evaluation tools are presented, while Chapter 4 shows two

examples of multi-greedy implementation design with sample formulations and performance demonstrations. Chapter 5 provides the thesis' conclusions.

CHAPTER 2

2 LITERATURE REVIEW

MANET literature indicates that the design of routing protocols for multi-hop data packet delivery is quite challenging. A MANET environment is typically dynamic because its routing context elements tend to occur at extremes, such as radio interferences, node mobility, and fluctuating density[3][67][68]. Its topological states can vary substantially, both spatially and temporally [14][16]; which escalates when the networking area is relatively large [22][28][30]. The emergent MANET types of vehicular ad hoc networks (VANET) [14][9][31] and flying ad hoc networks (FANET) [66][38] contain highly mobile nodes that also fluctuate much in size. The challenges of the MANET environments have prompted the design of several routing protocols, including the proactive, reactive, delay-tolerant, geographic, and hybrid types. Section 2.1 below provides an overview of the general trend in MANET hybrid routing protocol designs. Sections 2.2 to 2.4 outline the geographic packet routing approaches with a focus on the greedy forwarding method. Sections 2.5 and 2.6 cover models of MANET context ontologies and knowledge acquisition methods which support adaptive packet routing.

2.1 MANET routing environments and hybrid context-aware routing protocols

As mentioned above, the MANET environment presents several challenges to routing protocol development and implementation. Existing standalone protocols have limited applicability as each is only efficient in specified MANET environments [14]. Besides the goal of maximizing packet delivery rates, other issues that MANET protocol designs address include overhead minimization, complexity reduction, scalability, robustness, load balancing, fault tolerance [69], etc. The early designs of context-aware and hybrid routing protocols comprise approaches that switch methods based on hierarchical levels of a networking area or certain encounters such as network partitions.

2.1.1 The MANET zoning and hierarchical hybrid routing protocols

The classic MANET routing protocols, including the Optimized Link State Routing (OLSR), Destination Sequenced Distance Vector (DSDV), Ad-hoc On-Demand Vector (AODV), and Dynamic Source Routing (DSR) [70][71] operate with routing tables that are maintained at each network node. OLSR and DSDV are classed as *proactive* which requires that the tables be constantly updated, while AODV and DSR are *reactive* by updating only *on demand*. The proactive approach ensures that end-to-end routing information is readily available at each network node. While this minimizes *delay*, the periodic table updates generate relatively high overhead. The early hybrid protocol designs incorporate these two methods. The Zone Routing Protocol (ZRP) [27][72][55] is a pioneering hybrid method that *zones* its networking area to perform *intra-cluster* packet routing proactively and *inter-cluster* reactively. The ZRP model achieved significant control-messages reduction for large MANETs, albeit in near-static environments [27]. The Adaptive ZRP (AZRP) [73] extended ZRP for environments of high node mobility. AZRP reduces the zone sizes whenever network perturbation upsurges, thereby regulating the intra-cluster routing overhead. Other zone-carving protocols include the Zone-based Hierarchical Link State (ZHLS) [74], Hybrid Ad hoc Routing Protocol (HARP) [50], Sharp Hybrid Adaptive Routing Protocol (SHARP) [15], Adaptive Hybrid Domain Routing (AHDR) [75], and the Scalable Location Update-based Routing Protocol (SLURP) [76]. AHDR is designed for routing in tactical airborne and ground maneuvers which address nodes by geographical *home regions*. The SLURP [76] method utilizes the geographic greedy's MFR [77] technique for inter-zone routing, while intra-zone routing within *sub-regions* is performed using the proactive DSR protocol. Terminode routing protocol [78] also has two components for *remote* and *local* routing that respectively correspond to location-based forwarding and a proactive link state method. Terminode aims to

overcome incidents of greedy failure on irregular topologies. The prevalence of routing challenges in dynamic MANET advanced the paradigm of *context awareness* in protocol designs [14][79].

2.1.2 Context-aware routing improvement methods and frameworks

The term ‘context-aware’ was initially associated with the environments of handheld devices and their mobile users [80][81]. The acquisition of knowledge about a mobile device’s environment enables the context-aware routing method to function adaptively as the situation changes [14][21]. Context-aware schemes have involved engaging multiple existing routing methods for optimized performance in consonance with varying networking environments. An example of context-aware design is the Context-AwaRe Architecture for VANET (CARAVAN) [14] which delineates a networking environment into context levels as shown in Figure 2.1. Each network node that is implementing the CARAVAN method is comprehensively aware of its local environment across the planes of *mobility*, *connectivity*, and *application*.

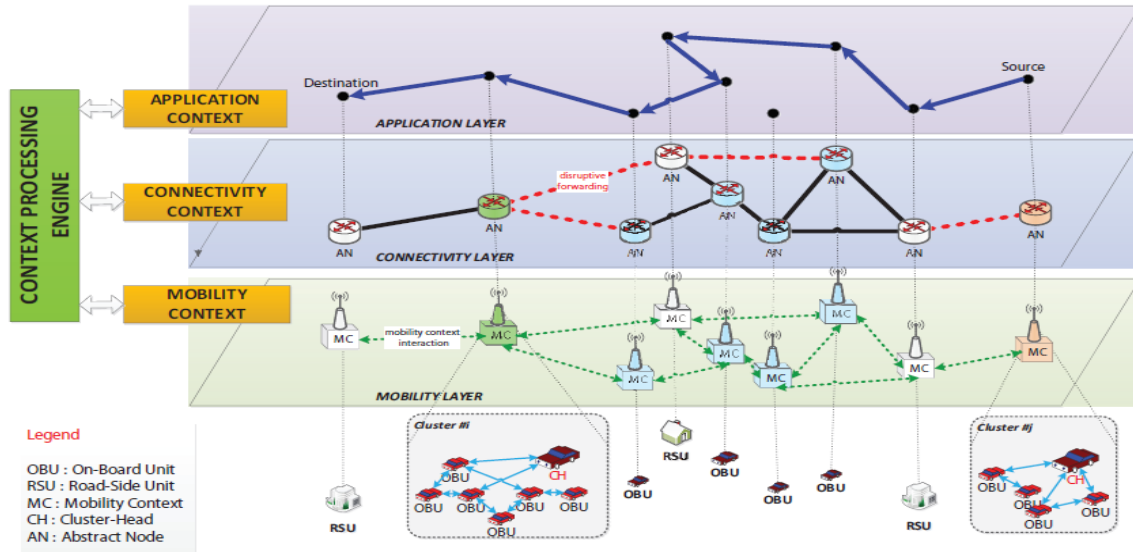


Figure 2.1: CARAVAN Context Framework [14]

Accordingly, each node periodically integrates the environment's multifarious knowledge elements using its *context processing engine*; which enables it to determine and assign from its available pool of protocols, including OLSR [71], etc., a most suitable one to the current packet routing task. The Adaptive Multimode Ad Hoc Routing (AMHR) [21] protocol is a design with an emphasis on *context monitoring* with nodes having the autonomy to switch routing methods. In AMHR framework, each node independently monitors its *local context*, such as traffic load or node mobility, and commensurately switches on a routing mode. The modes simulated for are pre-defined *proactive-reactive* conditions, which lead to engaging either the Wireless Routing Protocol (WRP) [82] or the reactive AODV [64] protocol. AMHR's authors discussed the complexities of autonomous mode switching by each network node and also recommend further studies on context monitoring at global levels using *estimation* and *heuristic* approaches. Figure 2.2 illustrates the role of context monitoring for an event-driven proactive-reactive hybrid protocol that is described as being adaptive to network traffic rates [83]. The model also highlights its context monitoring module, which amalgamates three knowledge inputs.

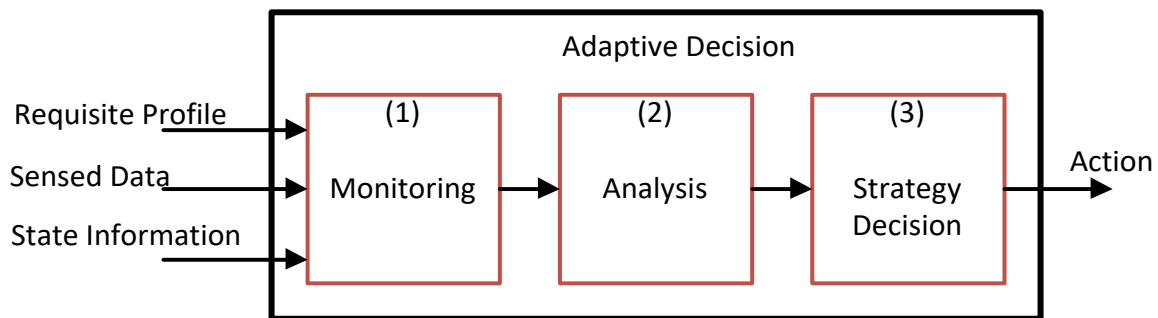


Figure 2.2: An adaptive decision-making model [83]

Figure 2.3 shows the Declarative Adaptive Wireless Networking (DAWN) framework [25]. Each DAWN node autonomously monitors the *local* network environment for its adaptive packets routing and forwarding policies. According to its authors, the DAWN’s *policy-based* and *declarative programming* approach simplifies its context-aware design objective as well as its formal proof [25]. DAWN’s nodes that exist in the same cloud tend to implement similar policies, thereby offering minimized overhead costs [25].

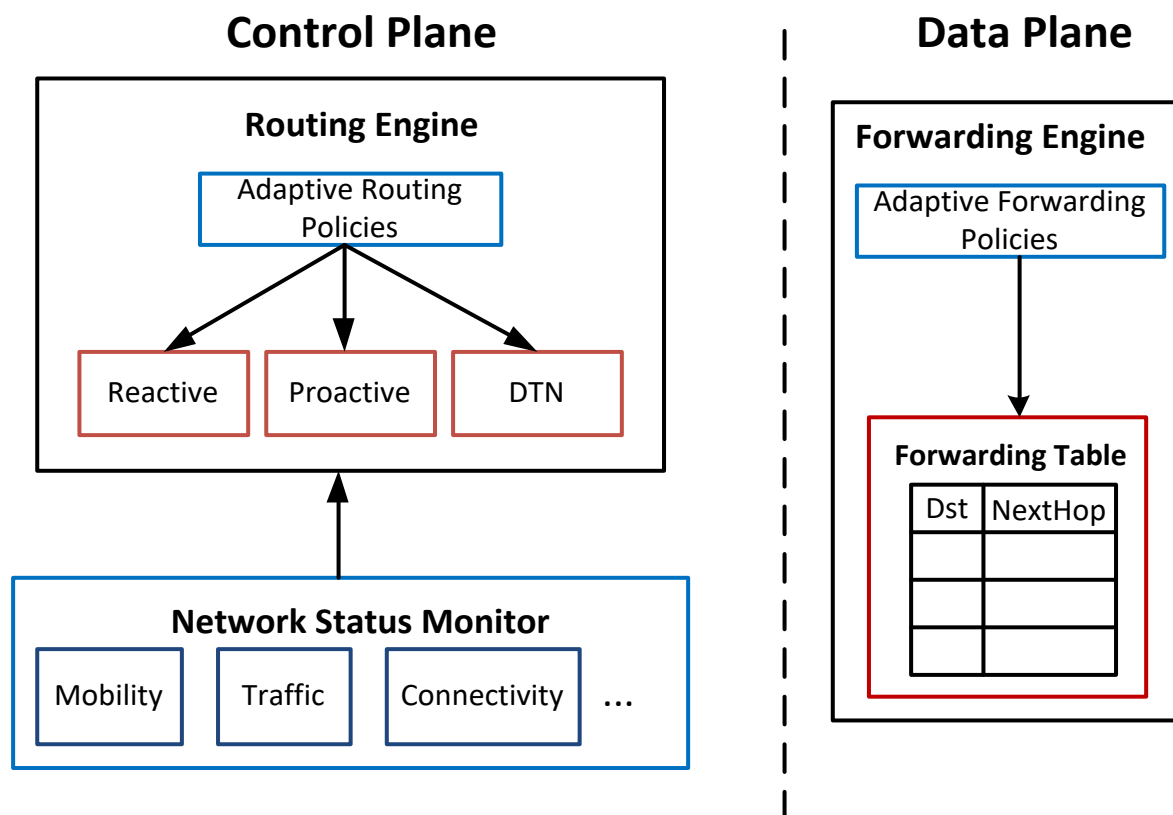


Figure 2.3: Components of a DAWN node [25]

DAWN is a hybrid method that switches between the proactive, reactive and delay-tolerant network (DTN) protocol types. The DTN approach is discussed in section 2.1.3. In contrast to DAWN’s local environment monitoring approach, Figure 2.4 shows a *global policy management*

(GPM) type [26] which is aimed at adaptive routing across *clouds*. The GPM module consists of a knowledge base, network state variables acquisition unit, etc.

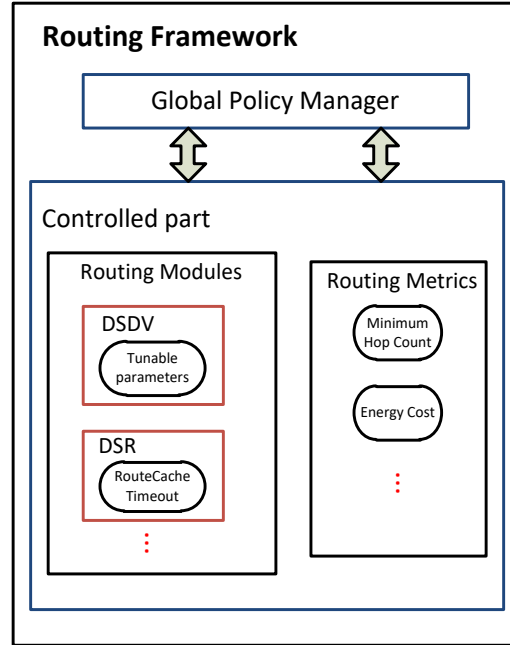


Figure 2.4: Adaptive routing framework [26]

2.1.3 The DTN routing in partitioned or intermittently connected environments

Figure 2.5 depicts an example network that has partitioned into three segments S_1 , S_2 , and S_3 . The normal packet relaying procedure is impracticable across because the nodes u and v that could link the two segments are too far apart. Routing between S_1 S_2 demands engaging a Delay Tolerant Network (DTN) protocol. The DTN protocols enable nodes to temporally buffer and carry packets across network segments and thereafter resume normal forwarding and routing [26][22][84]. Nodes that are deemed to be moving in the *expected* direction are employed in the carry-and-forward practice.

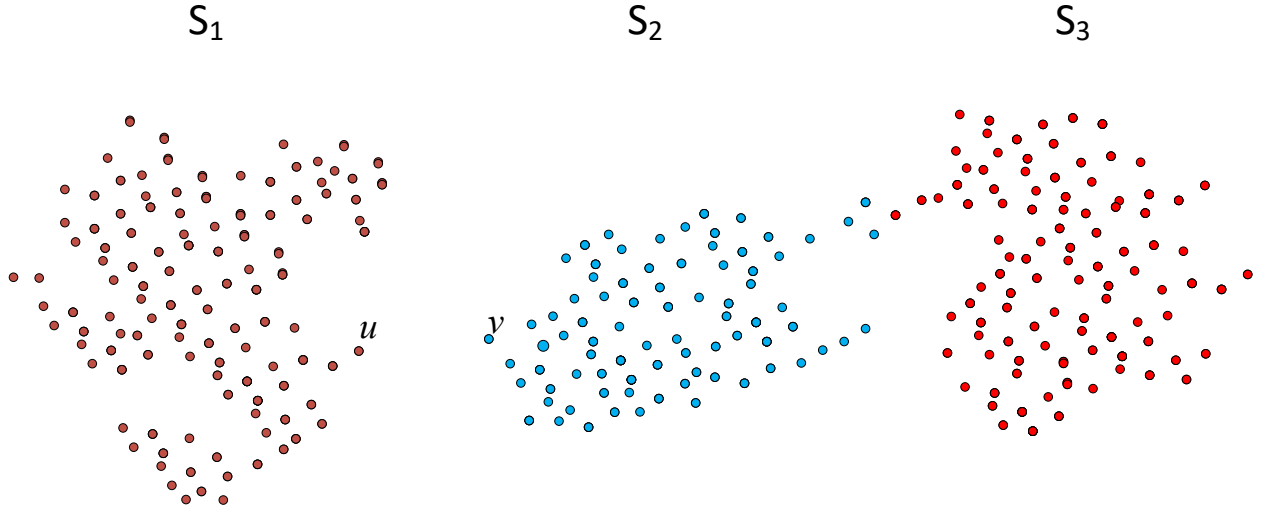


Figure 2.5: A scenario requiring DTN routing between the partitioned segments S_1 and S_2 .

Methods such as the Adaptive ROuting in Dynamic ad hoc networks (AROD) [22] and Context-Aware Routing (CAR) [85][79] are examples of DTN hybrids. The AROD protocol functions in relation to network node density. It performs proactive routing using modified DSDV [86] protocol in normal or dense conditions but switches to the DTN mode in partitioned conditions. Figure 2.6 illustrates a MANET with a dynamic sub-network (having nodes labelled 1-30) that would require the DTN routing method [22]. Like AROD, the CAR [85][79] method likewise employs the DSDV [86] or ‘any suitable protocol’ to perform *synchronous* routing when a normal multi-hop path exists. However, its core *prediction* mechanism for the *asynchronous* DTN routing is activated whenever a network partition is detected. The prediction of which next-hop node should *carry* the packets from one network segment to the other is determined by the synthesis of context knowledge. CAR’s mobility prediction feature employs the *Kalman filtering* method, which is based on observed performance trends over time.

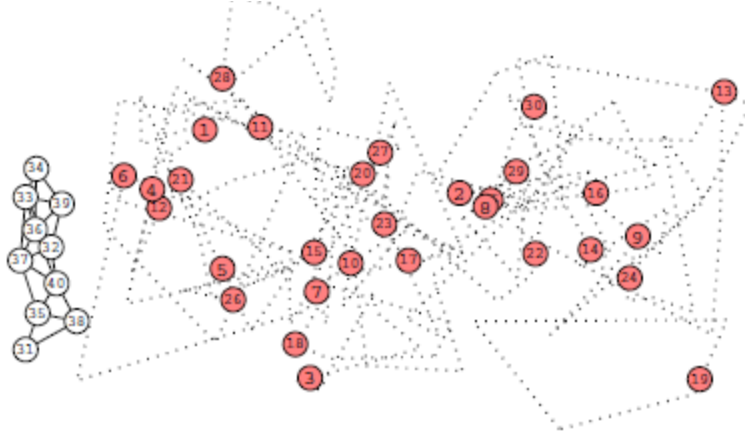


Figure 2.6: A dynamic MANET with sparse and dense sub-networks [22].

The AeroRP [7] protocol is a purposely designed DTN protocol for FANET environments having a sparsity of nodes and extreme mobility. The trajectory prediction *Time-To-Intercept* metric is AeroRP's main DTN tool. Routing protocols that are designed for high-mobility environments often function using *opportunistic*, prediction and trajectory-oriented schemes [87]. The authors in [87] described an *artificial neural network* location prediction DTN method to aid multimedia streaming applications for mobile users. GeoDTN+Nav [31] protocol that we describe later in section 2.2.4 is a hybrid geographic method incorporating greedy forwarding, *void* recovery, and DTN routing.

2.1.4 Summary

The dynamics of MANET routing contexts have spurred hybrid protocol designs that combine the proactive, reactive, DTN and geographical types. Context-aware adaptive routing is a requirement for effective and efficient protocol performance [43][88] over heterogeneous and varying topologies.

2.2 The geographic routing method

The concept of geographic routing was advanced by the introduction of greedy forwarding algorithms to ad hoc networking in the early 1980s [35][29][42][89]. Geographic routing is now synonymous with greedy forwarding. MANET nodes can obtain position knowledge from location services such as GPS, radio beaconing, etc. The geographic greedy packet forwarding method is attractive in MANET routing due to its lightweight neighbour-tables, distributed and localized determination of paths and scalability in dynamic topologies [35].

2.2.1 Position-based routing

The geographic method is also known as position-based, location-based, or geometric routing [29] [40] [89]. The position-based methods typically attempt to route data packets along the shortest multihop path. There have been non-*distributed* geographical methods, such as the Distance Routing Effect Algorithm for Mobility (DREAM) [90] and the Location-aided Routing (LAR) [91] protocols that operate with the traditional routing tables. But the routing tables that DREAM and LAR use contain entries of node positions rather than IP addresses. Moreover, DREAM and LAR are *geocast* methods, which focus route discovery messages in the direction of target destinations. The design goals of DREAM and LAR protocols are aimed at reducing the high overhead that is associated with routing methods. DREAM's table maintenance costs are minimized by the protocol's mechanism of the 'dynamic routing effect' and the 'distance effect'; whereby a node's frequency of table update requirement for distant nodes is proportionately reduced. From the view of any base node, distant nodes' comparative change of position by angular measure is relatively small. Hence, table updates are minimized for relatively distant nodes. Figure 2.7 illustrates the DREAM and LAR message flooding procedures toward an

expected zone of packet delivery. In LAR, a node floods discovery packets towards a destination's *direction* as described by a minimized rectangular area. The DREAM protocol on its part floods route discovery messages over to a more minimized cone-shaped *request zone*. A LAR source or intermediate node uses the positional and velocity values of the destination node to compute the target zone. The Geographical Routing Algorithm (GRA) [92] and the SpEctrum Aware Routing protocol for Cognitive ad-Hoc networks (SEARCH) [93] methods are other example protocols that work on the principle of developing routing tables based on positions information and destination mapping.

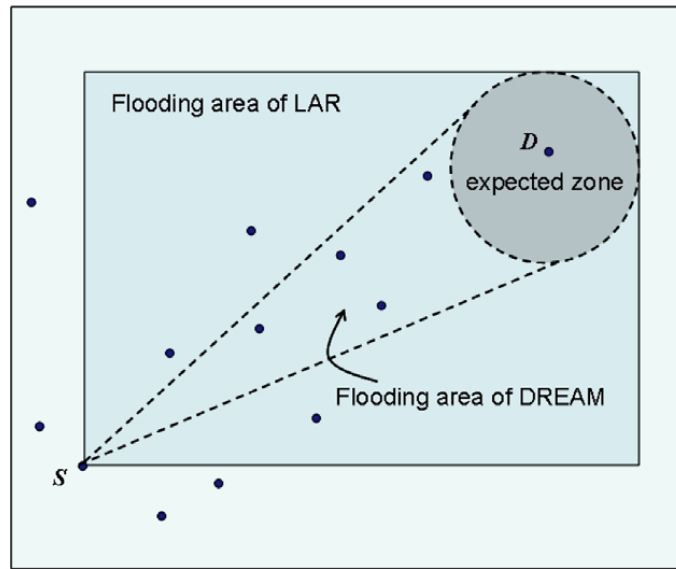


Figure 2.7: Controlled packets flooding of the LAR and DREAM methods [94]

“To route from S to D, with DREAM a current node will forward the packet to all the neighbours’ nodes inside the cone, while with LAR it will forward the packet to all neighbours’ nodes inside the square.”

2.2.2 The geographic greedy forwarding procedure

Greedy forwarding in geographic routing eliminates the predetermination of routes. Nodes only maintain knowledge of neighbours within their respective transmission ranges. Although location

designation replaces IP addressing in geographic greedy routing, the packets transmission procedures of TCP and UDP are yet applicable [95][96]. The geographic protocol copy at each intermediate node performs the greedy forwarding computation using the *distance* and/or *directional* knowledge of its neighbour nodes in relation to the final destination. A greedy forwarding algorithm aims to move data packets closer to the destination position at each stage of multi-hop routing performance. Greedy routing is a unicast method, which for highly mobile environments entails packet *swinging*, i.e., chasing moving target destinations along changing trajectories in real-time [97][46]. Figure 2.8 illustrates a geographic greedy selection process for node u that has packets to forward. The node u is required, within its transmission range, to choose a neighbour $v_i \in N(u)$ or strictly $v_i \in N'(u)$, that is closer in Euclidean distance to the destination d . $N'(u)$ consists of those neighbour nodes that are closer to the destination, i.e., in the *progress* region of a forwarding node. Neighbour nodes that are not closer to the destination than the forwarding node are excluded from greedy choice-making.

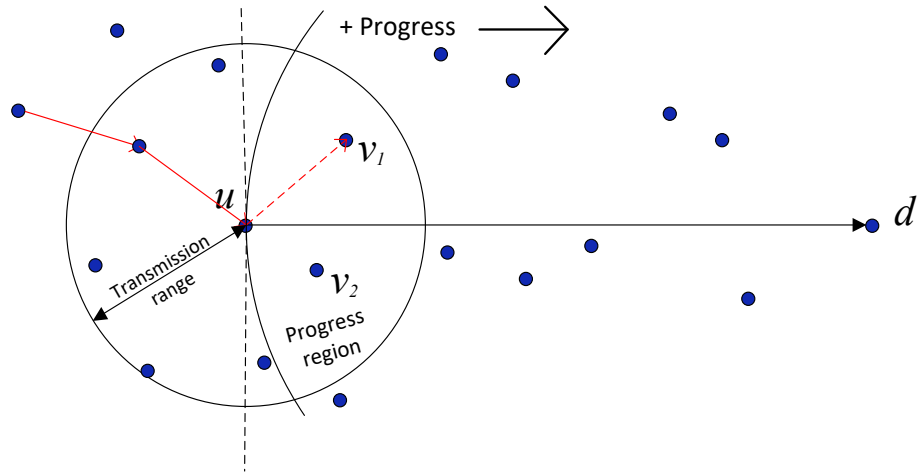


Figure 2.8: A greedy forwarding step

For the forwarding step depicted in Figure 2.8, the greedy algorithm has only two options to choose from, v_1 or v_2 . Continually at each step or hop, a previously selected neighbour v_i becomes the forwarding node u , i.e. $u \leftarrow v_i$, until the destination is reached.

2.2.3 Classic geographic greedy forwarding techniques

The classic greedy forwarding algorithms including MFR, NFP, GREEDY, CR and NC were introduced in section 1.2. Figure 2.8 and the subsequent figures shown in this section (Figures 2.9 to 2.13) respectively illustrate each of the methods. In each figure, the circle depicts the transmission range around node u , while the point labelled d is the destination. In addition to the objective of moving packets closer to the destination at each forwarding step, i.e., *progress* toward d , each greedy algorithm also has its distinct *maximize* or *minimize* criteria that allow it to select a relevant next-hop $v_i \in N'(u)$.

2.2.3.1 Most Forward within Range (MFR)

The Most Forward within Range or Radius (MFR) [77] technique enables the node u to select a neighbour $v_i \in N'(u)$ that achieves *maximal advance* distance along the straight line between u and the destination d as shown in Figure 2.9. The MFR metric is specified by Equation 2.1. MFR attempts to minimize the number of hops that packets traverse between a source and the destination, thereby achieving maximal network *throughput*. However, the nodes implementing MFR often choose next-hops at the transmission range fringes where *links* are intermittent due to weakening in transmitter signals [44][46][98]. MFR has the advantage of being loop-free [99], i.e., does not induce situations where packet traffic gets trapped in a circuitous non-terminating path. The SLURP [76] and the HGR [32] are examples of geographic protocols designed with the MFR technique.

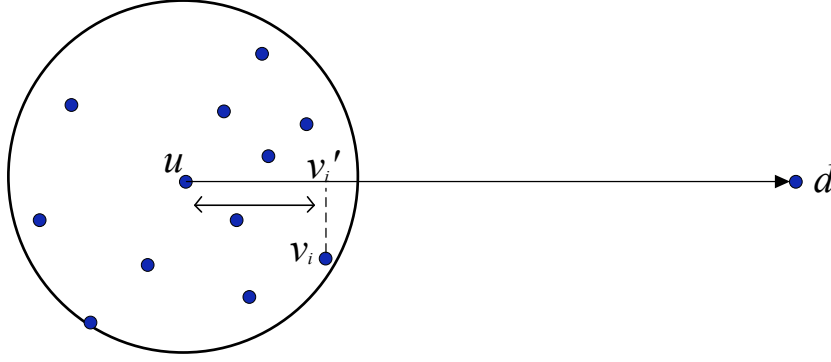


Figure 2.9: Illustration of the MFR technique's next-hop choice-making. MFR chooses a neighbour-node $v_i \in N'(u)$ having the maximum distance from u to the projection v_i' on the line $\overline{ud}$.

$$u_{next-relay}^{(MFR)} \leftarrow \{v_i \mid \max_{dist}\{|\overline{uv_i'}|\}\} \quad (2.1)$$

2.2.3.2 Nearest with Forward Progress (NFP)

The Nearest with Forward Progress (NFP) approach was introduced by [43]. In contrast to the MFR method, the NFP enables the forwarding node u to select a neighbour $v_i \in N'(u)$ that achieves the *minimal advance* distance with reference to the straight line between u and the destination d as shown in Figure 2.10. The NFP metric is specified by Equation 2.2. NFP implements a policy of minimizing nodal power consumption aided by an adjustable transmitting range mechanism. The method supports the reduction of packet collision in dense and highly dynamic environments having multiple transmitting sources.

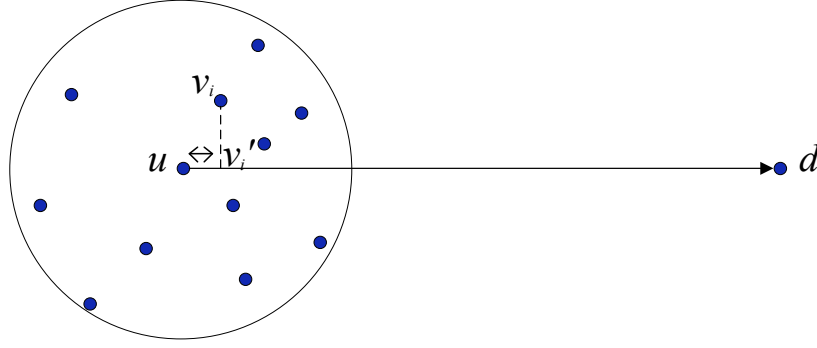


Figure 2.10: Illustration of the NFP technique's next-hop choice-making. NFP chooses a neighbour-node $v_i \in N'(u)$ having the minimum distance from u to the projection v'_i on the line $\overline{ud}$.

$$u_{next-relay}^{(NFP)} \leftarrow \{v_i \mid \min_{dist}\{|\overline{uv'_i}|\}\} \quad (2.2)$$

2.2.3.3 The Greedy Forwarding Strategy (GREEDY)

The Greedy Forwarding Strategy [95] (aka GREEDY or Greedy) enables the forwarding node u to select as next-hop a neighbour $v_i \in N'(u)$ with the minimum Euclidean distance to the destination d as shown in Figure 2.11. The GREEDY metric is specified by Equation 2.3. The lossy link effect that occur at transmission fringes also affects GREEDY as it does MFR [98]. The GREEDY method is found in the design of several geographic routing protocols. Its implementation in geographic protocols is said to be simple [29] and loop-free [100][99]. According to [101] [102], it performs satisfactorily on random graphs, but not so well on the Delaunay specialized graphs.

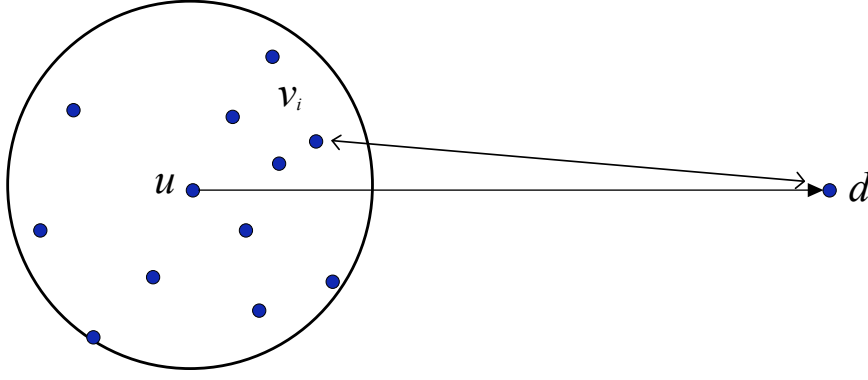


Figure 2.11: Illustration of the GREEDY technique's next-hop choice-making. GREEDY chooses a neighbour-node $v_i \in N'(u)$ having the minimum distance to destination d .

$$u_{next-relay}^{(GREEDY)} \leftarrow \{v_i \mid \min_{dist}\{|\overline{v_i d}|\}\} \quad (2.3)$$

2.2.3.4 Compass Routing (CR)

The Compass Routing (CR) technique [103] is based on the minimal angular deviation criteria. The forwarding node u selects as next-hop a neighbour $v_i \in N'(u)$ that has the minimum angular value between the lines that connects to the neighbour and to the destination, i.e., lines $\overline{uv_i}$ and $\overline{ud}$ as shown in Figure 2.12. The CR metric is specified by Equation 2.4. Note that the original CR by its definition may choose a neighbour that is not necessarily closer to the destination while the minimum angle may be obtuse rather than acute. The CR technique inherently keeps a resulting routing path close to the source-destination line $\overline{ud}$, leading to minimized total travel distance in packets delivery. The CR technique is not loop-free [99].

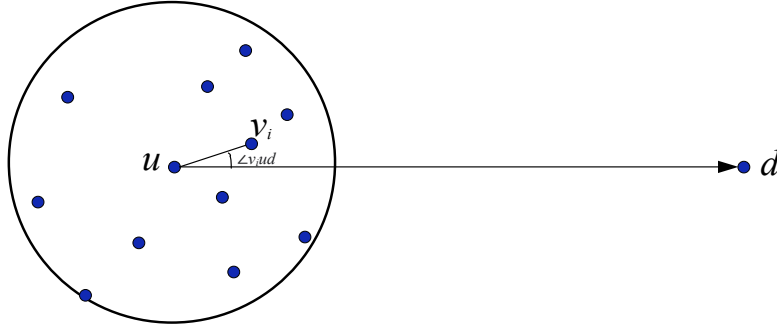


Figure 2.12: Illustration of the CR technique's next-hop choice-making. CR chooses a neighbour-node $v_i \in N'(u)$ having the minimum angle, $\angle v_i u d$, from u and d .

$$u_{next-relay}^{(CR)} \leftarrow \{v_i \mid \min_{angle}\{\angle v_i u d\}\} \quad (2.4)$$

2.2.3.5 Nearest Closer (NC)

The Nearest Closer [100] greedy method enables a node u to select a neighbour $v_i \in N'(u)$ that has the minimum Euclidean distance (i.e., closest) to the forwarding node u as shown in Figure 2.13. The NC metric is specified by Equation 2.5. The NC technique is primarily designed for minimizing nodal transmission power usage. It can equally serve to minimize packet collisions in highly dynamic MANET as does NFP [43]. NC is loop-free [100][99].

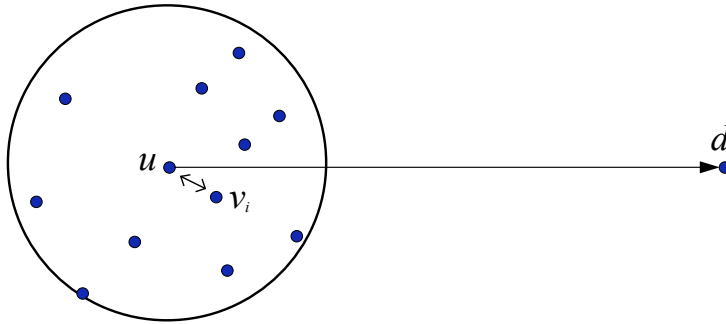


Figure 2.13: Illustration of the NC technique's next-hop choice-making. NC chooses a neighbour-node $v_i \in N'(u)$ having the minimum distance away from u but toward destination d .

$$u_{next-relay}^{(NC)} \leftarrow \{v_i \mid \min_{dist}\{|\overline{uv_i}|\}\} \quad (2.5)$$

2.2.4 Greedy failure recovery methods

Greedy forwarding failure (also called void, local minima, local maxima, dead-end, or obstacle problem) is a major limiting factor to the effectiveness of the geographic routing method [104][30][10][1]. A greedy failure or void occurs when there is no neighbour closer to the destination than a forwarding node itself, such as is demonstrated in Figure 2.14; which shows that there is no neighbour $v_i \in N(u)$ that is closer to d than u itself. Voids may be caused by the physical absence of the required neighbour nodes or their deadness due to energy depletion, blockage by some interfering obstacles, etc. The originally prescribed solution to greedy failure is the *face* or *perimeter* routing that finds a path around the void after which greedy forwarding may be resumed. The perimeter procedure involves the construction of a planar sub-graph, e.g. the Gabriel Graph (GG) or the Relative Neighborhood Graph (RNG) [104][30], over which packets are directed around the void.

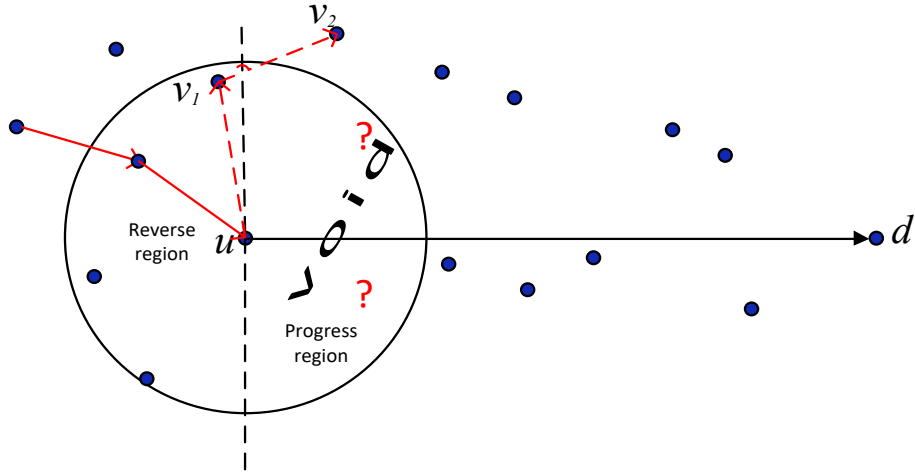


Figure 2.14: An illustration of a greedy failure or void situation.

The node u has no neighbour closer to the destination than itself. A face traversal may take the route around $u \rightarrow v_1 \rightarrow v_2$ to ameliorate the situation before greedy forwarding is resumed toward the destination d .

The classic GPSR [30] and GFG [104] are example protocols that implement the face traversal method. GPSR's graph planarization algorithm constructs the GG or RNG by removing cross-links over the void area to obtain the subgraph [30]. Subsequently, the right-hand rule is applied to route over the subgraph. In Figure 2.14, a viable perimeter routing path is $u \rightarrow v_1 \rightarrow v_2$. The authors of Cross Link Detection Protocol (CLDP) [105] developed a distributed cross-link removal method that is claimed to exhibit improved efficiency in real environments.

The Greedy Other Adaptive Face Routing (GOAFR+) protocol [20] is a design for achieving an asymptotically optimal face or planar path. Nonetheless, the planar graph construction approach also fails if applicable nodal links get disconnected during the planarization procedure [106], making the approach unsuitable for highly dynamic MANETs. The Greedy Perimeter Coordinator Routing (GPCR) protocol [107] prescribes street-aware routing that practically bypasses voids in urban environments. The Small-World Iterative Navigation Greedy (SWING+) protocol [108] and the Small World Topology Aware Routing (SWTAR) [109] methods maintain virtual links for bypassing voids or obstacles. The idea of *geographic routing without location information* in [110] which uses local connectivity information to form virtual coordinates is similar to those of SWING+ and SWTAR. The Geographic Distance Routing (GEDIR) method [99], the Angular Routing Protocol (ARP) [48], and the Ellipsoid protocol [57] all deal with voids by temporarily backtracking packets from the progress forwarding region to the reverse region (see Figure 2.14) where a suitable neighbour could be chosen to initiate recovery routing along alternative direction or path. The ARP authors claim that it performs voids recovery better than the GPSR [30] protocol. The GEDIR is equivalent to a GREEDY method that may resort to choosing next-hops in the reverse regions. However, GEDIR is designed to fail if packets cross an edge twice in succession. Its variant GEDIR+FACE [104] incorporates the planar recovery routing method. The Ellipsoid [57] and the SWING [108] methods are primarily designed to function in 3-D environments.

Implementing the perimeter and planar graph routing procedure in 3-D environments is more challenging. The Gabriel Graph and Relative Neighbourhood Graph that are derivable for recovery routing in the 2-D plane are not directly applicable in 3-D [111]. Nevertheless, by forming a Delaunay triangulation over the underlying graph, the Greedy-Hull-Greedy (GHG) [112], Multi-hop Delaunay Triangulation (MDT) [2], and Energy-efficient restricted 3D greedy routing (ERGrd) [113] demonstrated *guaranteed* packet deliveries in 3-D. Delivery is guaranteed if at least a path exists from a source to the intended destination. Another approach, the Greedy Distributed Spanning Tree Routing (GDSTR-3D) [114] involves recourse to some form of *tree* translation and traversal to achieve guaranteed delivery. In [115][116], the LAR [91] protocol's approach is used in forming an 'above and below' choice-making called 3DABLAR that engages 'partial flooding' to recover from greedy failures. The Self-adaptive On-demand Geographic Routing (SOGR) protocol [61] (a 2-D design) diverts to the *reactive* routing approach to recover from greedy failures. Other than the absence of a suitable neighbour to effect greedy forwarding performance, a network may sustain partitions (as depicted in Figure 2.5) that require a delay-tolerant network method (DTN) type of routing [84]. We have reviewed the DTN approach in section 2.1.3. GeoDTN+Nav [31] is an example hybrid geographic protocol incorporating the DTN method. Figure 2.15 depicts the architecture of GeoDTN+Nav showing the Greedy, the Perimeter, and the DTN routing modules.

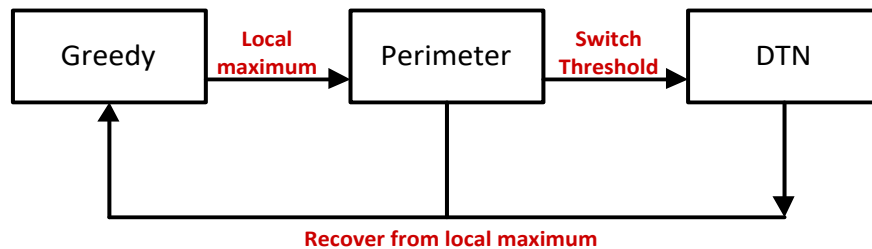
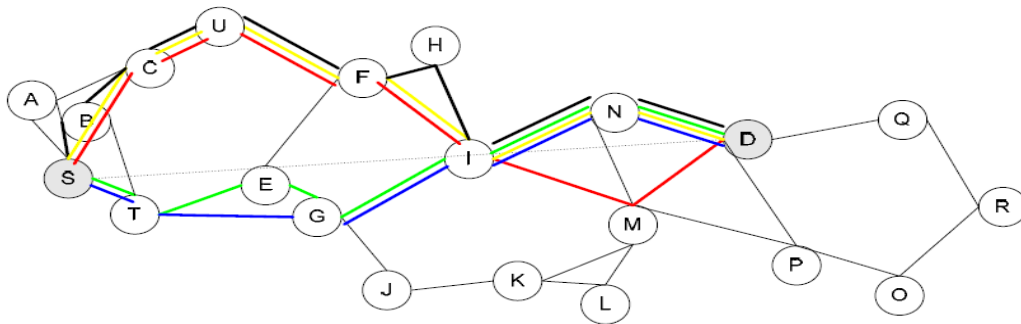


Figure 2.15: GeoDTN+Nav architecture [31]

2.2.5 Summary

The DREAM [90] and the LAR [91] are early examples of position-based or geographic protocols that use node locational values as routing-table entries. However, routing tables quickly become stale in highly dynamic MANETs, such as VANET and FANET. The geographic greedy packet forwarding approach, which incorporates per-hop routing and lightweight neighbour tables, is more suitable for highly mobile environments. The classic greedy metrics, as well as other variants described in section 2.3 below, perform the selection of packets' next-hop from among the neighbours of a forwarding node. The greedy algorithms use distance and directional proximity data in selecting a next-hop node. A next-hop choice must be closer to the destination than the forwarding node itself, otherwise, greedy forwarding fails. Figure 2.16 illustrates greedy selections from S to D by different algorithms. Note that the ELLIPSOID [57] method would also choose its relays along the Shortest Path route in this scenario.



GREEDY: S-C-U-F-I-N-D MFR: S-C-U-F-I-M-D CR: S-T-E-G-I-N-D
 NC: S-B-C-U-F-H-I-N-D NFP: S-T-E-G-J-K-L-M-D Shortest Path: S-T-G-I-N-D

Figure 2.16: Path selections with different greedy algorithms [42]

Recovery from greedy failures is addressed by designs such as the perimeter or face routing method; while packet forwarding across partitions is addressed with DTN routing [117].

2.3 Supplementary variants of the geographic greedy forwarding method

In addition to the classic greedy packet forwarding approaches, there are other variants and enhancements which are outlined below.

2.3.1 A*-metric-like forwarding methods

The A* algorithm is a popular path-finding method with its characteristic ‘open’ and ‘closed’ data queueing design [58][118]. Figure 2.17 and Equation 2.6 illustrate the A* metric. Aside from the queueing approach, the algorithm’s metric $f = g + h$ has the same structure as those of the greedy methods described in Figures 2.9 to 2.13 and Equations 2.1 to 2.5 respectively. Greedy methods use distributed neighbour-tables rather than queues. Generally, the use of global data structures is incompatible with distributed implementations on non-static highly mobile nodes. Geographic Routing Algorithm (GRA) [92] is an example protocol that explicitly prescribes the A* metric (without queues) in its design. GRA functions like the DREAM [90] and the LAR [91] methods’ in its use of routing tables with position entries.

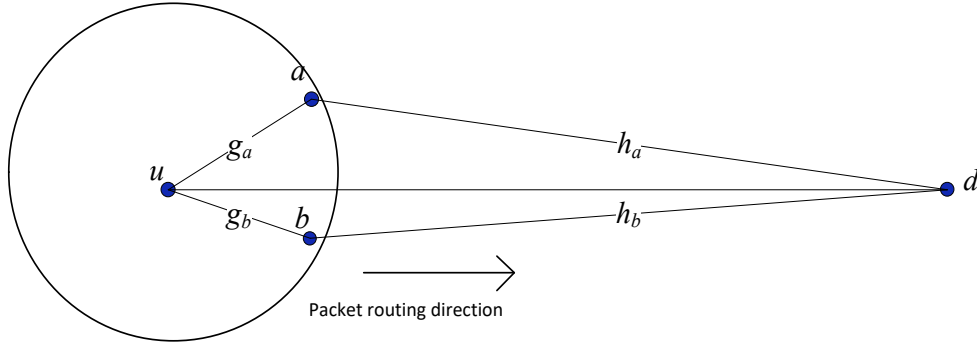


Figure 2.17: The A* algorithm next-hop selection metric for routing path determination. The node u has to select a neighbour as the next-hop in path planning. In comparison to node a , node b is preferred as the next-hop because it reflects the ‘least’ $f = g + h$ value, i.e., $f(g_b + h_b) \leq f(g_a + h_a)$.

$$u_{next-relay}^{(A*)} \leftarrow \{v_i \mid \min_{dist}\{|\overline{uv_i}| + |\overline{v_id}|\}\}. \quad (2.6)$$

The Ellipsoid [57] geographic packet routing protocol also uses the metric of Equation 2.6 based on the neighbour-table approach for greedy forwarding performances. Ellipsoid is designed for 3-D routing, with the prime objective of seamless recoveries from greedy forwarding failures. For normal forwarding, if a node using the Ellipsoid protocol has no next-hop candidate neighbour in the transmission progress region; it simply chooses one in the reverse region, which enables it to subsequently bypass the void or failure area (see Figure 2.14). In [119], the distributed Depth First Search (DFS) greedy forwarding method implicitly uses the Equation 2.6 form of distance criteria to explore child nodes. Note that the traditional DFS tree-mapping method considers the number of nodal links to be traversed in its path-finding procedure and not necessarily the length measures [120][39] as with the A* algorithm. The authors in [102] state that substituting the GREEDY part in GREEDY-COMPASS with the Equation 2.6 metric, i.e. the A* metric, forms the GREEDY-COMPASS-2 that demonstrates improved packet forwarding in Euclidean distance values. Also, note that geographic packet routing designers now refer to the A* metric as ‘Ellipsoid’ based on the Ellipsoid [57] protocol. In this thesis, we refer to it as the ELLIPSOID greedy forwarding method.

2.3.2 The k -hop reach in greedy forwarding

Each geographic forwarding node maintains a topology knowledge comprising of its immediate neighbours’ positional coordinates. However, involvement beyond the 1-hop level may be implemented for improving packet forwarding performance [121][122][123]. A k -hop, $k \geq 2$, forwarding node looks ahead to neighbours of neighbours up to the order of k in determining next-hop choices. Note that the n -hop or the ∞ -hop forwarding is equivalent to an end-to-end connectivity scheme such as in table-driven methods [123]. It has been shown that the 2-hop level optimally improves protocol performance [99][114][124][125], beyond which procedural

complexities and overhead costs grow disproportionately [35][123][126]. The 2-hop greedy forwarding variants, including GEDIR-2, DIR-2 (DIR is equivalent to CR [103]), and MFR-2 have been shown to exhibit better packet delivery success rates compared to their respective base types [99].

2.3.3 The 3-D greedy forwarding

The geographic greedy forwarding method is applicable in both the 2-D and the 3-D modes [111][113][127][128]. While 2-D packet routing is applicable on terrestrial plains, the 3-D design is especially required for airborne and underwater environments [94][129][130]. The Ellipsoid [57] protocol, which we described in section 2.3.1, is purposely defined for greedy forwarding in 3-D environments. Figure 2.18 illustrates the 3-D forwarding step for the ELLIPSOID, CR [103], GREEDY [95], and MFR [77] methods.

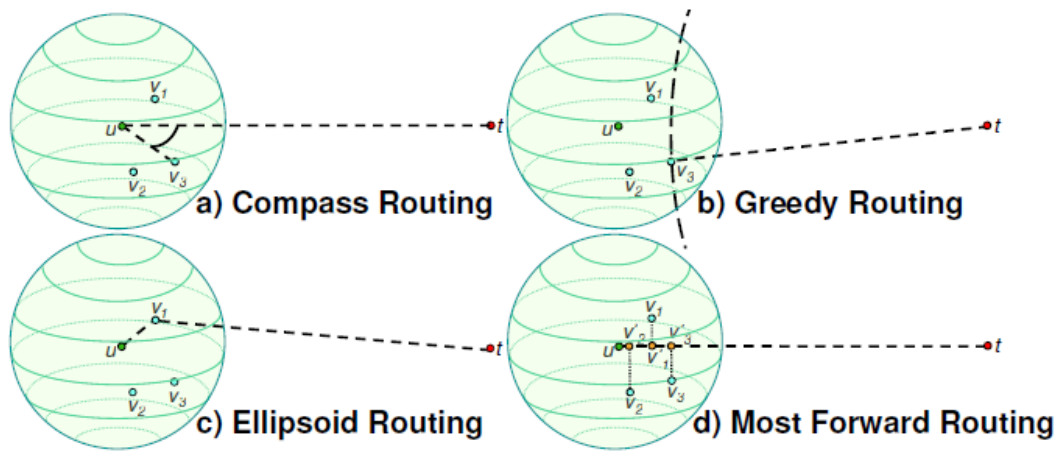


Figure 2.18: Illustrations of greedy forwarding in 3-D. [111]

The 2-D type of planarization for perimeter routing, to recover from greedy failures, is found impracticable in 3-D [122] [131]. Nevertheless, a 3-D to 2-D *projective face* planar approach to perimeter routing is shown to improve successful packet deliveries [111]. The *Delaunay triangulation* is a prescribed scheme for routing around voids in 3-D [112][2][113]. A Multi-hop Delaunay Triangulation (MDT) protocol is shown to guarantee delivery in d -dimensional ($d > 3$) spaces [2].

2.3.4 Randomized greedy forwarding

Randomized greedy forwarding entails the introduction of some *randomness* criteria to the process of choosing a next-relay node [94]. An early approach, the *Random Progress Method* (RPM) [132][94][1] chooses with equal probability from among all the neighbour nodes in a *progress* transmission region (Figure 2.8). RPM aims at randomly distributing packets' forwarding load among all eligible nodes rather than depleting the resources of a frequently chosen few. The method of *Random Compass* [101] selects with equal probability one out of two CR [103] deterministically chosen nodes. The two CR choices would have been made from the opposite sides of the line that links the currently forwarding node with the destination. Its authors show that the Random Compass method, unlike its base CR method, is not *defeated* by any triangulation or convex subdivision [102], i.e., would get out of a loop. The randomized greedy methods of *RCOMPASS*, *WEIGHTEDRCOMPASS*, *BEST2COMPASS*, and *BEST2GREEDY* are reported to improve packet delivery rates [133] in comparison to their related classic deterministic constituents; albeit in some cases with an increase in the *stretch factor*, i.e., the number of hops traversed to a destination in comparison to the shortest possible. Some other forms of the randomized methods involve delineating the progress forwarding region into sectors, from which

a next-hop node is then chosen by pre-set probability attempts. The sectorial random types include the GREEDYINSECTOR, RANDOMINSECTOR, and FARINSECTOR [133]. Figure 2.19 depicts the FARINSECTOR-choosing process going through the periphery, the wings, and lastly the core, in that order to find the most eligible next-hop [133]. The 3-D versions of the foregoing, tagged ‘AB3D’, i.e., Above and Beneath 3D have been proposed [94]. An AB3D random choice can be performed using any of the classic methods over 3-D environments [94]. For example, we can implement AB3D(MFR), AB3D(GREEDY), etc. Another example 3-D method is the *Greedy Random Greedy* (GRG) [131] protocol, which also performs greedy failure recovery through a *random walk* upon the underlying unit ball graph of the forwarding node. The randomized approaches have appeared to minimize packet dropping caused by greedy failures [30][133].

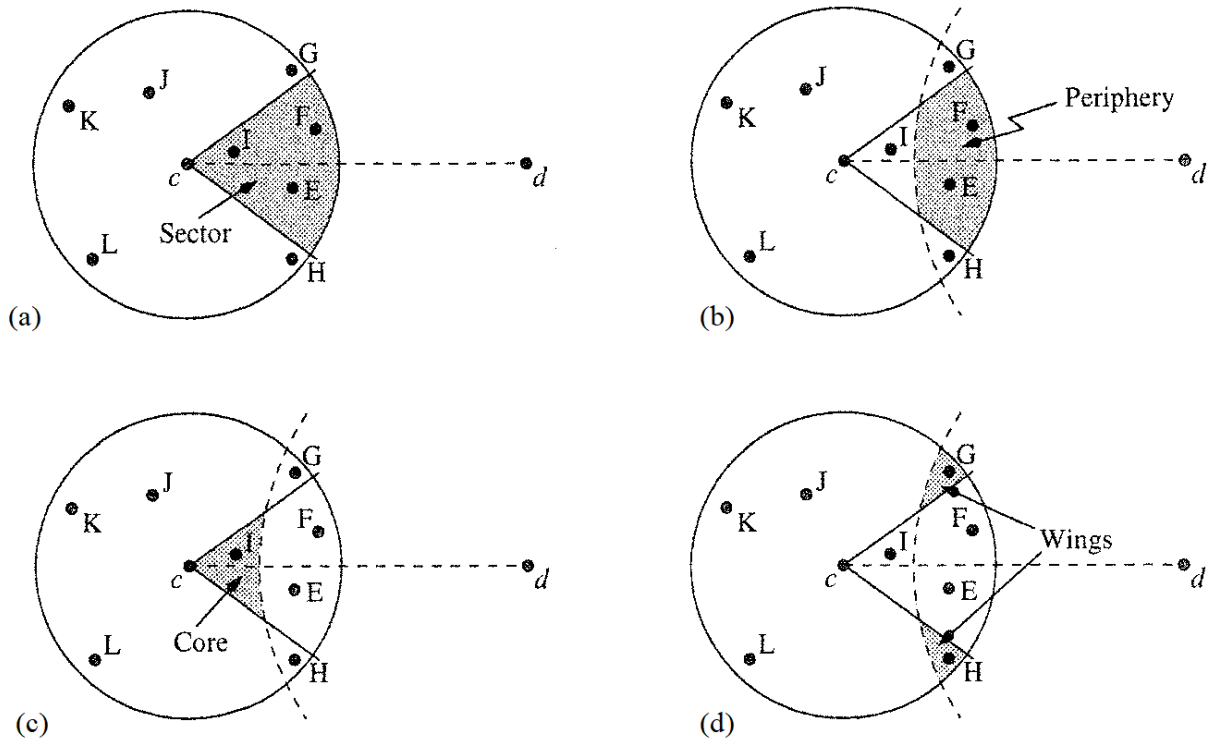


Figure 2.19 (a) – (d): Illustration of the FARINSECTOR random method [133]. FARINSECTOR chooses a neighbour-node from either the *sector*, *periphery*, *core*, or *wings*, in that order.

2.3.5 Beacon-less and opportunistic greedy forwarding

The geographic packet routing's *beacon-less* approach aims at eliminating the overhead of neighbour-beaconing that is associated with neighbour-table maintenance [134][135][136][137]. The neighbourhood of a node is the network area it perceives which covers at least 1-hop radius. Conventionally, a node maintains its neighbour-table through periodic beacon broadcasts. However, with the beacon-less method, a node only broadcasts discovery HELLO messages or data packets as at when forwarding is to be performed. Neighbour nodes within the 1-hop range that hear a HELLO message shall *contend* for the role of next-hop based on their respectively computed measures of distance to the target destination. Accordingly, the neighbour that is closest to the destination succeeds as the packets' next-relay. Beacon-less protocols include Geographic Random Forwarding (GeRaF) [134][138], Beacon-less Routing (BLR) [135], Beacon-less On-Demand Strategy (BOSS) [139], and the Contention-Based Forwarding (CBF) [140]. CBF is said to be entirely *beaconless* as its mechanism, also attributable to the *opportunistic forwarding* approach [141][142], enables the contending nodes to take full responsibility for ceding the next-hop choice to one of them. In opportunistic routing, the *broadcast* characteristic of wireless transmission is leveraged by passively listening nodes to pre-enlist potential forwarders toward overheard destinations, which reduces fresh discovery routines. The authors in [35] declare beacon-less greedy forwarding to be overhead-efficient in VANET routing. Nevertheless, the IEEE 802.11p and the DSRC vehicular standards make beaconing mandatory for continuous neighbourhood awareness in VANETs [143].

2.3.6 The composite greedy metrics

The CR [103] method has been combined with other methods as composite trade-off greedy metrics. For the hybrid GREEDY-COMPASS [102] method, the CR metric first shortlists two

candidate nodes from the *clockwise* and the *counter-clockwise* sides of the line between the currently forwarding node u and the destination d , i.e., line $\overline{ud}$ in Figure 2.20. Subsequently, out of the two shortlisted, the GREEDY metric part selects the node that is closer to the destination as the next-hop. The authors of GREEDY-COMPASS showed that the method guarantees packet delivery in any connected network. In GREEDY-COMPASS-2 [102], the GREEDY base method is substituted with the A* or ELLIPSOID metric, which is said to minimize packet travel distance and performs faster delivery.

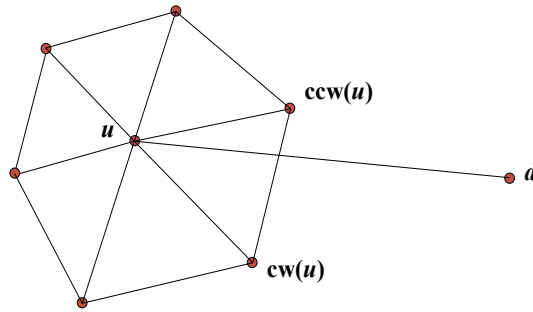


Figure 2.20: An instance of the *GREEDY-COMPASS* hybrid approach [102]

The Hybrid Geographic Routing (HGR) [32] amalgamates MFR [77] and CR metrics based on suitably determined trade-off ratios. HGR is designed as a QoS composite metric for WSN environments where minimizing nodal energy consumption is important. The MFR-part causes nodes to select the next-relay neighbour at the maximal progress distance toward the target destination, which though incurs high transmission costs. The CR-part causes nodes to select the next-relay that contributes to the minimization of the total distance covered in packet travel. Thus, the trade-off of the two corresponds to a balance between the energy and the delay budgets.

2.3.7 Summary

The variants and enhancements of the standard greedy algorithms have been designed to address various packet forwarding challenges in geographic routing. For example, the methods associable with the A* algorithm metric, such as the Ellipsoid [57], aka ELLIPSOID metric, have the attribute of shortest path routing. The randomized greedy types support load-sharing among eligible next-hop nodes. The 2-hop neighbourhood greedy forwarding, but not beyond, improves successful packet delivery at reasonable costs. The 3-D greedy designs address pertinent environments such as airborne or seaborne routing. The beaconless approaches have the goal of minimizing packets' broadcast overhead. The composite greedy metric types, such as the GREEDY-COMPASS and the HGR, leverage the advantages of the composed parts. The utility of all these greedy forwarding variants depends on the routing environment and the application requirements.

2.4 Context-aware geographic routing

Context-awareness implies that a node or a routing protocol knows the current conditions of a MANET environment which may be utilized to enhance efficient packet delivery [14][21][144]. Context attributes include node density, link quality, node energy level, transmission power, node speeds, etc. The context-aware geographic protocol designs are classifiable into two groups namely: the *cost-aware* [88][98][100][145][146][147][148][149] and the *topology-aware* [150][151] types.

2.4.1 Cost-aware geographic greedy metrics

Most geographic greedy forwarding methods that perform with cost-awareness are designed for WSN environments where sensor nodes operate with limited battery or energy lifetime [88][98][100]. Examples are the *cost-to-progress ratio (CPR)* [145] and the *progress-to-cost ratio* [146] which are trade-off metrics designed to minimize cost in greedy forwarding transmissions. With *CPR*-routing, every route establishment activity through a node as well as all neighbourhood communication engagements constitutes its *load* factor. Each node accumulates the load factor value during the course of network operations. The load factor, as well as other node attributes such as expected hopcount, delay, power_reluctance, packet error rate, transmission power [145][146], etc., may be cast as *Cost*. Hence, progress forwarding from node u to a $v_i \in N(u)$, $i = 1, 2, \dots, |N(u)|$, is evaluated using the *CPR* minimization function [145]:

$$F_{CPR}(v_i) = \frac{Cost(\overline{uv_i})}{|\overline{ud}| - |\overline{v_id}|}, \quad (2.7)$$

where $|\overline{ud}|$ and $|\overline{v_id}|$ represent the Euclidean distance between the respective nodes, as depicted in Figure 2.21.

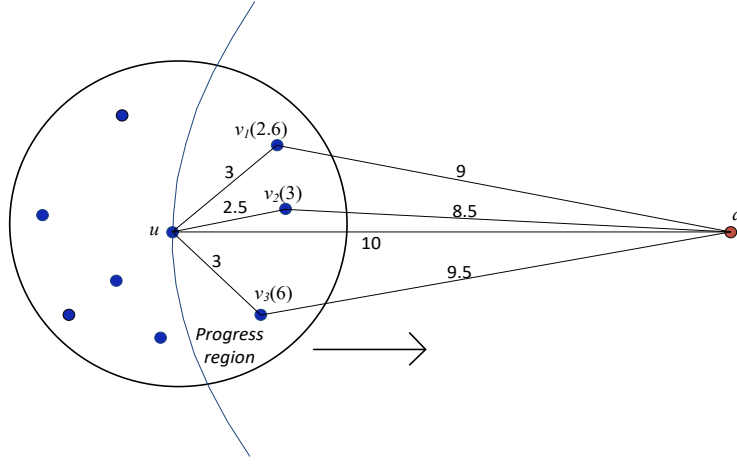


Figure 2.21: Example $F_{CPR}(v_i)$ cost-aware evaluation for a greedy next-hop selection [145].

Distance values are shown on the links, while *Cost* values are in the brackets.

In Figure 2.21, the node u that has packets to forward towards the destination d considers the weight edges of each eligible neighbour. In this example, v_2 shall be the next-hop choice since $F_{CPR}(v_2) = 2.0$, i.e., $\frac{3}{10-8.5}$, is the minimum compared to 2.6 and 12.0 values of $F_{CPR}(v_1)$ and $F_{CPR}(v_3)$ respectively¹.

The *Normalized Advance (NADV)* [44][146] progress-to-cost method emphasizes the selection of next-hops with maximal greedy progress *advance*, ADV , per unit *cost*. It has the objective of maximizing the per-hop length of the packet forwarding distance, while also minimizing retransmissions and packets dropping due to *grey* reception spots. A node that has packets to forward selects a next-hop neighbour $v_i \in N(u)$ using the *maximize* function [146]:

$$NADV(v_i) = \frac{ADV(v_i)}{Cost(v_i)} = \frac{D(s) - D(v_i)}{Cost(v_i)} \quad (2.8)$$

¹ Note that our use of the definition (2.7) yielded a result that is different from that of the authors in [145].

with $D(s)$ and $D(v_i)$ respectively corresponding to $|\overline{ud}|$ and $|\overline{v_id}|$ of Figure 2.21. Thus, Equation 2.8 may be rewritten as:

$$\begin{aligned} NADV(v_i) &= \frac{|\overline{ud}| - |\overline{v_id}|}{Cost(uv_i)} \\ &= \frac{1}{F_{CPR}(v_i)} \end{aligned} \quad (2.9)$$

which is a reciprocal of Equation 2.7 (i.e., the *CPR* metric). In [147][148][100][149][152], numerous greedy forwarding trade-off metrics are defined based on *costs* such as energy or power consumption, *bit error rate (BER)*, delay, hopcount, and *Packets Reception Rate (PRR)*. The cost-aware metrics of *Power Progress*, *Projection Power Progress*, etc., are defined in [147], which generally implement the greedy forwarding distance and transmission power ratios. The Adaptive Load-Balancing Algorithm (ALBA) [153], HOle-BYpassing routing with Context-AwareNess (HOBYSAN) [154], and Energy-aware Dual-path Geographic Routing (EDGR) [155] are variously designed to isolate nodes that are found to be low in functional power or energy. Nodes with very low transmission power could cause *holes* that result in greedy forwarding failures [69] [144].

2.4.2 Topology awareness in geographical packets forwarding

The aggregation of *links* that connect one node to the other in a network describes its topology. *Link length* and *link duration* are examples of topology-aware metrics. The *link-aware* metric of $PRR \times d$, i.e., packets reception rate against distance, is defined in [88][98] for *lossy* WSN environments. We discuss in sections 2.5 and 2.6 the factors that influence links, and accordingly the network topology. Spectral graph theory (SGT) methods of a distributed kind have been proposed to make nodes aware of network topologies, such as *clusters* and *critical links* [150][121][156]. Nevertheless, the computations involved with the SGT approaches would be

challenging to implement in high-mobility environments that exhibit rapid position changes. In [19], a ‘topology change aware-based routing protocol choosing scheme’ (TARCS) is described based on the integration of the *relative velocity*, *angular direction*, and *distance* measures among nodes. The TARCS’s Topology Change Degree (TDU) metric is designed for awareness of mobility models, including random way point and two others, that underlie a MANET environment; based on which appropriate methods are adaptively assigned. The study in [56] demonstrates the adaptive performance potential of the Depth First Search (DFS) path navigation protocol which can differentiate whether an underlying network graph is the *Delaunay* or the *random graph* type. The routing method can make the distinction by sampling the link or *arc length* histogram distribution patterns. Consequently, the most suitable protocol variant, either DFS, Positional-DFS, or Directional-DFS could be assigned to perform future navigation tasks for each topology type. The authors in [102] corroborated the findings in [56], and further clarify that the P-DFS (or GREEDY [95]) and the D-DFS (or CR [103]) outperform each other in the distance covered and the hopcount measures of packets delivery depending on the underlying graph type. The design of networks with topologies that support effective greedy forwarding has been proposed., i.e., greedy-embedding [157][158][159][160][161]. *Stretch factor* or *dilation* (hopcount to shortest path ratio) and load balancing are some of the issues dealt with in the embedding designs.

2.4.3 Summary

Context-aware routing and forwarding schemes have been designed by researchers for adaptive and efficient routing performance in different MANET environments. The cost-aware types are generally concerned with nodal energy or power conservation; while the link-aware methods are concerned with the quality of connectedness in making forwarding choices. Issues of links and topology configurations are further discussed in sections 2.5 and 2.6.

2.5 Impact of MANET topologies on protocol performance

The topological states that a MANET assume correlates to a packet routing protocol's effectiveness in it [70][162][163][164][165]. A protocol that routes packets excellently in one environment may perform dismally in another. Link and topology perturbations cause substantial degradation in protocol performance. The MANET factors that influence routing protocols' performance are broadly classifiable as (i) the density and spatial distribution of nodes, (ii) relative mobility rates, and (iii) wireless medium vagaries [3][43][68].

2.5.1 Effect of node density and spatial distribution on packet routing and forwarding

The density and spatial distribution of nodes fluctuate highly in certain dynamic MANETs such as the VANET. Spatial distribution determines the level of connectedness between adjacent node pairs and *connectivity probability* for packet routing performances [166][167][168]. Figure 2.22 illustrates a network with dense, less dense, and sparse clusters. High node density coupled with multiple packets transmitting sources exacerbates wireless interferences and packet dropping. High node density in VANETs has been shown to cause transmission range degradation of up to 90% and 50-70% respectively in a simulation study [169] and a testbed study [170]. On the other hand, node sparsity diminishes network cohesion which is exhibited by easily breaking links, isolation of nodes, and network partitions [167]; resulting in packet re-transmissions and transmission failures.

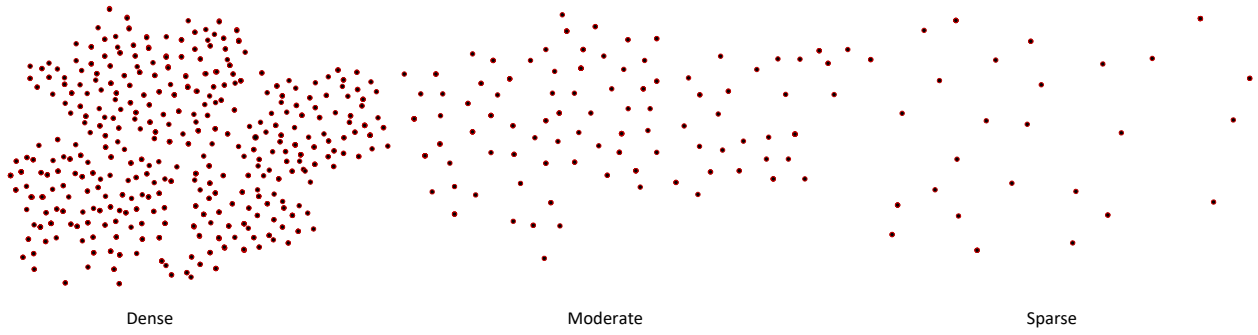


Figure 2.22: An illustration of node density variations

The relationship between node density and connectivity in random graphs has been studied. A k -connectivity is said to exist or be achieved when every node in the network has a minimum of value k as the *node degree*, i.e., the number of neighbours that are linked to a node. Modelling k -connected networks, based on *nearest neighbour distance* analytical expressions is shown in [166]. The authors in [77] and [171] have determined that the optimum functional connectivity in a MANET is achieved when k is about 7 to 8. The AODV routing protocol's performance study in [171] showed that with increasing node degree, the PDR increases only up to a threshold, after which it declines. The work in [172] shows a similar performance decline pattern for Greedy-based Backup Routing (GBR), HARP [50], etc. In a 3-D environment comparison of the geographic greedy routing methods of GREEDY, CR, MFR and Ellipsoid, it was shown that the PDR performance of the four methods generally declines as network node size is increased from 50 to 150 [65]. Although, for all four methods, the PDR at 200 node size improved slightly in comparison to the 150 size measures. In another work [98], it was shown that the GREEDY method and its energy-efficient hybrids, performed with *monotonically* increasing PDR over a range of 25 to 200 node sizes. The apparent disparity in the PDR performance patterns of the works in [65] and [98] can be attributable to differences in the settings of simulation parameters such as packet sources and mobility rates.

2.5.2 The effect of node speed and mobility patterns on packet routing and forwarding

The characteristic of random motion of nodes in dynamic MANET causes its wireless communication links to be transient, intermittent, and of fluctuating quality. Nodes' mobility impact routing protocols' packet delivery performance most significantly in comparison to other MANET context factors [165][70][162][173]. The authors in [70], [162], and [163] showed that the packet routing protocols of DSDV, TORA, DSR and AODV exhibited better PDR performances when the average node motion pause time is high with low mobility. All four protocols demonstrated PDR performance above 98% at the mobility rate of 0-1m/s and 0-900s pause times, i.e., near-static situation. On the other hand, that of DSDV-SQ deteriorated rapidly to about 70% at pause times of less than 300s and speed of 0-20m/s [70]. Nevertheless, the routing overhead increases at the 0-20m/s mobility rates compared to that of 0-1m/s. The authors in [174] and [175] studied the mobility traces data of vehicular and personal digital assistant networks. Their findings showed that after certain threshold values in average speed [174] or motion duration [175], the networks' connectivity deteriorates rapidly. According to [174], for the example of Shanghai city, the average vehicular speed of up to 18-20m/s rarely affects connectivity, beyond which it is destroyed by a power-law decline.

Mobility patterns, including the Random Way Point (RWP), Reference Point Group Model (RPGM), Freeway and Manhattan, were found to differently impact the routing protocols of DSR, DSDV and AODV [24][19][176]. In the case of self-adaptive mobility-oriented routing protocols, the authors in [176] enunciate the importance of assessing the suitability of performance metrics that are used for evaluating different routing environments. They assessed the conventional metrics of *Link State Changes* (LC), *Link Connectivity Duration* (LD), and *Link Stability* (LS = LD/LC) over the mobility models of the RWP and RPGM; which showed that routing protocols' performance and rankings do vary widely across mobility models [176]. Figure 2.23 is a

framework showing the relationship between mobility, connectivity and protocol performance [24]. The framework's authors emphasize the place of the node mobility attribute in the design *building blocks* of routing protocols. The notion of building blocks is akin to protocols' *basic mechanisms* [70] or *routing mechanisms* [162]. Examples of protocol building blocks are the *caching* and *error-handling* mechanisms and policies. For instance, the caching property of the DSR protocol is deemed to be a reason why its throughput performance appears to be highest in comparison to those of AODV and DSDV under conditions of mobility with longer or moderate *link duration* [24] [176]. But the DSR's throughput performance is said to decline rapidly when link durations are very short.

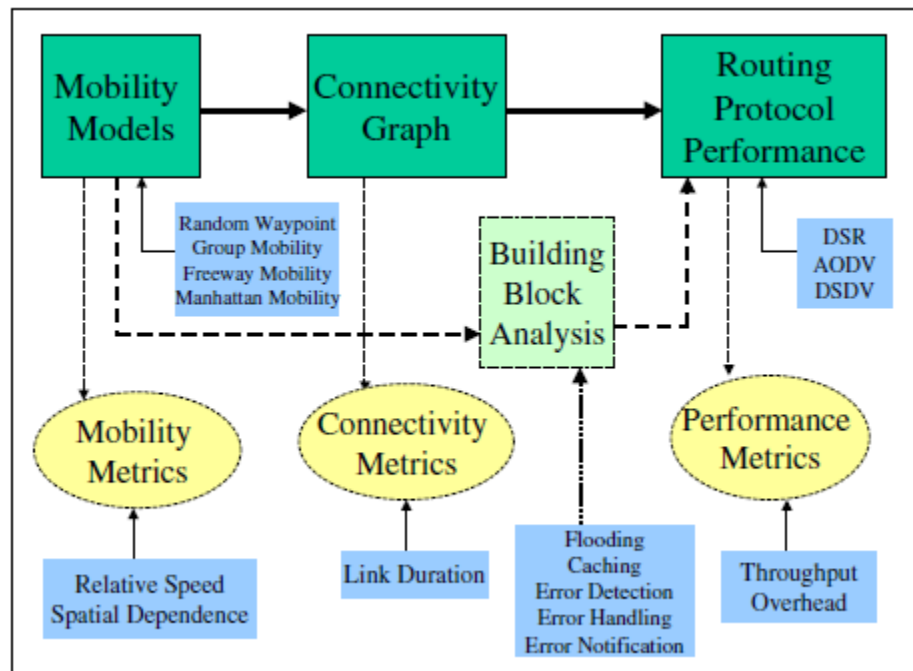


Figure 2.23: The IMPORTANT framework [24].

The importance of using experimental mobility models that appropriately depict each target MANET is stressed in [177]. The authors showed the variations in PDR and other performance

measures generated by different mobility models. For example (Figure 2.24) in comparison to RD, the RWP model has a wider PDR performance margin as node speed is increased between 0 to 20m/s. Nevertheless, the overhead of RWP is comparatively lower than that of RD [177]. The varied performance behaviour of the RWP and RD mobility models is corroborated in [171]. With the RWP model, nodes tend to spasmodically converge and diverge over the simulation area, causing *density waves*, while the RD's topology fluctuation is characteristically minimal [171]. The RWP model is popularly utilized in MANET simulation experiments [24]. In [164], the authors investigated packet routing performance of the NFP, MFR, CR, and GREEDY against mobility patterns including Random, Manhattan (City Grid), City (for WSN), and Small City (a subset of City). They reported that the greedy methods' PDR and hopcount performances vary by up to 25% across different mobility models.

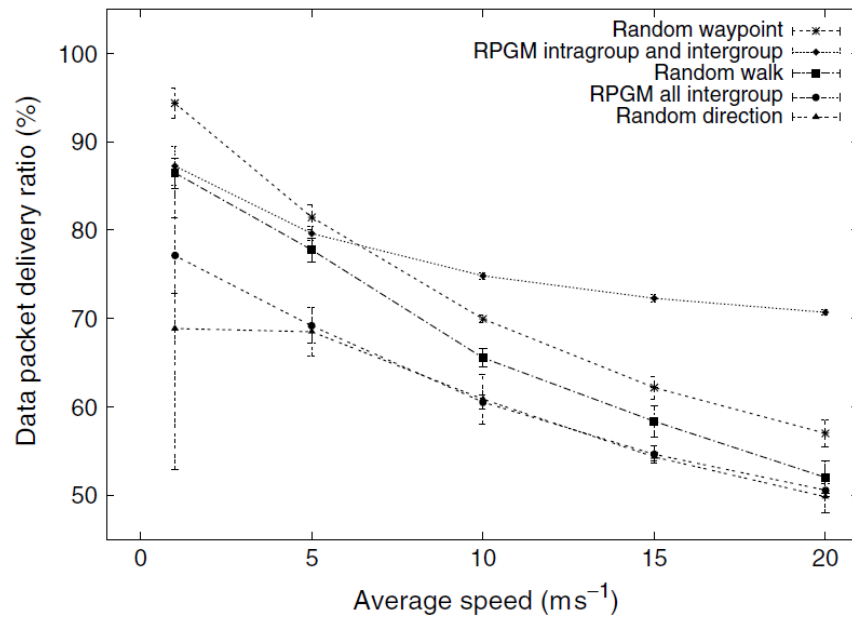


Figure 2.24: Data packet delivery ratio vs speed for different mobility models [177].

2.5.3 The effect of wireless transmission on packet routing and forwarding

When nodes are within effective radio transmission range of each other they can connect and communicate. Figure 2.25 depicts the effect of nodes' transmission power on connectedness and network topology. Connections that are weak cause network partition (Figure 2.25a), while a tightly connected mesh (Figure 2.25b) results in packet collisions and droppings when traffic is high. Figure 2.25c depicts a desirable model of connectedness.

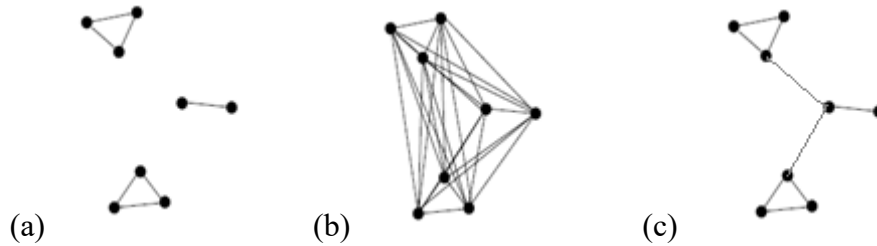


Figure 2.25: Effect of transmission power level on network connectivity (Adapted [178])

The connectedness and connectivity of MANET links are inherently intermittent [3][4][68] [167]. The links are formed by transient radio waves that beam from each node. Generally, radio signals are affected by physical obstacles, transceiver quality, channel congestions, electromagnetic interferences, etc. [68] [179] [180] [178] [88]. The metric of *signal-to-interference-and-noise ratio* ($SINR$, SIR , SNR) [178] is a popular measure of wireless transmission's effectiveness or link quality. In real environments, nodes' omnidirectional radio beaming is non-isotropic in its effectiveness [179], i.e. not of uniform radius around a transmitting node; whereas theoretical and simulation models often depict it as such, e.g., the unit-disk model [35] [89]. In dynamic MANETs, the effective communication range is typically less than a node's actual radio

transmission range. According to [170], a node's normal communication range can reduce by up to 90% in excessive load conditions.

Modelling a realistic wireless environment is challenging [181] due to the several factors to which it is subject. Among other things, wireless transmission modelling should take *lossy links* and the *grey reception area* or the *transitional region* problem into account [35][68][88][98][182][123]. The grey reception area, with high *link variance*, occurs in the middle parts of the transmission radius. Packets' retransmission overhead is high about neighbour nodes that are located within the grey region. Likewise, neighbour nodes that are located at the outer fringes of a transmission radius are apt to experience erratic packet reception. The geographic greedy forwarding techniques of MFR and GREEDY typically perform next-hop selections at the fringes [35]. It is interesting to note that the greedy forwarding techniques of MFR [77] and the NFP [43] evolved as products of studies on the transmission radius problem. Figure 2.26 shows the effect of radio interferences on the connectivity and packet forwarding performance patterns of the MFR, MVR and the NFP methods [43]. The interference factor is highest for the MFR method, especially when the node density is high, while converse is the case with the NFP method.

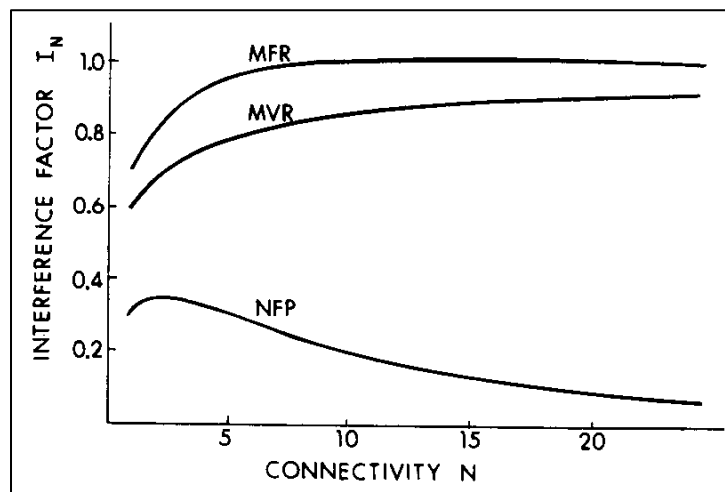


Figure 2.26: Interference and connectivity effects on MFR, MVR and NFP [43]

The intermittencies of wireless links influence *packet reception rates* (*PRR*) and consequently the PDR measures. The link quality metric of *PRR* or its offshoot *expected transmission count* (*ETX*) forms a trade-off with the greedy metrics in some packet forwarding designs [183][98][180]. For example, to achieve the goal of reduction in re-transmission costs, the authors in [183] defined the Greedy Algorithm in Real Environment (GARE) method which trade-off metrics of $\frac{ETX(uv)}{|uv|}$ and $\frac{ETX(uv)}{|ud|-|vd|}$; where u , v , and d respectively represent the node currently having packets to forward, a potential next-hop, and the destination (see Figure 2.8).

2.5.4 Summary

Optimal connectedness and connectivity are requirements for effective MANET multi-hop packet routing and delivery [76][174]. Well-connected MANETs support effective progress-based geographic greedy packet forwarding [178]. Figure 2.27 (derived from Figure 2.23 [24]) shows a broader classification of the metrics relating to MANET contexts, connectivity, and packet routing.

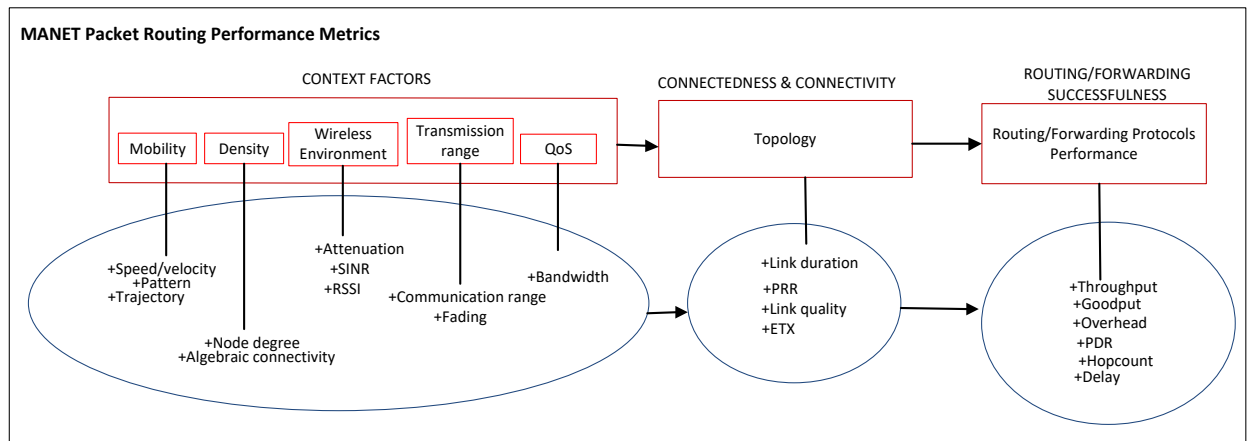


Figure 2.27: Metrics of MANET packet routing (adapted [24])

Numerous attributes characterize MANET environments and topologies, each entity of which can be best served by specific packet routing protocols [150]. The listed metrics in the oval shapes of the figure are not exhaustive. Dynamic regulation of transmitter power has been studied as an aid in controlling wireless transmission ranges for optimal routing performance [43] [178][184][185][186]. Issues of QoS as a context factor, which is veiled in this thesis, can be found in [19][47][87][187][188]. Likewise concerns about packet level issues such as queues, overhead, etc., [70] [69] are not discussed since the applicable environment is uniform in our topology simulations for all forwarding performance comparison runs. The node density and mobility factors are the main influence on MANET topology, which similarly impact routing performance [70][162][173].

2.6 Perceiving network contexts in routing and forwarding

This section presents a review of MANET context knowledge acquisition and syntheses methods for effective environment perception in adaptive packet routing performance.

2.6.1 Context knowledge acquisition for packet routing and greedy forwarding

MANET routing contexts are generally dynamic, characterized by sporadic wireless channel events and variability in the positionings of nodes. Identifying the elements that are relevant to context description, i.e., ontology, and the scheme for monitoring the same is a requirement in the design and implementation of efficient adaptive routing over varying topologies [24][21][19]. Context knowledge over a networking area may be obtained at the *micro* or *macro* dimensions [189]. For example, the node degree measure, i.e., the number of neighbours directly connected to a node, is a micro-level view of topology, while the average of the same over a network segment is at the macro level. Furthermore, context data may be perceived by the depth and breadth criteria. Knowledge depth regarding a specific measure is its granularity. For example, the node degree can be investigated at the extended depths of *probabilistic degree* and *threshold degree* [190]. On the other hand, knowledge breadth may refer to its rich formulation in composite metrics or availability such as in *big data* schemes [191][192]. In [19], the composite metric of *topology change degree*, $TCD_{(u, v)}$ combines the *displacement distance*, *direction*, and *relative velocity* measures between nodes u and v .

The authors in [123] showed that in WSN multi-hop geographic packet routing decisions, a k -hop ($k \geq 2$) forwarding knowledge acquisition improves end-to-end *reachability*. Figure 2.28 depicts attainable packet routing performance improvement with increasing knowledge breadth [84]. The figure also shows the pertinent algorithms, including FC, MED, etc., that respectively operate with

knowledge entities of Zero (i.e., context knowledge is nil), Contacts Summary (i.e., context knowledge is basic, e.g., nodes know number only of neighbours without further information about each one), etc. However, the realism of asymptotic limits on knowledge versus routing or forwarding improvement is suggested by the figure [123].

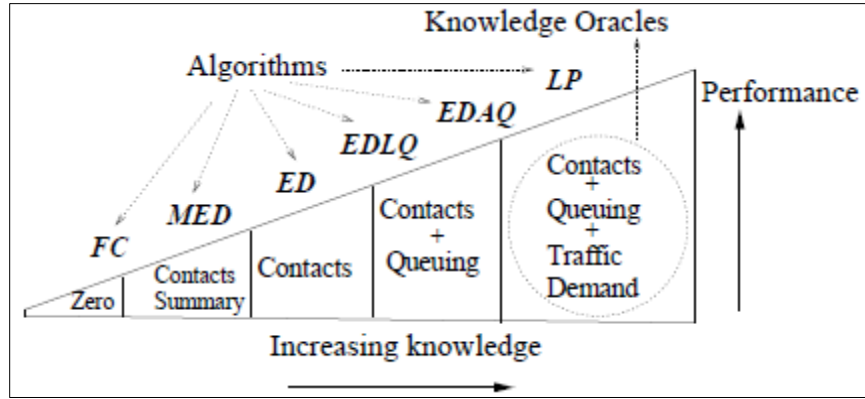


Figure 2.28: Conceptual performance vs knowledge trade-off [84]

2.6.2 Context knowledge syntheses for greedy forwarding and packet routing

The authors in [85][193][194] present example mathematical models of context data synthesizing for network characterization and packet routing decision-making. The Context-Aware Routing (CAR) [85][79] protocol implements a context prediction mechanism for performing the carry-and-forward DTN routing. In CAR, the choice of a suitable next-relay node for packets' *carriage* or *hauling* is rated by a utility function, U , that synthesizes applicable context information. The context attributes are denoted by $X_1, X_2, \dots, X_n$, while the related values are given as $x_1, x_2, \dots, x_n$. The CAR's context attributes are regarded as mutually preferentially independent, i.e., in its *additive* metric [85][79]:

$$U(x_1, x_2, \dots, x_n) = \sum_{i=1}^n U_i(x_i). \quad (2.10)$$

Each attribute's weight of *significance*, w , and its runtime *adaptation*, a , are further incorporated in the maximization function thus,

$$\max\{f(U(x_1, x_2, \dots, x_n)) = \sum_{i=1}^n a_i(x_i) U_i w_i(x_i)\}. \quad (2.11)$$

Each CAR [85][79] node periodically updates its packet *delivery probability* value, which it propagates to the network alongside the conventional proactive routing table updates. A source node with packets to send then selects its routing path based on an appropriate delivery probabilities mapping. Besides, CAR employs the estimation technique of *Kalman filtering* on predictable context metrics.

In contrast to CAR, the Multi-Criteria Context-based Decision (MCCD) [193] protocol introduces a *multiplicative* utility function, as shown by Equation 2.12. The authors of MCCD argue that the additive metric scheme may permit selecting relay nodes having attributes with undesirably low values, e.g., relating to depleted node energy in WSN. This implies that the additive function allows compensation for a poor attribute value when the others are high enough. However, with the multiplicative function approach, an attribute having a low threshold value, likely set to zero, shall simply result in an output that disqualifies the associated node.

$$U = \prod_{i=1}^N ((f_{s,i}(c_i))^{w_i} \quad (2.12)$$

In Equation 2.12, $f_{s,i}$ defines a function for scaling the output in the range $[0, 1]$. C_i is the value of the i^{th} context criterion. The MCCD method allows flexibility in the attributes of interest that a node considers or prioritize. This is similar to the proposal in [195][81] for the human-computer-interaction systems, “scatterbox”, which prescribes dynamic context composition based on current attributes of interest within a *situation*. For example, when a user-device comes to the stationary state, then an attribute such as *velocity* may be disregarded as it is no longer relevant.

The Context-aware Adaptive Opportunistic Routing (CAOR) [194] protocol, in similarity to the MCCD [193] method, asserts the use of a multiplicative function for synthesizing context information. In CAOR, the *Dynamic Forwarding Delay (DFD)* [135] metric, shown by Equation 2.13, is the primary measure by which eligible neighbour nodes may contend to be the next-relay in a packet forwarding session. The successful contender is the node with minimum DFD:

$$DFD(x_1, x_2, \dots, x_n) = DFD_{max} \cdot \prod_{i=1}^N w_i(x_i) \cdot DFD(x_i) \quad (2.13)$$

In Equation 2.13, DFD_{max} describes the maximum allowed delay setting at each node. Context information is represented by x . The trade-off weights, w , are runtime-adaptive and are derived using the Analytic Hierarchy Process (AHP) and Spectral Graph Theory (SGT) procedures. Using AHP, a context's *importance comparison matrix* A_{nn} about the nodes is derived. A then is the SGT basis for deriving the eigenvector W of the required weights in $AW = \lambda W$, where λ is the highest eigenvalue.

2.6.3 The modelling of MANET topology in packet routing and forwarding

MANET topologies are often modelled as graphs consisting of vertices (nodes) and edges (links) [159][196][197][198]. The models also presume bidirectional communication links between MANET nodes, which are thus depicted with *undirected* graphs. Nodes within transmission range of each other are assumed linked, disregarding wireless vagaries. Some recent works have introduced the issue of 'realistic' MANET environments to improve the graph model. Node position change or displacement metrics are also discussed in this section.

2.6.3.1 Graph notations of relative node positions in MANET

In *graph theory*, the graph G or $G(V, E)$ denotes a set of vertices and edges:

$$V = \{v_i\} \quad (2.14)$$

while
$$E = \{e_{i,j} | v_i, v_j \in V; i \neq j\} \quad (2.15)$$

which depicts a network [196]. A graph is directed if the edges are asymmetrical; otherwise, it is undirected with $e_{i,j} = e_{j,i}$. The small example network of Figure 2.29 has 4 undirected edges and 5 vertices. In geographic routing, the set of neighbours that each node u has is defined as:

$$N(u) = \{v_i \mid |\overline{u, v_i}| \leq r\} \quad (2.16)$$

Nodes within transmission radius r of each other are neighbours and can communicate directly [35][89]. In Figure 2.29, the values in brackets state the *node degree*, $d(v)$, i.e., the number of neighbours to which a node is directly connected:

$$d(v) = |N(v)| \quad (2.17)$$

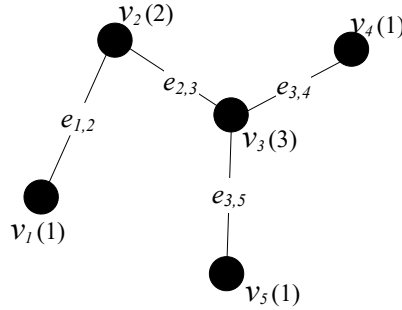


Figure 2.29: Example graph with 5 vertices and 4 edges

The node degree is often used in describing the topology of a graph or network. The average node degree over a *cluster* (aka *component*) or an entire network consisting of n nodes is given by [199]:

$$\begin{aligned} d_{avg} &= \frac{1}{n} \sum_{i=1}^n |N(v_i)| \\ &= \frac{1}{|V|} \sum_{i=1}^{|V|} |N(v_i)| \end{aligned} \quad (2.18)$$

The average node degree reflects the density of a network. A network's density (d) is defined as its number (n) of nodes per unit area (A), given as:

$$d \text{ or } \text{density} = n/A \quad (2.19)$$

In [166], the correlations between the node degree, network density, and nodal transmission range on one hand, and connectivity is described, which led to the prescription of the *geometric graph model*. In Figure 2.28, the connectivity from v_1 to v_5 is describable by the path $e_{1,2}-e_{2,3}-e_{3,5}$. A graph is *connected* if each node has connectivity to any other node. *Random graph* types are often used to depict communication networks. The basic models are the *uniform random* $G(n, m)$ and the *binomial random* $G(n, p)$ types, depending on the *probability* function used for creating the network links [197] [198]. For the $G(n, m)$ model, the number of nodes n and the number of edges m to be inserted are first decided. The insertion algorithm uses an *equally likely* probability $p = 1$ or $p = 0$ to insert the m links out of all the $\binom{n}{2}$ possible choices to form the graph [197]:

$$P(G) = \left(\binom{n}{2} \right)^{-1} \quad (2.20)$$

For the $G(n, p)$ model, the n number of nodes required is also first fixed, together with a chosen probability value p : $0 \leq p \leq 1$. The edge insertion algorithm performs $\binom{n}{2}$ *Bernoulli trials* that connect m , $0 \leq m \leq \binom{n}{2}$, pairs of nodes independently with the probability p . The connectedness of the network will be dense or sparse depending on if p is chosen to be high or low. The $G(n, p)$ graph is formed by [197]:

$$P(G) = p^m (1 - p)^{\binom{n}{2} - m} \quad (2.21)$$

In [200], it was noted that the general random graph models, including $G(n, m)$ and $G(n, p)$, do not consider the dependency that communication networks exhibit, especially concerning transmission radius. Such dependencies imply that if an edge connects a node to a neighbour, then any other

equidistant neighbour should be connected with a high probability. Moreover, real networks exhibit varying node densities in different segments of a network area [17][181]. Consequently, a *random geometric graph* model $G(n, r)$ was proposed in [166]. The $G(n, r)$ model inserts edges between two vertices u and v if the link between them is no longer than a designated transmission radius r , i.e., if $|u-v| \leq r$ (see Equation 2.16). The related unit disk distance model $|\overline{uv}| \leq 1$ is typically employed in the design of geographic routing and forwarding methods [35][89]. To model real-life wireless networks, the authors in [190] proposed a set of distance-based metrics, namely: *nearest neighbour distance distribution*, *probabilistic degree*, *threshold degree*, and *inhomogeneity factor*. The threshold degree and probabilistic degree are based on the *connection probability (CP)* between adjacent nodes, which in turn is dependent on the nearest neighbour distance measures. The probabilistic degree of a node is obtainable by the summation of the *CP* values of its neighbours; while the threshold degree demands that a neighbour node has a ‘good enough’ *CP* in forming a link. Note that *CP* is different from *connectivity probability* [166][167][168] which describes the existence of a multi-hop *path* from one node to another. The inhomogeneity metric assesses non-uniformity in the spatial distribution of nodes. It implements a grid-based approach which locally assesses the deviations of node positions from a set norm of uniform distribution for each subarea. The *inhomogeneity factor* h is given by [190][201]:

$$h := \sum_{j=1}^r w^{1-j} h(2^j) \quad (2.22)$$

where $h(\cdot)$ is a subarea inhomogeneity measure based on an even number segmentation of a grid area, and w is a constant. The inhomogeneity of each subgrid $h(s)$ varies from 0 (uniform distribution) to 1 (all nodes located in a spot).

The subgraph $G' \in G(V(t), E(t))$, at time t , that describes a set of proximally co-existing nodes in a network is often referred to as a *cluster* [165] or a *component* [17][174]:

$$C(t) = \{v_i \cup v_j \mid v_i, v_j \in V, \cap p_{i,j} \neq 0\} \quad (2.23)$$

in which there exists a path p_{ij} that connects $v_i, v_j \in G', i \neq j$; while the link or edge set is described as:

$$E_c(t) = \{e_{i,j} \mid v_i, v_j \in C(t)\} \quad (2.24)$$

Over a period of consideration, connectivity exists among the nodes in a cluster. In their studies on real-life topology characterization, the authors in [165][17][174] found that *isolated* clusters are rife in VANET environments. For a cluster $C(t)$, its *clustering coefficient* $C(t)_k$ describes its *cliquishness* [202], which is the ratio of its active links compared to the maximum possible [202][165][203], expressed as:

$$C(t)_k = \frac{2|E_c(t)|}{|C(t)|(|C(t)|-1)} \quad (2.25)$$

A $C(t)_k$ value close to 1 indicates a *clique* of densely connected nodes. The graph of Figure 2.29, which depicts a network with 4 links and 5 nodes, has a clustering coefficient of: $\frac{2(4)}{5(5-1)} = 0.4$.

The Spectral Graph Theory (SGT) approach particularly supports determining the levels of clustering and connectivity that exist in networks [121][204][205][206]. Several SGT metric derivations, including *eigenvectors*, *eigenvalues*, *Fiedler vector*, *algebraic connectivity*, *betweenness centrality*, etc., have been described as powerful topology inference constructs. The *Adjacency* (A), *Degree* (D) and *Laplacian* (L) are basic SGT matrices that encode network connectedness. Figure 2.30 shows an example structure of the matrices, $(D - A) = L$, based on the small example graph in Figure 2.29. The matrix A describes the links that exist between nodes, with 1 representing the existence of a link and 0 otherwise. The matrix D simply records the

degree of each node along the diagonal, and 0 in other cells. Thus, the resulting Laplacian matrix L encodes the connectedness properties of a network.

$$\begin{array}{c}
 v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \\
 \\
 \begin{array}{c}
 v_1 \\
 v_2 \\
 v_3 \\
 v_4 \\
 v_5
 \end{array}
 \begin{bmatrix}
 1 & 0 & 0 & 0 & 0 \\
 0 & 2 & 0 & 0 & 0 \\
 0 & 0 & 3 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 1
 \end{bmatrix}
 -
 \begin{array}{c}
 \\
 \\
 \\
 \\
 \\
 \end{array}
 \begin{bmatrix}
 0 & 1 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 & 0 \\
 0 & 1 & 0 & 1 & 1 \\
 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0
 \end{bmatrix}
 =
 \begin{array}{c}
 \\
 \\
 \\
 \\
 \\
 \end{array}
 \begin{bmatrix}
 1 & -1 & 0 & 0 & 0 \\
 -1 & 2 & -1 & 0 & 0 \\
 0 & -1 & 3 & -1 & -1 \\
 0 & 0 & -1 & 1 & 0 \\
 0 & 0 & -1 & 0 & 1
 \end{bmatrix}
 \end{array}$$

Degree (D)
Adjacency (A)
Laplacian (L)

Figure 2.30: Example topology matrices of the graph in Figure 2.29

In SGT, the node degree is referenced as *degree centrality*, which describes the local property of influence or importance that a node has with respect to its immediate neighbours. Moreover, *eigenvector centrality* describes a node's importance based on the aggregate of its neighbours' degree centralities. The implicit solution for finding the degree centrality c_v for each node v_i in a network is given as [121]:

$$c_{v_i} = \frac{1}{\lambda} \sum_{v_j \in N(v_i)} A_{ij} c_{v_j}, \quad \forall v \in C \subseteq G \quad (2.26)$$

$N(v_i)$ denotes neighbours of node v_i , and λ is a normalizing factor. Equation 2.26 is rewritable as [121]:

$$A_{nn} \mathbf{x} = \lambda \mathbf{x} \quad (2.27)$$

Equation 2.27 generates a set of eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$, according to the number of nodes in the network. Associated with each eigenvalue is an eigenvector. The eigenvector centrality values are stacked in the eigenvector related to the largest (non-negative) eigenvalue. Figure 2.30

demonstrates the eigenvector centrality values (column 2) related to the example network in Figure 2.29.

$$\begin{pmatrix} v1 & 0.76536 & -1.67513 \\ v2 & 1.41421 & -0.80606 \\ v3 & 1.84775 & 0.48119 \\ v4 & 1.00000 & 1.00000 \\ v5 & 1.00000 & 1.00000 \end{pmatrix}$$

Figure 2.31: Eigenvector (column 2) and Fiedler vector (column 3) for the graph of Figure 2.29.

Moreover, the *algebraic connectivity*, which is the second smallest non-zero eigenvalue, λ_2 , and the associated *Fiedler vector* (such as in column 3 of Figure 2.31) is derivable based on [121]:

$$Lx = \lambda x \quad (2.28)$$

In Figure 2.31, the negative Fiedler values of the nodes v_1 and v_2 indicates weakly connected nodes. A suitable partitioning of the network would separate the negatives from the positives. Algebraic connectivity represents the magnitude of network nodes' connectedness. For the small example graph (Figure 2.29), the algebraic connectivity is 0.51880569, an average value. It has been shown in different works that the distributed *in-network* computation of the SGT measures can be performed using the power iteration method [205][204][121][207]. However, node mobility adds a layer of complexity to SGT computations, which limits its use over highly varying topologies.

2.6.3.2 Notations of relative node displacements in MANET

In geographic routing, the length measure of the link between a pair of adjacent nodes u and v is generally given by the Euclidean distance equation [111]:

$$dist(u, v) = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2} \quad (2.29)$$

MANET topology fluctuations and variations often result from the displacements of nodes due to mobility. The basic measure of node mobility is the velocity, v , which is the rate of change in distance coverage, Δd , compared to time change, Δt , i.e.,

$$v = \frac{\Delta d}{\Delta t} \quad (2.30)$$

The relative velocity, *relvel*, at instance i between two adjacent nodes u and v is defined as [208]:

$$relvel_i^{u,v} = \frac{d_i^{u,v} - d_{i-1}^{u,v}}{t_i - t_{i-1}} \quad (2.31)$$

where d is the measure of distance between the two nodes, and t is the time of the velocity reading.

The relative velocity may be normalized to factor in other events such as GPS fluctuations and the effect of wind gusts on small UAVs [208].

The *encounter rate* metric describes the new neighbours that a node comes in contact with over a given time. The encounter between nodes u and v is defined as [209]:

$$e^{u,v} = \{u, v, t, \Delta t\} \quad (2.32)$$

where t is the encounter incident time and Δt is the duration. For a node u , the set of all its encounters over a duration T is annotated as $E^{(u)}$, while the *average encounter rate* can be estimated to be [210][209]:

$$\begin{aligned} AER^{(u)} &= \frac{|E^{(u)}|}{T} \\ &= R \times vel_{max} \times density \end{aligned} \quad (2.33)$$

Equation 2.33 shows that the *AER* is also an indicator of a node's relative velocity with respect to its neighbours; where R is the communication radius, vel_{max} is the nodes' maximum velocity limit (in the Random Waypoint model) and *density* is that of the network. Furthermore, based on the least square sum value of the *AER*, the *path encounter rate* metric, *PER*, is defined in [49] for selecting routes having events of minimal node mobility and density over m hops:

$$PER = \sum_{i=1}^m AER_i^2 \quad (2.34)$$

The *ETX* (*expected transmission count*) metric [211][212][183] finds a path of high throughput in packet delivery. It is defined as a function of the beacon messages successfully sent, r^f , and the corresponding *ACK* replies, r^r , defined as [211]:

$$ETX = \frac{1}{r^f r^r} \quad (2.35)$$

Together with the relative velocity, the *ETX* for predicting *link durability* over UAV's link is defined in [208] as:

$$ETX^{u,v} = \frac{e^{relvel_i^{u,v} \beta}}{r^f r^r} \quad (2.36)$$

where e is Euler's constant and β is a non-negative parameter. Unlike the *ETX*, the *link expiration time* (*LET*) metric defined in [213] is a free space method that does not consider wireless conditions. *LET* describes the amount of time that a link will stay connected for a pair of moving nodes.

The *change degree of connectivity* [79], *cdc*, is a measure of node encounters based on *neighbour sets*. Mobility causes changes in the neighbour set $N(u)$ of a node over time intervals. The *cdc* of node u within an interval $[t-T, t]$ is defined as [79]:

$$cdc_{(u)}(t) = \frac{|N_{(u)}(t-T) \cup N_{(u)}(t)| - |N_{(u)}(t-T) \cap N_{(u)}(t)|}{|N_{(u)}(t-T) \cup N_{(u)}(t)|} \quad (2.37)$$

High values of $cdc_{(u)}$ imply a higher number of new neighbours encountered, in similarity to a high *AER* [210][49] of Equation 2.33. In [214], the *mobility factor*, MF , determines the stability of the neighbour set of a node u [214][49]:

$$MF_{(u)} = \sqrt{1 - \frac{|N_{(u)}(t) \Delta N_{(u)}(t-T_{HELLO})|}{|N_{(u)}(t) \cup N_{(u)}(t-T_{HELLO})|}} \quad (2.38)$$

where $A \Delta B$ is the symmetric difference, i.e., excluding intersection elements, in the union of the sets A and B . If the neighbour set is stable over a period T , Equation 2.38 yields an MF that is close to 1. A related metric is the *change rate of connectivity* [85][79], which measures relative mobility by the difference in the number of connections and disconnections over duration T .

The *dependency* metrics of $D_{spatial}$ and $D_{temporal}$ are defined in [24] to indicate the influence of mobility patterns, such as the RWP and the Manhattan models, which underlie the dynamic topologies of mobile nodes. The $D_{spatial}$ metric, based on two other metrics, *relative direction* and *speed ratio*, describes the *spatial dependency* between a pair of connected nodes at time t . A MANET segment with nodes moving in a similar direction and at similar speeds will have a high average $\bar{D}_{spatial}$ value. The $D_{temporal}$ metric describes the intra-dependency of a node's velocity between time steps t and t' . The average value $\bar{D}_{temporal}$ will be high for a network segment where nodes' velocities are influenced by those of the previous time step. The Equations of the $D_{spatial}$ and $D_{temporal}$ metrics are not shown here due to their high level of intricacies. In [19], a *topology change degree*, TCD_{ntwrk} , metric describes dynamics of varying topologies, which is based on $TCD_{u, nbrs}$ for the neighbourhood of node u , which in turn is based on $TCD_{(u, v)}$ that is obtainable from relative interactions between adjacent nodes u and v . $TCD_{(u, v)}$ is defined as a composite metric of *displacement distance* (d), *direction* (θ) and *relative velocity* (*relvel*) thus:

$$TCD_{(u,v)}(t, t + T)$$

$$= \left(w_1 \cdot \frac{|d_{(u,v)}^{t+T} - d_{(u,v)}^t|}{d_{(u,v)}^t} + w_2 \cdot \frac{|\theta_{(u,v)}^{t+T} - \theta_{(u,v)}^t|}{2\pi} + w_3 \cdot \frac{|relvel_{(u,v)}^{t+T} - relvel_{(u,v)}^t|}{relvel_{(u,v)}^t} \right) \quad (2.39)$$

The TCD_{ntwrk} metric [19] over a network determines the topological change perception and thresholds for the adaptive performance switching of routing protocols, including OLSR, AODV, and DSDV.

2.6.4 Summary

Effective context awareness by a routing protocol equips it for efficient adaptive packet forwarding performance over fluctuating and varying MANET topologies. Context knowledge can be acquired at the macro and the micro levels as well as in-depth and in-breadth through metrics and tools that enable data acquisition and synthesis. Apart from those described in this section, various other metrics for discerning the MANET routing environment can be found in [215][216][196]. The increasing availability of technological tools, such as GPS, has been described as a motivation for the advancement of the position-based or geographic routing method [24]. Enabling technologies, such as the *5G* and *big data* [191][192] schemes offer opportunities for improving the monitoring and acquisition of context knowledge in dynamic MANETs. For example, the application of these tools can mitigate the processing demands for efficient packet forwarding, such as for k -hop forwarding, $k > 2$.

CHAPTER 3

3 THE MULTI-GREEDY FORWARDING DEFINITION

The definition of a *multi-greedy* approach to progress-based forwarding in MANET geographic packet routing is given in this section. Multi-greedy forwarding is a self-adaptive multimethod [61][217] approach in which each network node can autonomously, through its copy of the geographic routing protocol having a set of greedy metrics, determine a suitable method that should be used for optimal packet forwarding performance depending on the current topological condition. In a multi-greedy scheme, multiple greedy metrics including GREEDY [95], GREEDY-COMPASS [102], ELLIPSOID [57], etc., can be adaptively and optimally applied or re-applied sequentially to a packet forwarding task over varying and fluctuating topologies.

3.1 The multi-greedy packet forwarding model

The multi-greedy model, illustrated in Figure 3.1, is a *switching* hybrid greedy approach to *progress-based* geographic packet forwarding and routing [32][102][48][94]. Existing greedy forwarding methods, including different hybrids that are described in chapter 2, differ from the switching progress-based multi-greedy method. For example, the ARP [48] protocol uses the GREEDY [95] method for normal packet forwarding but switches to the CR [103] mode for recovery from greedy failures only. The GREEDY-COMPASS [102] method is a flat and non-switching trade-off of the GREEDY and CR methods. Similarly, the HGR [32] protocol, which aims at nodal energy use optimization, is a trade-off comprising the CR and MFR [77] methods. Note that the switching methods described in [94], e.g., the T-ABLAR-T methods, address greedy failures or routing traffic minimization rather than adaptiveness to progress-based forwarding. Furthermore, the multi-greedy definition is not in the form of concurrent or parallel running of

multiple algorithms [217]. Rather, in a multi-greedy forwarding scheme, a set of greedy metrics are engaged or re-engaged sequentially in performing a geographic packet routing task.

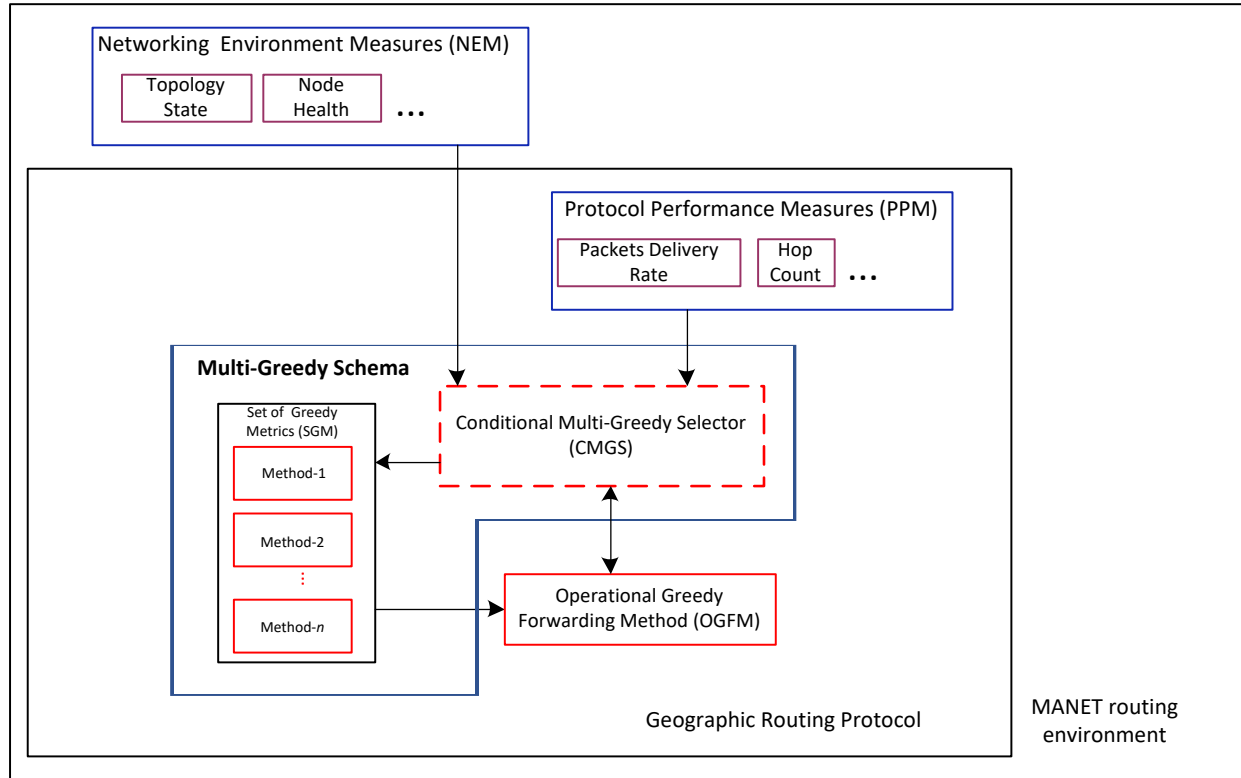


Figure 3.1: Multi-greedy packet forwarding model with context-aware measuring units

The Operational Greedy Forwarding Method (OGFM) aspect in Figure 3.1 shows the standard greedy forwarding module that is part of every geographic routing protocol. However, the OGFM module is subject to adaptive functioning in the multi-greedy forwarding performance. The Multi-Greedy Schema of the model is a novel introduction to geographic packet forwarding, which consists of the (1) Set of Greedy Metrics (SGM), i.e., the repository of the greedy algorithms meant for adaptive switching, and the (2) Conditional Multi-Greedy Selector (CMGS) that determines

and assigns the most optimal greedy method to packet forwarding at any determined point during routing. With regard to redundancy systems of *multi-version* cast, such as are incorporated in fault-tolerant systems [218] [219], a multi-greedy component may also add such layer(s) where necessary. For example, in Figure 3.1, it may be desirable in the SGM to incorporate multiple versions of each greedy metric to handle situations like levels of floating-point calculations, 2-D and 3-D inputs, etc. However, the multi-greedy method that we specify focuses on optimizing geographic packet forwarding through dynamic switching of multiple greedy metrics in relation to varying MANET topologies.

In the multi-greedy model (Figure 3.1), the specific functions of the CMGS are:

- (i) to perform the aggregating of context data from the Networking Environment Measures (NEM), Protocol Performance Measures (PPM), or any other. CMGS formats aggregated context data for the ‘if-else’ conditional testing towards designating a suitable OGFM.
- (ii) to monitor the current packet forwarding efficiency of the OGFM against the prevailing context measures, and simultaneously assess the methods in the SGM for a potentially more efficient one, and if any is found then such will be designated to become the OGFM.
- (iii) to perform adaptive switching of the OGFM with any newly designated method.

Note that the NEM and PPM modules are external to the multi-greedy scheme although firmly integral to its functioning. The items shown in the two context modules, i.e., NEM and PPM, are not exhaustive, including Topology State, Node Health, Packets Delivery Rates, Hopcount, etc. The NEM and the PPM modules can be implemented using different combinations of existing or newly derived context awareness mechanisms, such as network topology state sampling [56][220], flow reporting [221], etc.

The context-aware adaptive multi-greedy method is an optimization function involving any suitable set $\{x_1, \dots, x_n\}$ of the geographic greedy metrics, by which the most optimal method to apply at each stage of a packet routing task is continually decided by [217]:

$$\begin{aligned}
\underset{x_j \in \mathcal{X}}{\text{optimize}} \mathcal{Y} &= f(x) \\
&= f(x_1, \dots, x_n). \\
&= \bigwedge_1^n x_j
\end{aligned} \tag{3.1}$$

Each greedy metric x_j is assignable, and re-assignable, sequentially to a routing task for efficient packet forwarding performance as the related MANET topology varies. The multi-greedy optimization function $f(x)$ resides in the geographic routing protocol copy of each network node, by which the greedy selection metric assignment is independently decided for each packet(s) forwarding action. An assumption, in the case of Equation 3.1, is that the routing terrain is highly varying, while the function performs a topology-aware matching at each intermediate node. However, when the required topology awareness is beyond a forwarding node's immediate neighbourhood, i.e., h -hop ($h \geq 2$), the multi-greedy function could perform a greedy assignment covering a contiguous set of relay nodes. For a contiguous h -hop assignment, we have

$$f(x_1, \dots, x_n) = \bigwedge_1^h (\bigvee_1^n x_j) \tag{3.2}$$

which results in a sequence of h number of greedy elements that are to be applied one after the other in performing a part, $y_i \in \mathcal{Y}$, or all of the task $\mathcal{Y}$. A contiguous specification for the complete performance of task $\mathcal{Y}$ implies an end-to-end n -hop topology awareness. The contiguous type of assignment could be based on a sub-global efficiency policy, such as flat routing over established homogeneous network segment(s).

3.2 Multi-greedy forwarding designs

Based on the model shown in Figure 3.1, practitioners can derive multi-greedy designs in relation to context-aware schemes in geographic routing. Each geographic MANET node has its routing protocol copy which implements multi-greedy forwarding.

▪ Base multi-greedy algorithms

When a greedy method is assigned to perform a forwarding task at a node u , the basic data that it requires are the location coordinates of the destination d and those of its neighbours in the set $N(u)$. The greedy forwarding method selects from $N(u)$ a next-hop v based on computations that obtain the applicable Euclidean distances using the function $dist(\cdot)$, which in the 2-D is [111]:

$$dist(u, v) = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2} \quad (2.29)$$

where x and y are location coordinates. For a packet forwarding performance task toward a destination d , and $v_i \in N(u)$, $i = 1, 2, \dots, |N(u)|$, the employable greedy algorithm at u are here specified for the classic methods [35] (see Figure 1.2):

(i) GreedyGREEDY($u, d, N(u)$) //GREEDY [95]

$next_u \leftarrow v_i: \min\{dist(v_i, d)\}$

(ii) GreedyNC($u, d, N(u)$) //NEAREST CLOSER [100]

$next_u \leftarrow v_i: \min\{dist(u, v_i)\}$

(iii) GreedyNFP($u, d, N(u)$) //NEAREST WITH FORWARD PROGRESS [43]

$next_u \leftarrow v_i: \min\{dist(u, v_i')\}$

(iv) GreedyMFR($u, d, N(u)$) //MOST FORWARD WITHIN RANGE [77]

$next_u \leftarrow v_i: \max\{dist(u, v_i')\}$

(v) GreedyCR($u, d, N(u)$) //COMPASS ROUTING [103]

$next_u \leftarrow v_i: \min\{angle(v_i, u, d)\}$ //calculates $\angle v_i u d$

Similar to the above algorithms, those for other greedy forwarding methods such as the GREEDY-COMPASS [102], RCOMPASS [133], etc., are given in the literature. If any existing greedy method has the potential to perform exceptionally in an environment, then such can be engaged in a multi-greedy formulation. The *Procedure_Multi_Greedy_Assignment(.)* that assigns the different greedy algorithms according to their context optimality matching is of the form:

```

Procedure_Multi_Greedy_Assignment(GreedyMethod_1(), GreedyMethod_2(),
..., GreedyMethod_n())
{
    DO WHILE (packetsForwardingIsTrue)
    {
        get.ContextIndications();
        EvaluateContextAgainstGreedyMethods();
        AssignOperationalMethod();//engages the designated greedy method
    }
}

```

In a packet routing instance, the *ContextIndications(.)* routine of the algorithm obtains values of the topology state and/or the packet delivery performance measures, which are assessed by *EvaluateContextAgainstGreedyMethods(.)* to designate a most suitable method. This module would include submodules that perform such tasks as exploration learning to formulate or reformulate, a multi-greedy method for an environment. The *AssignOperationalMethod(.)* switches the currently operational greedy metric, i.e., OGFM as shown in Figure 3.1, with the designated alternative if found different.

3.3 Evaluation environment

The multi-greedy scheme is essentially aimed at progress-based greedy forwarding improvement of geographic protocols, including GPSR [30], GFG [104], ARP [48], etc. MANET routing protocol performance is mostly evaluated using simulation experiments [222] [223][224], which is adopted in this study. Other possible evaluation methods include mathematical analysis, testbeds, and real-life performances. A mathematical analysis of dynamic MANET becomes problematic when several network parameters must be represented in the requisite equations. The issue with the testbed and real-life experiments is the high cost of setup, and the difficulty of process management as well as scenarios replicating [225] [181]. Nevertheless, our future investigations in multi-greedy forwarding for FANET-specific environments could involve the UAV testbed, which increasingly is becoming affordable [208][226][227].

Simulation experiments should closely and accurately model the intended environment [228], while the related process parameters should be selected carefully, as improper inputs affect results [162][229]. Relevant open-source network simulators include Omnet++ [85][79][194], OPNET Modeler [193] and NS-3 [222]. The NS-3 is meant to replace the popular NS-2 which was decommissioned in 2008 [222] [25]. An improved feature of NS-3 is its scalability to simulate

very large networks. Evidence shows that NS-3 is currently widely patronized in academic research [7][25][59][210][223][224][230][231]. A disadvantage of NS-3 is its lack of a dynamic node-size modifying mechanism. Nevertheless, we chose to use the NS-3 due to its popularity and availability. This study is mostly concerned with the geometric configurations of the MANET environment which NS-3 fairly simulates. A mobility generator, Bonnmotion [232], was also employed alongside the NS-3. Bonnmotion generates scenario files of mobile nodes based on input parameters such as simulation area, number of nodes, mobility pattern, and velocity. In addition to the data generated using Bonnmotion, another set was obtained from the Github repository, alongside the GPSR routing protocol's program code [233][234]. Parts of the code are shown in Appendix C. Since we are concerned only with the progress-based greedy forwarding improvement, GPSR's greedy recovery mode was disabled in the simulation experiments. GPSR [30] primarily implements the GREEDY method. Each network node operates with a copy of the geographic routing protocol, i.e., GPSR, which in turn can incorporate multiple greedy forwarding algorithms of various sorts, i.e., MFR, NC, etc.

The primary measures in geographic routing protocol evaluation are the packet delivery rate (PDR), *hopcount* and end-to-end *delay* [65][7][31]. The PDR measure was described in section 1.7 to be a ratio value:

$$PDR = \frac{\sum r}{\sum s} \quad (3.3)$$

In a packet delivery session, r represents the amount received at the destination while s is the amount originally sent from the source. Hopcount is the number of intermediate nodes that packets traverse from the source to the destination. The nature of bottlenecks and the number of intermediate points traversed by packets contribute to the delay (in terms of time) measure between

endpoints. An efficient multi-greedy forwarding formulation should show a performance edge in comparison to those of its constituent parts.

CHAPTER 4

4 MULTI-GREEDY FORWARDING DESIGNS AND DEMONSTRATIONS

Two example designs of multi-greedy forwarding based on the context-aware measures of packet flow report and packet forwarding distance are presented in this section. Multi-greedy designs must be performed in conjunction with applicable context-aware schemes. Context awareness provides conditional testing and options switching of greedy forwarding during a routing task.

4.1 Multi-greedy forwarding based on network packet flow reports

In the packet flow report design of the multi-greedy forwarding scheme, the networking context knowledge is obtainable from the periodic stream measures that the IP Flow Information Export (IPFIX) protocol [62][63] generates. Additionally, the task performance paradigm of *exploration-exploitation* is employed to stress the implementation approach using IPFIX reports.

4.1.1 Background: IPFIX-based packet flow reporting

The IPFIX tool [62], also known as *flow monitor*, is a network performance measurement tool that detects streams of packet flows emanating from some source(s) toward some destination(s) and exports related aggregate reports. A *flow* is tagged based on packets' common header labels such as port number, source, a destination address, timestamp, etc. The flow-based measures provide network performance information including bitrates, packet loss ratio, delay, etc. [63][221]. The IPFIX approach is described as a robust alternative to the traditional packet capture method. Flow-based measurement is used in [221] to characterize and differentiate network segments in a campus. Employing the IPFIX reports for performance improvement, e.g., in a multi-greedy scheme, would imply a prior or interleaved *exploration* of the reference environment. The works in [56] and [220] show the benefits of the exploration-exploitation approach to path-

search tasks performance, based on a periodic sampling of the current network context and deciding the most optimal search method for engagement(s). The study in [220] demonstrated it is advantageous even in a singly repeated case of the path search task. Moreover, they showed that the exploration-exploitation method outperformed comparable heuristic or randomized approaches when the required task repetition is low. Their findings also recommend interleaved *learning while performing* better than a priori *learning before performing* method.

4.1.2 Methodology

A multi-greedy forwarding design based on IPFIX reports would need to often perform optimization assessments of a handful of greedy methods every time an unknown environment is encountered or in response to network topology metamorphoses. The criteria of the number of packets delivered at the destination in each labelled IPFIX flow report shall be used to evaluate the performance of each greedy method. The method that has the highest number of packets received at the destination endpoint is considered optimal for that flow. For a session of packet routing over an environment, the IPFIX protocol normally generates p flow reports, in proportion to the duration of task performance. Subsequently, a multi-greedy performance execution file is to be developed containing a sequential list that associates each flow with an optimal greedy method. Consider the flow report i ($i = 1, 2, \dots, p$) resulting from a subtask y_i to be optimized; and based on Equation 3.1, the Equation 4.1 shows the optimization subfunction relating to the flow which applies an x_j greedy method continuously for that part.

$$\begin{aligned} \underset{x_j \in \mathcal{X}}{\text{optimize}} y_i &\subseteq y = (f(x_1, \dots, x_n))_{IPFIXflow_i} \\ &= (V_1^n x_j)_{IPFIXflow_i} \end{aligned} \quad (4.1)$$

Then, for the complete performance of task y , and based on Equation 3.2, we have

$$\begin{aligned}
\underset{x_j \in \mathcal{X}}{\text{optimize}} y &= \bigwedge_{i=1}^p \left((f(x_1, \dots, x_n))_{IPFIXflow_i} \right) \\
&= \bigwedge_1^p (V_1^n x_j)_{IPFIXflow_i}
\end{aligned} \tag{4.2}$$

A currently assigned greedy method x_j applies itself to subtask y_i until completed; and subsequently, another or the same selected x_j is assigned to y_{i+1} , and so on until the final subtask y_p is performed. This process enables the multi-greedy scheme to optimize the PDR, hopcount, delay or any other measure for each flow; by choosing the most optimal greedy method identified with the flow report i . Consequently, the scheme generates the optimization template for performing the whole of repetitive task y consisting of p contiguous flows. For instance, based on Equation 4.2, the multi-greedy PDR performance *maximization* is expressible as:

$$\begin{aligned}
\underset{x_j \in \mathcal{X}}{\text{max}_{PDR}} y &= \bigwedge_{i=1}^p (f(x_1, \dots, x_n)_{\text{max}_{PDR}})_{IPFIXflow_i} \\
&= \bigwedge_1^p \left((V_1^n x_j)_{\text{max}_{PDR}} \right)_{IPFIXflow_i}
\end{aligned} \tag{4.3}$$

and

$$= \bigwedge_{i=1}^p ((GREEDY \vee ELLIPSOID \vee GREEDY_COMPASS)_{\text{max}_{PDR}})_{IPFIXflow_i} \tag{4.4}$$

when the greedy metrics of ELLIPSOID, GREEDY and GREEDY-COMPASS are reflected. That is, after *exploration* performance of task y by each of the greedy metrics, comparisons is made for each IPFIX flow y_i using formulation 4.4 which is the multi-greedy selection component for *exploitation* task performances.

In the following demonstration, the greedy algorithms of ELLIPSOID [57], GREEDY [95] and the composite GREEDY-COMPASS [102] are each employed to perform *exploration* running of the routing task over the MANET environment of interest. The metrics of GREEDY and ELLIPSOID have been described in chapters 1 and 2. These greedy methods are known to exhibit higher performance levels in PDR measures, which is the chosen maximization objective in this

example. For the GREEDY metric, an x_j , its computation designates the next node u to be a selected neighbour node v_i having a minimum Euclidean distance to the destination d :

$$u_{next-relay}^{(GREEDY)} \leftarrow \{v_i \mid \min_{dist}\{|\overline{v_i d}|\}\}, \quad (2.3)$$

while the ELLIPSOID metric performs [111] [57]:

$$u_{next-relay}^{(ELLIPSOID)} \leftarrow \{v_i \mid \min_{dist}\{|\overline{u v_i}| + |\overline{v_i d}|\}\} \quad (4.5)$$

Next, is a description of the GREEDY-COMPASS [102] [133] metric, which first computes with the CR metric to shortlist two candidate nodes ‘above (a)’ and ‘below (b)’ and then selects the one having the minimum distance to the destination as the next relay. Based on Equations 2.3 and 2.4, we have

$$u_{next-relay}^{(GREEDY-COMPASS)} \leftarrow \{v_i \mid \min_{dist} \{|\overline{v_{i(a)}^{(CR)} d}|, |\overline{v_{i(b)}^{(CR)} d}|\}\} \quad (4.6)$$

given that $v_{i(a)}^{(CR)} \leftarrow \min_{angle} \{v_i \mid \{\angle v_{i(a)} u d\}\}$, and $v_{i(b)}^{(CR)} \leftarrow \min_{angle} \{v_i \mid \{\angle v_{i(b)} u d\}\}$.

4.1.3 Simulation parameters

A VANET routing environment was simulated using NS-3 [63][222] which incorporates the IPFIX [62] module. Table 4.1 shows the settings of the simulation environment. Packet transmission was performed with unicast 512 bytes CBR/UDP from a single source to a single destination over multi-hop connections. The network contains 110 mobile nodes that have a moving velocity of 0-15m/s over an area of 1100m². The transmission range of each node is 280m. The simulation durations are 200, 600, and 1000 seconds. Simulations were also performed for the durations of 1400, 1800 and 2200 seconds to confirm the related trend in the multi-greedy forwarding

outcomes. Appendix C shows parts of the NS-3 [59][222][235] C++ code for the multi-greedy IPFIX-based context-aware implementation, which is contained in the employed GPSR [30] packet routing protocol.

Table 4.1: Simulation parameters for the IPFIX flow-based multi-greedy packet forwarding

Parameter	Value
NS-3 simulator	v. 3.23
Simulation time	200s, 600s, 1000s, 1400s, 1800s, and 2200s
Data packet size	512 bytes
Traffic application	CBR/UDP unicast
Number of nodes	110
Simulation area	1100m ²
Transmission range	280m
Node speed	0 - 15m/s

4.1.4 Results and discussion

At the end of each simulated *exploration* period of running a base greedy method, the IPFIX flow tool [62] provides a list of the flows and the number of packets delivered in each one. The flows are compared by their similar IDs in the three exploration sessions of running the base greedy methods involved. Each flow report was then associated with the most efficient greedy method, which is used to determine the multi-greedy formulations and the respective input files. Three formulations were tested, all including the GREEDY method as follows:

- GREEDY~ELLIPSOID (GE)
- GREEDY~GREEDY-COMPASS (GGC)
- GREEDY~GREEDY-COMPASS~ELLIPSOID (GGCE)

Each derived multi-greedy method is designated using its constituent methods' names concatenated with the symbol '~'. For example, the multi-greedy method GREEDY~GREEDY-COMPASS consists of the GREEDY [95] and the GREEDY-COMPASS [102] base methods.

Each simulation run, including those of the base and the multi-greedy methods, was for 200s, 600s, and 1000s durations, which respectively generated 371, 886, and 1047 IPFIX flow reports. A reason for taking evaluation measures for three different durations is to determine whether the task exploration duration or even the number of flows generated, has any influence on successful packet delivery outcomes. The consistent number of flows generated in a simulation duration, irrespective of the base greedy or even multi-greedy method involved, enhanced the per-flow comparison assessments. For example, in the 200s period runs, we could isolate which method performed best in each of the associated 371 flows. This further enhanced the multi-greedy assignment of the most efficient base greedy method to each flow.

Figures 4.1 to 4.5 show the resulting packet delivery performance charts. The PDR performance measures (Figures 4.1, 4.2 and 4.3) as shown across and within each figure demonstrate certain similarity trends for all the methods, including the multi-greedy types. This implies that any arbitrary and fair sampling period suffices for the exploration assessment of the base greedy methods; while the exploitation running of the multi-greedy forwarding method, if beneficial, over the same period should yield commensurate performance improvements. In general, the multi-greedy methods show improved PDR performances compared to the base greedy types. But in some cases, the GREEDY-COMPASS base method showed better performance than the GE multi-greedy method. The GGCE method showed overall superior performance. The multi-greedy method of GGCE outperformed its constituent types as well as the other two multi-greedy methods in the PDR measures. For example, the GGCE in the duration 1000s (Figure 4.3) shows a PDR of 66.96 in comparison to 59.51, 62.58, and 61.61 of its constituents including GREEDY, GREEDY-COMPASS, and ELLIPSOID respectively. Therefore, in terms of optimum PDR performance, the GGCE formulation can be considered the choice for use in repetitive forwarding tasks regarding this packet routing environment. Note that the geographic greedy packet routing is inherently repetitive in path search since no end-to-end connection is involved.

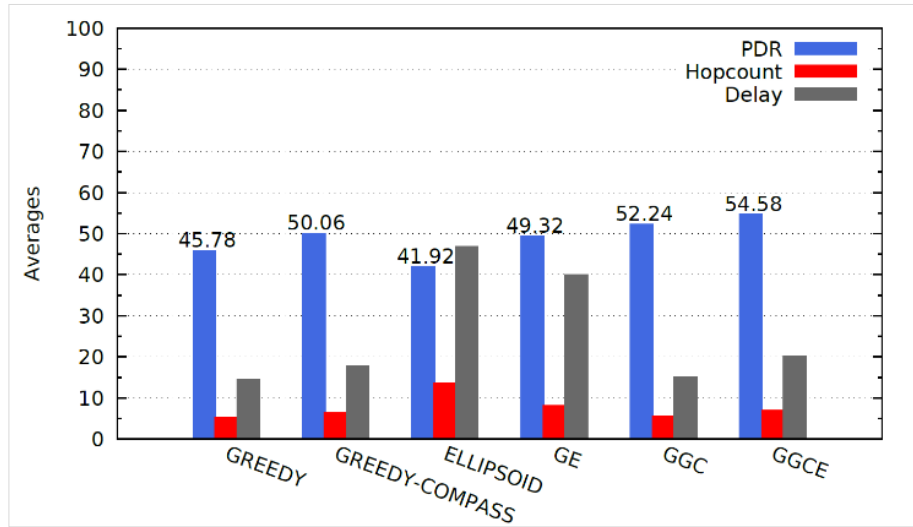


Figure 4.1: The 200s (371 flow reports) performances of the greedy and multi-greedy methods.

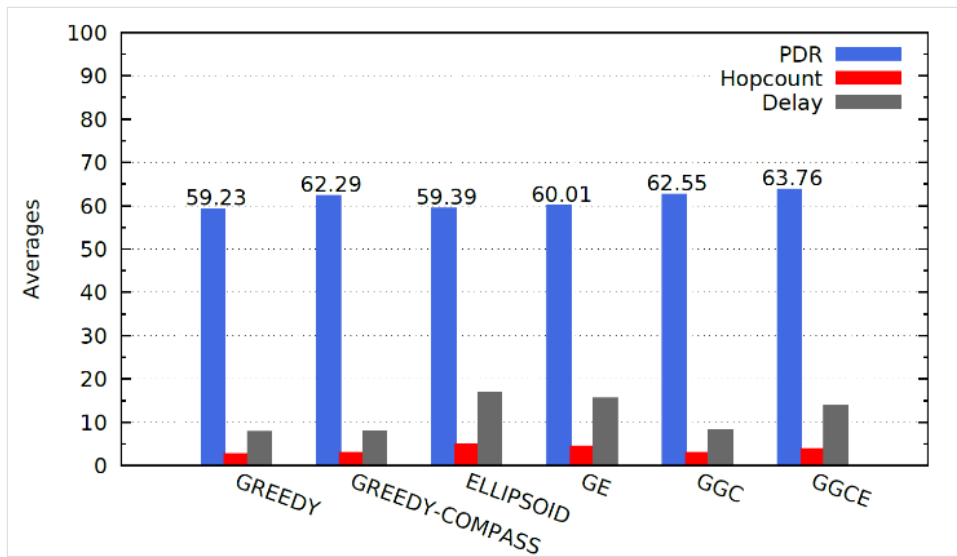


Figure 4.2: The 600s (886 flow reports) performances of the greedy and multi-greedy methods.

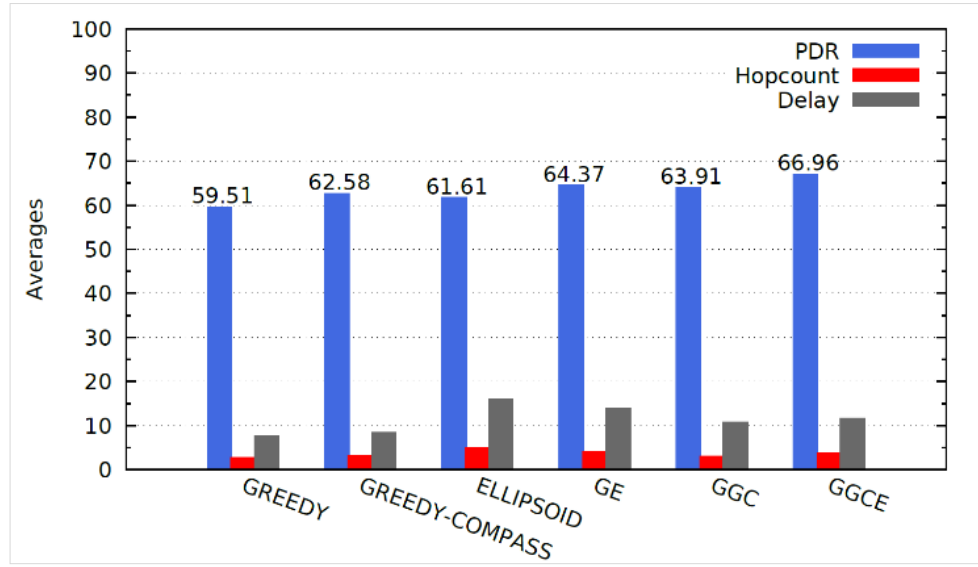


Figure 4.3: The 1000s (1047 flows) performances of the greedy and multi-greedy methods.

Figure 4.4 shows the GGCE's trends of performance in the averages of PDR, delay and hopcount measures for the execution periods extending from the 200s to the 2200s. Despite slight variations over the range of increasing packet forwarding durations, the average PDR showed a rise from about 50% and remained relatively high above 60%; while the delay and the hopcount dropped and remained positively low at around 10 seconds and 5 hop levels respectively. Therefore, using the GGCE multi-greedy method for exploitation task performance in this routing scenario, especially in long-duration cases, leads to highly improved PDR, hopcount, and delay measures. A similar pattern of performance (like that of the GGCE) is reflected for the GGC (Figure 4.5) and the GE (Figure 4.6) methods.

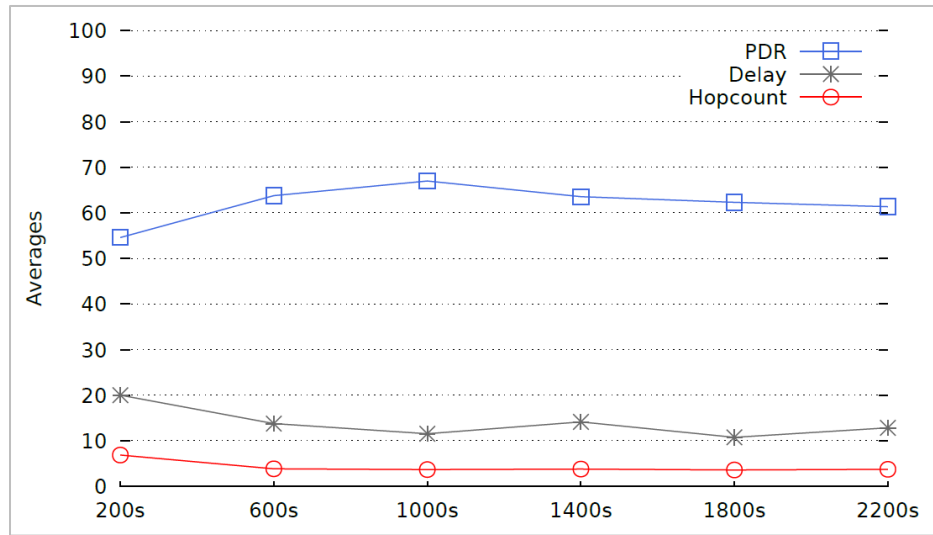


Figure 4.4: The GGCE multi-greedy method performance for different forwarding durations

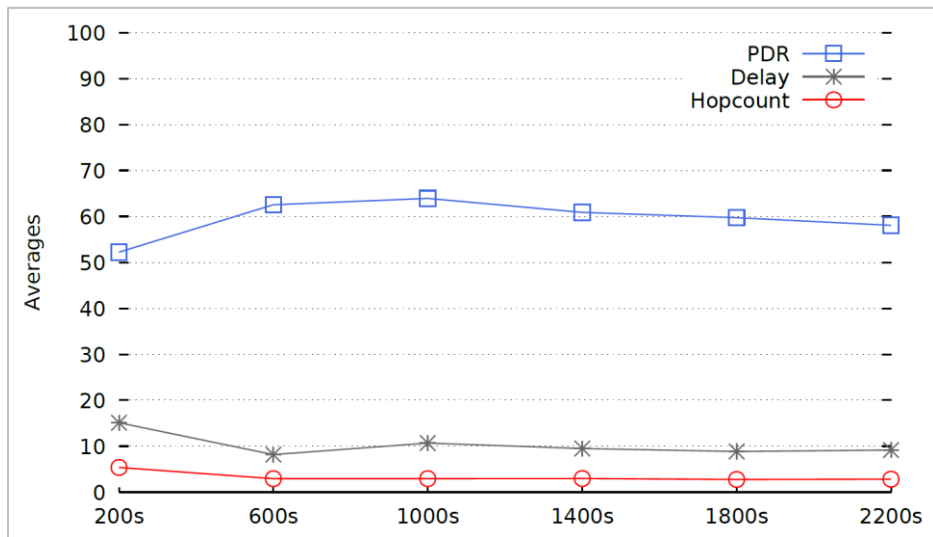


Figure 4.5: The GGC multi-greedy method performance for different forwarding durations



Figure 4.6: The GE multi-greedy method performance for different forwarding durations

For the simulated environment, Figure 4.7 shows the differences in the *expected* versus the *actual* successful packets' delivery for the multi-greedy methods of GGC, GE, and GGCE. These are akin to *exploration* discoveries of the optimal performance expectations which led to the current multi-greedy formulations, and the follow-up *exploitation* outcomes. The expected values are based on the aggregate optimal of the base greedy methods, while the actual values are the outputs of running the multi-greedy methods. In each comparison case for the three methods, there is a slight average drop of about 2% in the actual packets successfully delivered, which is quite tolerable considering MANET transmission vagaries. Overall, in comparison to the constituents, the formulated multi-greedy methods of GGC, GE, and GGCE showed improved PDR, as well as relative reductions in hopcount and delay measures.

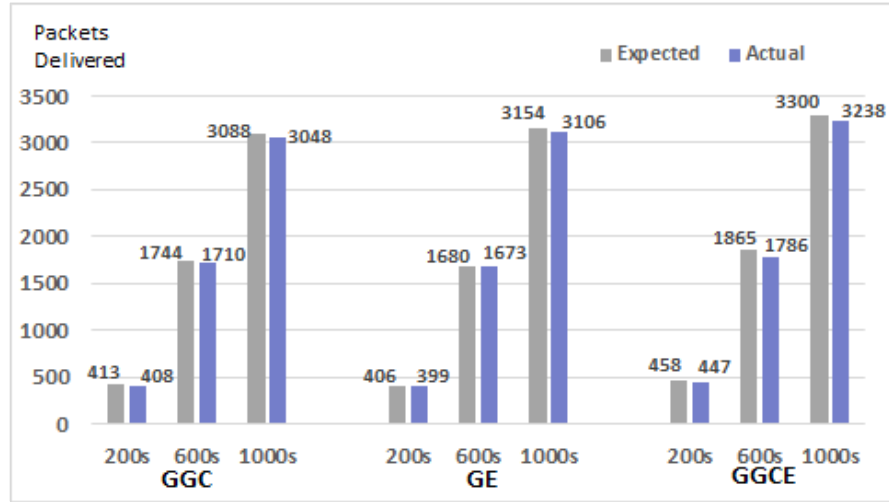


Figure 4.7: The multi-greedy methods' PDR optimal performance: expected versus actual.

4.2 Multi-greedy forwarding using packet forwarding distance dissimilarity measures

The multi-greedy forwarding design described in this section is a topology-aware type, based on *packet forwarding distance* [111][56] measures and the derived *dissimilarity* [236][235] indices by node size. Multi-greedy formulations are not explicitly presented in this section, but the design demonstrations in the simulated environments of a VANET and a MANET reveal how this may be beneficially performed.

4.2.1 Background: topology characterizing and dissimilarity indications

The characterizing of MANET topologies provides context-aware means by which packet routing protocols may perform under different environments [56][215][203]. The applicative characterizing of complex networks contribute to solving real-world problems including social

collaborations, genetics, communication interactions, etc. [235]. For example, in a social or computer network environment, communicating node pairs can be characterized based on the dissimilarity mapping of their message exchange rates [237]. The study in [56] showed that a path-search method can improve its performance over a previously unknown environment through exploration learning of the underlying topology or graph type. In the illustration given, the *Delaunay* and the *random graph* types are shown to be differentiable by the averages and histogram patterns of their respective *arc length* measures. The authors showed that the *depth-first search* (DFS) method and its *positional* (P-DFS) and *directional* (D-DFS) variants exhibit different performance efficiencies over the two graph types. The finding in [56] is corroborated by the study in [102] which also showed the forwarding methods of GREEDY (i.e., P-DFS) and CR (i.e., D-DFS) exhibit performance differentiation over the Delaunay and the random graphs. In [215] several topology-characterizing metrics are described including clustering coefficients, degree distributions, degree centralities, dissimilarity, etc. The notion of *dissimilarity* is also known as *distance* in the sense of some quantifiable difference(s) between the objects of comparison [238][239]. Thus, characterization metrics have included lexical distance, trust propagation distance, geographic distance, discrete distance, bottleneck distance, Minkowski distance, etc., [237][239]. The Minkowski distance with $p = 2$ is equivalent to the Euclidean distance, which is a prime measure in geographical routing computations. Dissimilarity characterizing to distinguish between geometrical objects (points, lines, polygons, or polyhedrons) is discussed in [239]. When not obvious in the objects of interest, the features for dissimilarity comparison can also be acquired from relevant data using some *learning* methods. In [238], for example, a review of approaches to dissimilarity learning by matrix spaces and vectors is presented.

4.2.2 Methodology

The next-hop *link* measures of *greedy packet forwarding distance* [111] are normally generated in geographic routing. Consider a networking environment that exhibits spatiotemporally varying node sizes. For each determined node-size, observe the dissimilarities in the average packet forwarding distance generated using different greedy methods from a set. Also observe the packet delivery performance, i.e., PDR, hopcount, delay, etc., of each greedy method for each node-size or dissimilarity index. The leveraging of these observations forms the basis of the multi-greedy efficient forwarding scheme using packet forwarding distance.

4.2.2.1 Context awareness metrics

1) The packet forwarding distance metric

A node's *link* to each of its neighbours in the progress region is a potential packet forwarding distance (see Figure 1.2). Thus, if a node u directly sends packets to a neighbour v_i , the related Euclidean *length* measure is a packet forwarding distance [180]. Designate the occurrence of a *greedy packet forwarding distance* as ${}^2d_{GF}$, defined to be the weight in terms of the length measure of the concerned link or edge such that based on Equations 2.15 and 2.29 we have

$$d_{GF}(e_{u,v_i}(t)) = \text{dist}(u, v_i)(t). \quad (4.7)$$

The packet forwarding distance generated in geographic routing generally satisfies the properties specified for a dissimilarity metric [239], which are (i) $\text{dist}(a, a)(t) = 0$, (ii) $\text{dist}(a, b)(t) = 0$ implies $a = b$, and (iii) $\text{dist}(a, b)(t) + \text{dist}(a, c)(t) \geq \text{dist}(b, c)(t)$.

Using (4.7), the set

² In [240], h_{GF} designates the hopcount, i.e., the number of relays traversed by packet(s), in greedy forwarding.

$$D_{GF} = \{d_{GF_i}(t) \mid i = 1, 2, 3, \dots, |D_{GF}|\} \quad (4.8)$$

contains unique $d_{GF}(t)$ elements, having different time stamps t , even when some are of equal length values. For convenience, we shall discontinue including the time t on the terms. The average greedy packet forwarding distance is described by:

$$\bar{d}_{GF} = \frac{1}{|D_{GF}|} \sum_{i=1}^{|D_{GF}|} d_{GF_i}. \quad (4.9)$$

2) The dissimilarity metric

For any two entities a_i and a_j possessing comparable structural features, the relevant dissimilarity measure may take on a value from 0 (no dissimilarity) to a positive number of any magnitude, i.e., $[0, \infty)$. The higher the value, the more the dissimilarity distinction between the two objects. This is unlike the ‘similarity’ measure that varies between the range $[0,1]$, where 0 means no similarity but values tending to 1 imply close similarity. Note the distinction between ‘feature distance’ or ‘dissimilarity’ that distinguishes the comparative structural difference between any two objects of interest, and the ‘geographical distance’ that describes the length between any two locations. It happened that we employ the Euclidean distance metric for both measures in this study. The dissimilarity metric that is based on Euclidean distance measure concerning n pairwise attributes of objects a_i and a_j is defined as [215] [238]:

$$D(a_i, a_j) = \sqrt{\sum_{k=1}^n (a_{ik} - a_{jk})^2}. \quad (4.10)$$

Since we shall be dealing with a single attribute, i.e., the average packet forwarding distance, Equation 4.10 reduces to finding the difference between any two given average measures:

$$D(a_i, a_j) = |a_i - a_j| \quad (4.11)$$

Equation 4.10 especially computes the dissimilarity value for the pairwise average packet forwarding distance of the employed greedy methods, e.g., between GREEDY and ELLIPSOID.

3) The differential factor metric

Let $b^{(k)}$ be the dissimilarity value obtainable for a k^{th} node size of a network, and $b^{(m)}$ that of the peak size, $0 \leq k \leq m$, then the differential factor is:

$$df(b^{(k)}, b^{(m)}) = \frac{b^{(m)}}{b^{(k)}} \quad (4.12)$$

The differential factor represents the difference between two dissimilarity values. High values of differential factors reflect a MANET environment where multi-greedy forwarding is advantageous.

4.2.2.2 Topology dissimilarity metric for GREEDY and ELLIPSOID methods

For a networking scenario that exhibits varied node sizes, as an example, we specify the related packet forwarding distance *averages* for the ELLIPSOID [57] and GREEDY [95] methods. The GREEDY method is popularly used in geographic routing for its efficient PDR performances, but the ELLIPSOID method is also noted in the same perspective [111] [102] [57]. Invariably, with the two methods, we want to answer the following questions.

- (i) What relationship exists between the greedy packet forwarding distance and the node size of a MANET?
- (ii) How do the routing protocol's PDR performances relate to the relationship mentioned in (i) above?

Based on the greedy step illustration for node u in Figure 4.8, the next relay neighbour choice of the GREEDY forwarding method will be v_1 having the distance $dist(v_1, d)$ that is apparently the least to the destination in comparison to those of v_2 or v_3 . On the other hand, the ELLIPSOID method's choice will be v_2 because the distance $(dist(u, v_2) + dist(v_2, d))$ is apparently the least compared to those of v_1 or v_3 . Note that the ELLIPSOID's neighbour choice v_2 is closer to the forwarding node u in comparison to v_1 of the GREEDY metric, which generally shows $dist^{(ELLIPSOID)}(u, v) \leq dist^{(GREEDY)}(u, v)$ in a greedy forwarding step.

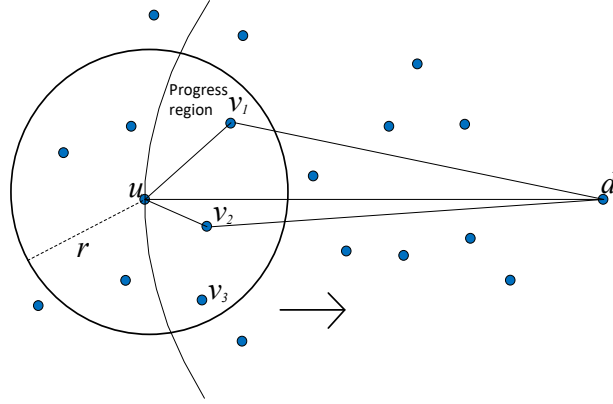


Figure 4.8: GREEDY (v_1) and ELLIPSOID (v_2) methods' next-hop choice-making. A geographic packet routing protocol, such as GPSR [30][233], can be enhanced to periodically sample packet forwarding distance dissimilarity values for the characterizing of the prevailing network topology. Using Equation 4.9, let $\bar{d}_{GF}^{(ELLIPSOID)}$ represent the average of the packet forwarding distance sampling for the ELLIPSOID [57] method usage in a network, and the corresponding $\bar{d}_{GF}^{(GREEDY)}$ represent that of GREEDY [95]. To avoid mix up of the notation 'D' in Equations 4.8 and 4.11, we specify $ddist(.)$ as the dissimilarity metric between the two methods' average packet forwarding distance as:

$$\begin{aligned}
& ddist\left(\bar{d}_{GF}^{(ELLIPSOID)(k)}, \bar{d}_{GF}^{(GREEDY)(k)}\right) \\
&= \left| \bar{d}_{GF}^{(ELLIPSOID)(k)} - \bar{d}_{GF}^{(GREEDY)(k)} \right|
\end{aligned} \tag{4.13}$$

which also takes into account the k^{th} node size measures of the concerned MANET.

The associated differential factor metric, based on Equations 4.12 and 4.13, is

$$\begin{aligned}
& df\left(\frac{ddist\left(\bar{d}_{GF}^{(ELLIPSOID)(k)}, \bar{d}_{GF}^{(GREEDY)(k)}\right)}{ddist\left(\bar{d}_{GF}^{(ELLIPSOID)(m)}, \bar{d}_{GF}^{(GREEDY)(m)}\right)}, \right) \\
&= \left| \bar{d}_{GF}^{(ELLIPSOID)(m)} - \bar{d}_{GF}^{(GREEDY)(m)} \right| / \left| \bar{d}_{GF}^{(ELLIPSOID)(k)} - \bar{d}_{GF}^{(GREEDY)(k)} \right|
\end{aligned} \tag{4.14}$$

The foregoing topology-characterizing and multi-greedy forwarding models have involved only two greedy metrics, namely ELLIPSOID [57] and GREEDY [95]. Other methods, such as MFR, NFP, etc., may as well have been involved in any number of combinations.

4.2.2.3 The multi-greedy optimization metric

Consider that for a particular node size k , a set $\{x_1, \dots, x_n\}$ of greedy metrics are applied in multi-greedy forwarding. That is, for a k^{th} node-size or rather the associated dissimilarity value $b^{(k)}$ and based on Equations 3.1, the multi-greedy function that continually selects an x_j method to adaptive packet forwarding for optimal PDR performance is:

$$\begin{aligned}
& \max_{x_j \in \mathcal{X}} y_{b^{(k)}} = \left(f(x)_{\max_{PDR}} \right)_{b^{(k)}} \\
&= \left(f(x_1, \dots, x_n)_{\max_{PDR}} \right)_{b^{(k)}}
\end{aligned}$$

$$= (V_1^n x_{j(max_{PDR})})_{b(k)} \quad (4.15)$$

Note that, as in Equation 4.15, we may optimize for other protocol performance measures, such as *minimize*-hopcount. Moreover, we may define trade-offs.

4.2.3 Simulation parameters

The NS-3 with the GPSR [233] protocol platforms were used in simulating the VANET and MANET routing environments. The simulation parameters are shown in Table 4.2. The VANET data, of Krauss' mobility model, together with the GPSR code that we used are obtainable from [234][233]. The VANET topology is an example of MANETs that characteristically vary widely in node size [189]. The environment was varied with a range of 30-110 node sizes which supported the effective demonstration of the topology dissimilarity phenomenon. Besides, the dissimilarity characteristic of the greedy packet forwarding distance measure was validated using a Random Way Point (RWP) MANET mobility model having 20-560 node sizes. The RWP model was generated using BonnMotion [232]. The node speed for both the VANET and MANET conditions was 0-15m/s. The simulation time was 200s. The ELLIPSOID [57] and GREEDY [95] packet forwarding metrics were implemented in the GPSR packet routing protocol. Appendix C shows parts of the NS-3 [59][222][235] C++code for the multi-greedy scheme using packet forwarding distance and topology characterizing.

Table 4.2: Simulation Parameters for Topology Characterizing

Parameter	Value
NS-3 simulator	v. 3.23
Routing protocol	GPSR
No. of nodes	30-110 (Krauss), 20-560 (RWP)
Simulation area	1100m ² (VANET), 2000m ² (MANET)
Simulation time	200s
Data packet size	512 bytes
Traffic application	CBR/UDP unicast
Transmission range	280m
Node speed	0 – 15m/s

4.2.4 Results and discussion

The simulation-generated *packet forwarding distance* measures were collated by node size and averages computed for both the VANET and MANET cases. The commensurate PDR outcomes displayed instability in patterns which necessitates efficiency improvement by multi-greedy forwarding using packet forwarding distance dissimilarity indices. Similar to the PDR, the

unevenness of the hopcount and delay results suggest the need for a multi-greedy efficient forwarding.

4.2.4.1 The VANET scenario results

Figure 4.9 shows the averages of the greedy packet forwarding distance measures for both the ELLIPSOID and the GREEDY methods against different node sizes in the simulated VANET environment. The linear trendlines of the outcomes for the two methods depict a correlation between the average packet forwarding distance and changes in node size.

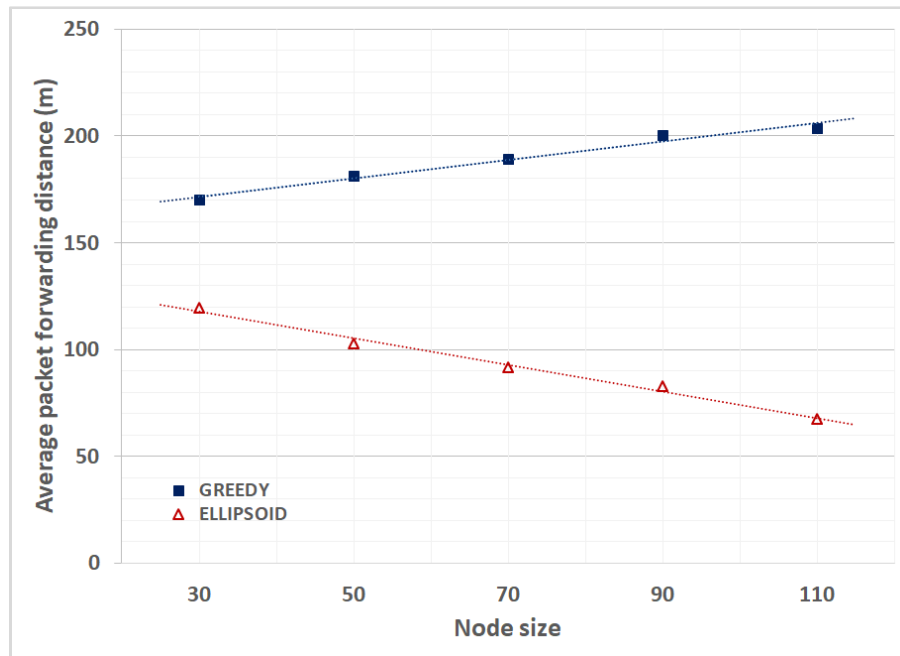


Figure 4.9: Packet forwarding distance averages by node size (VANET case).

The charts in Figure 4.10 shows the results for the MANET environment, which reflects outcomes that have parallel patterns as those of the VANET case (Figure 4.9). This suggests a generalizability in the topological characteristics of the greedy packet forwarding distance by network node size, which exhibits a correlation. Another observation is that the average packet forwarding distance for ELLIPSOID decreases while that of GREEDY increases with increasing node size. It is apparent that a pairwise comparison of the average packet forwarding distance by both methods, as defined in section 4.2.2, is of the form: $\bar{d}_{GF}^{(ELLIPSOID)(k)} \leq \bar{d}_{GF}^{(GREEDY)(k)}$.

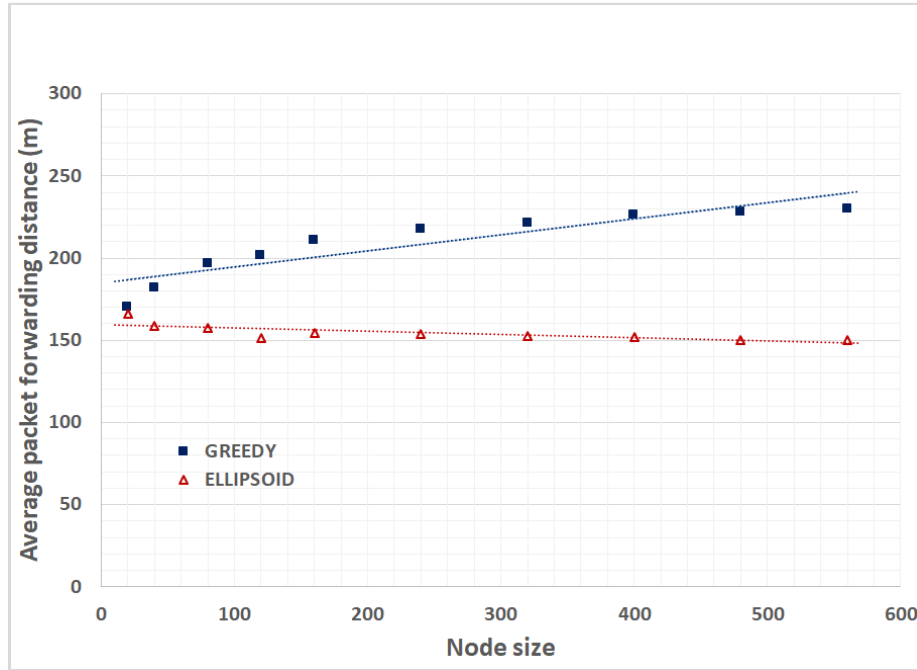


Figure 4.10: Packet forwarding distance averages with trendlines (MANET case).

The performance pattern of each method, such as GREEDY in Figure 4.9 or 4.10, reflects a dissimilarity in the consecutive average packet forwarding distance by node sizes. However, the dissimilarity measures of our focus for supporting multi-greedy forwarding are the pairwise differences between the ELLIPSOID and the GREEDY methods at each node-size level. The

differential factor values in Table 4.3 gives a vivid picture of the dissimilarity characteristics of the ELLIPSOID and the GREEDY methods' packet forwarding distance in the simulated VANET. The multi-greedy application would be advantageous in this environment. This is shown by the distinctiveness of the differential factor values having significant margins of variance one from the other.

Table 4.3: Topology character of the VANET environment

Node size (k)	Average packet forwarding distance		³ Dissimilarity value ($b^{(k)}$)	⁴ Differential factor (df)
	ELLIPSOID	GREEDY		
30	119.57	169.87	50.30	2.70
50	103.08	181.00	77.92	1.75
70	91.69	189.24	97.55	1.39
90	83.09	200.14	117.05	1.16
110	67.39	203.38	135.99	1.00

$$^3 b^{(k)} = \left| \bar{d}_{GF}^{(ELLIPSOID)(k)} - \bar{d}_{GF}^{(GREEDY)(k)} \right|$$

$$^4 df(b^{(k)}, b^{(m)}) = \frac{b^{(m)}}{b^{(k)}}, b^{(m)} = 135.99.$$

The charts in Figure 4.11 show the average dissimilarity values against the node sizes for the ELLIPSOID and the GREEDY methods (taken from Table 4.3) in the VANET case. It also shows PDR performance capabilities for the two methods. The solid gradient line connects the dissimilarity points, while the dashed lines in the lower part show the commensurate PDR measures. For example, at the node size of 70, the PDR for the ELLIPSOID method is lower at 36% compared to that of GREEDY at 54%. But at the node size of 90, the PDR of ELLIPSOID is higher at 67% compared to the 53% of the GREEDY method. Therefore, with this knowledge, a routing protocol can adaptively switch the greedy metrics based on each method's optimal PDR performance capability in relation to the dissimilarities or node-size variations.

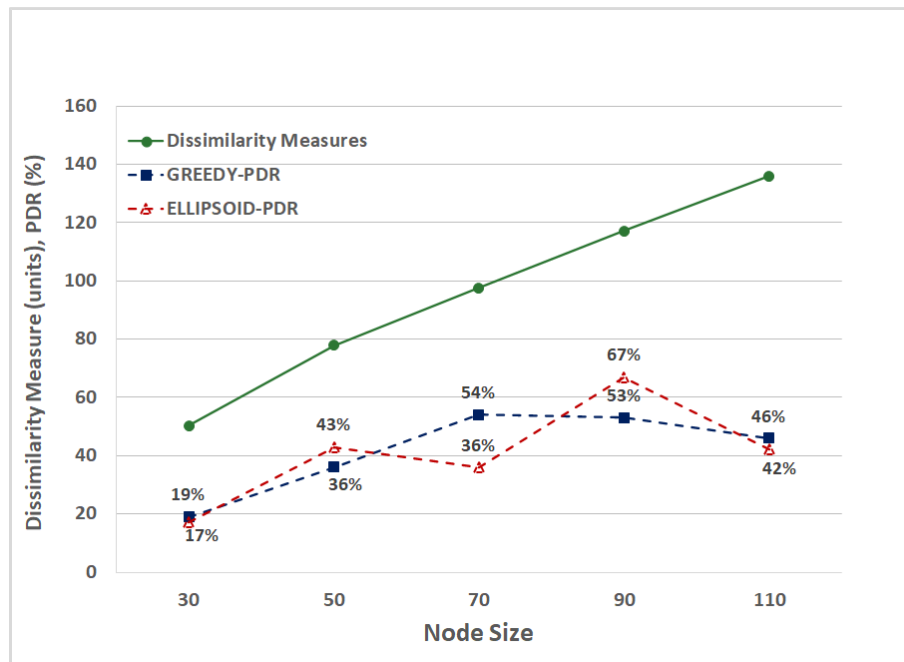


Figure 4.11: Packet forwarding distance dissimilarity distribution and PDR averages (VANET)

Figures 4.12 and 4.13 respectively show the averages of the hopcount and delay outcomes in the VANET scenario. As expected, GREEDY performed with a positive lower hopcount in comparison to ELLIPSOID for all node sizes. Nevertheless, regarding node sizes 50 and 90 where the PDRs of ELLIPSOID are higher in comparison to that of GREEDY (Figure 4.11), a hopcount trade-off may be implemented in a multi-greedy forwarding formulation. The average delay (Figure 4.13) for both methods at node-size 90 is at about the same level of 15ms, but ELLIPSOID has the edge with its 67% PDR against 53% of GREEDY's. Therefore, generally in VANET scenarios of this sort, the dissimilarity characterizing of topology based on the average packet forwarding distance comparisons offers potential support for an efficient multi-greedy scheme involving the ELLIPSOID and GREEDY methods.

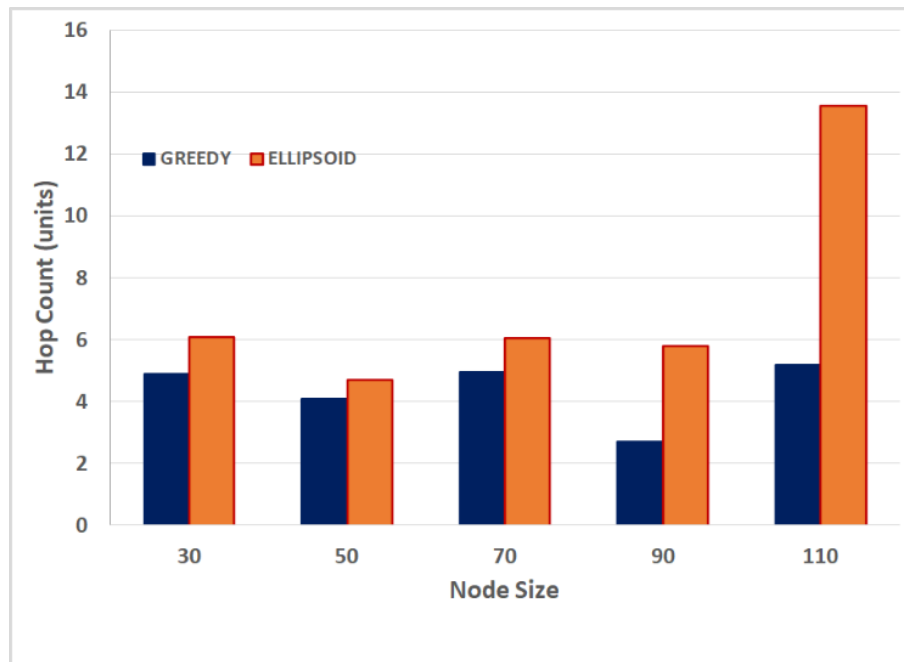


Figure 4.12: Average hopcount (VANET case)

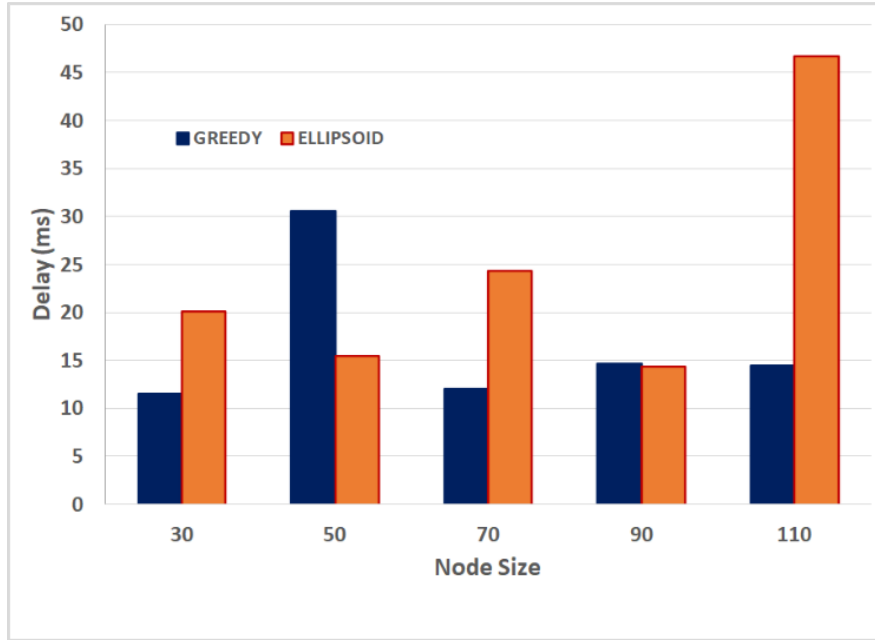


Figure 4.13: Average delay (VANET case)

4.2.4.2 The MANET scenario results

Table 4.4 shows the computed dissimilarity values and the corresponding differential factors for the simulated MANET scenario. The differential factor values of 1.15, 1.08, 1.02, and 1.00 corresponding to 320, 400, 480, and 560 node sizes respectively show that the average packet forwarding distance has little comparative differences at high node size levels. For example, for the node-sizes 480 and 560, the corresponding dissimilarity differential factor values of 1.02 and 1.00 have a low variance of 0.02. Thus, the multi-greedy formulation application may not be so beneficial in this MANET, especially at high node-sizes.

Table 4.4: Topology character of the MANET environment

Node Size (k)	Average packet		Dissimilarity	Differential
	forwarding distance		Value (b)	Factor (df)
	ELLIPSOID	GREEDY		
20	166.09	170.27	4.18	19.11
40	158.76	181.94	23.18	3.45
80	157.50	196.62	39.12	2.04
120	151.25	201.75	50.50	1.58
160	154.61	210.75	56.14	1.42
240	153.62	217.50	63.88	1.25
320	152.31	221.49	69.18	1.15
400	152.16	226.12	73.96	1.08
480	150.11	228.20	78.09	1.02
560	150.28	230.18	79.90	1.00

Figure 4.14 reflects the MANET case dissimilarity values (taken from Table 4.4) shown by the solid line chart, while the dashed lines are the corresponding PDR measures for both ELLIPSOID and GREEDY at the respective node sizes. In comparison to the VANET case (Figure 4.11) which appears linear, the gradient of the dissimilarity values of the MANET case (Figure 4.14) is more of a curve which quickly tends to an asymptotic level. The PDR of $\pm 90\%$ at all node-sizes except at 100, for both ELLIPSOID and GREEDY methods, is quite high in this network. But the comparative difference is high at the node-size 400 which shows a PDR of 96% for ELLIPSOID and 85% for GREEDY.

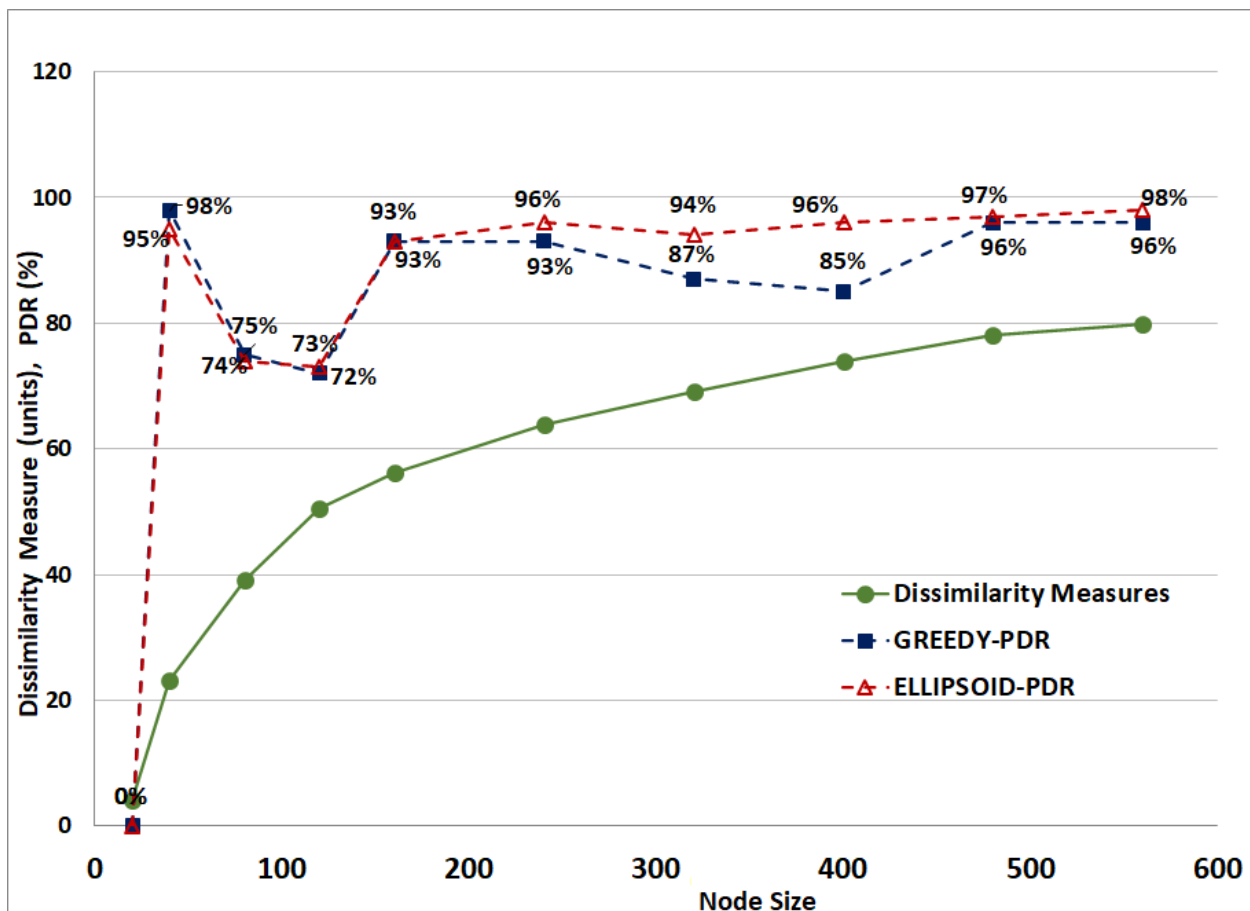


Figure 4.14: Packet forwarding distance dissimilarity distribution and PDR averages (MANET)

Generally, the outcomes suggest that a multi-greedy formulation involving the GREEDY and ELLIPSOID methods would have only marginal improvement effectiveness in this depicted MANET.

Figures 4.15 and 4.16 show the hopcount and delay measures for the ELLIPSOID and GREEDY forwarding at the different node sizes. As expected, the hopcount of GREEDY is lower in comparison to that of ELLIPSOID at the different node sizes. However, the delay of GREEDY is higher at node sizes 160 and 560. Based on the hopcount and delay that show unevenness in their comparative averages, a related multi-greedy formulation involving ELLIPSOID and GREEDY shall appropriately yield packet forwarding efficiency in the simulated MANET. Nevertheless, the topology characterizing method that was described in this section is most suitable for improving geographic greedy routing in highly dynamic environments, such as VANET and FANET, where node sizes vary significantly in space and time.

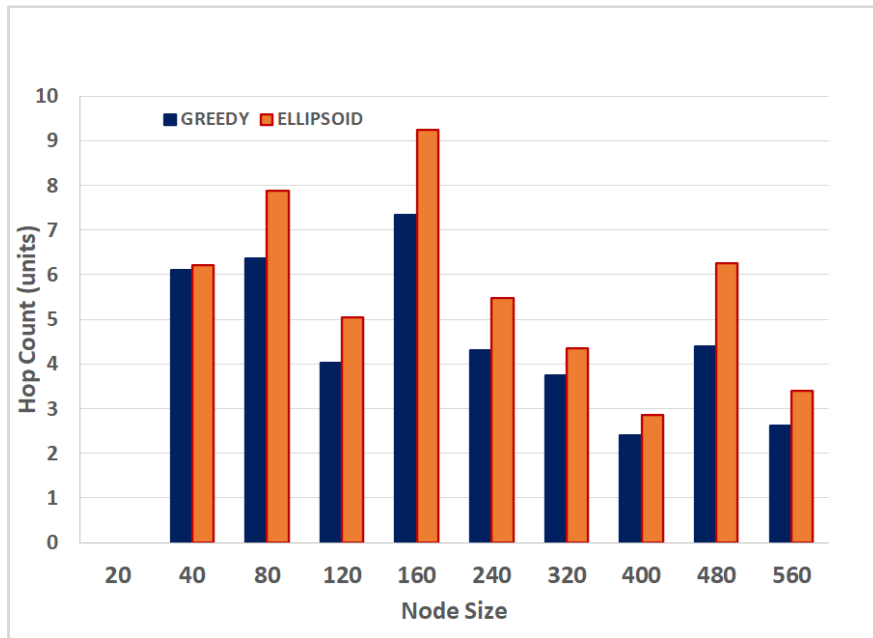


Figure 4.15: Average hopcount (MANET case)

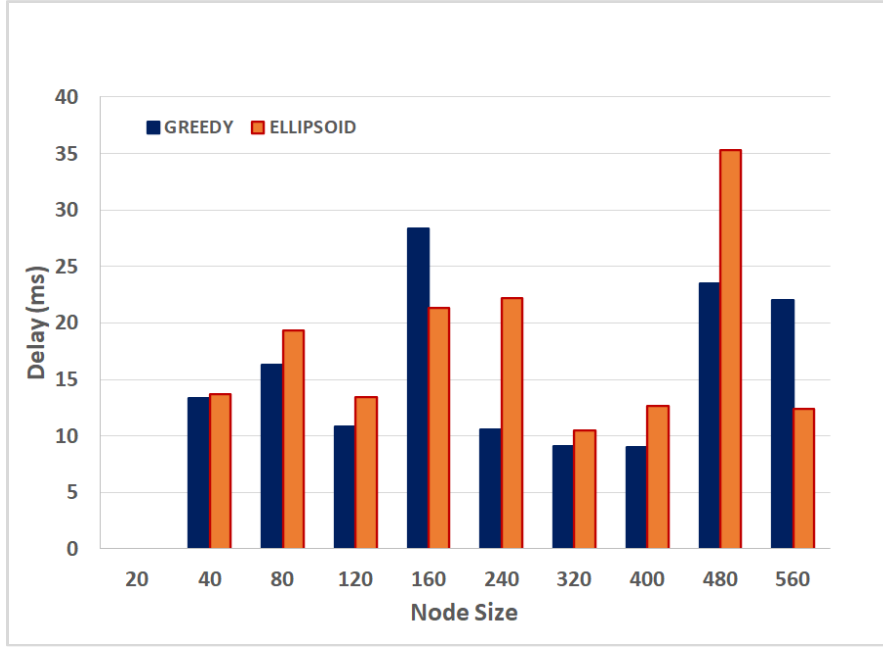


Figure 4.16: Average delay (MANET case)

Overall, the results of the VANET and the MANET scenarios show the generalizability of the topology characterizing approach using the greedy methods' *packet forwarding distance* and based on node size. The topology characterizing method can be applied over a wide range of networking environments and irrespective of the underlying graph type; in contrast to the approach in [56]. But, following the approach in [56], a *learning* method can practically discern the average greedy packet forwarding distance variations.

CHAPTER 5

5 CONCLUSIONS

This thesis addressed the question of whether multiple greedy metrics can be hybridized to perform efficient adaptive packet forwarding in geographic routing over fluctuating and varying MANET topologies. The spatiotemporal factors of node velocity, density, and wireless transmission vagaries [172][241] occur at various proportions to characterize MANET topologies. Researchers have resorted to hybrid routing protocol designs in addressing the challenges of varying dynamic MANETs (chapter 2). A *multi-greedy* approach to efficient progress-based packet forwarding is defined (chapter 3) with two example designs and sample formulations (chapter 4). A design is associated with a context-aware scheme while a formulation is related to a specific MANET environment. The evaluated multi-greedy formulations demonstrated improvement capacities in the standard geographic routing performance measures of packets delivery rate, hopcount, and delay.

5.1 Multi-greedy forwarding efficiency over highly dynamic MANET topologies

The multi-greedy packet forwarding model (chapter 3) describes a hybrid approach that leverages the capabilities of existing geographic greedy metrics for context-aware switching performance over varying MANET topologies. The formulation of a multi-greedy method, based on a context-aware design, depends on the application environment or packet delivery performance requirements. The two examples of multi-greedy packet forwarding schemes that were demonstrated (chapter 4) involves the GREEDY [95], ELLIPSOID [57], and GREEDY-COMPASS [102] metrics; together with context measures of (i) IPFIX packet flow reports [62], and (ii) dissimilarity indices of the packet forwarding distance [215][238] [111] which characterize topology by node size. The multi-greedy method is apt to improve packet forwarding in environments of highly dynamic topology changes such as VANET and FANET; while

MANETs with mild topology variations (such as in WSNs) have only little benefit using multi-greedy forwarding. Greedy failure is a major issue in geographic packet routing [104][30][10][1]. In a failed instance, a routing protocol engages its recovery module to temporarily re-route packets; and thereafter it can resume the multi-greedy forwarding as in the conventional method. Greedy failure-minimization approaches such as hole bypassing in energy-aware schemes that isolate energy-depleted nodes from the neighbour tables [154][155] are also applicable to multi-greedy forwarding. Related to greedy failure is network partitioning, i.e. when no link or path exists across a divide of segments [26][22][84]. In the case of a network partitioning, a DTN method is temporarily engaged to ‘store-and-carry’ packets while the normal routing is suspended. AROD [22] and GeoDTN+Nav [31] are examples of hybrid protocols having the DTN module. The GeoDTN+Nav protocol, like any other geographic routing method, may employ the multi-greedy scheme as its normal forwarding module while its DTN module addresses occurrences of network partitions.

5.2 Benefits and constraints of the multi-greedy scheme

Generally, the multi-greedy approach preserves the geographic method’s fundamental property of localized greedy decision-making. The comprised greedy forwarding metrics are without modification in a multi-greedy composition; only switching of the methods for adaptive performance is involved. Thus, both the conventional forwarding method and the multi-greedy approach exhibit similar geographic packet routing issues. The following are specific benefits and constraints of the multi-greedy forwarding approach.

- (i) The multi-greedy scheme can be deployed on most standard geographic packet routing protocols including GPSR [30], SLURP [76], ARP [48], GPCR [107], GeoDTN+Nav [31], etc.

The multi-greedy mechanism is easy and simple to deploy in place of the conventional method; except for its necessary coupling (albeit loose) with a context-aware module. When fitted with the multi-greedy mechanism, the fundamental character of a routing protocol remains unaltered as it handles any operational greedy metric alike.

(ii) Virtually all existing greedy forwarding metrics can be engaged in specific multi-greedy formulations. Therefore, apart from the classic metrics such as GREEDY [95] and ELLIPSOID [57], a multi-greedy method may be composed of other derived types such as GREEDY-COMPASS [102], RCOMPASS, WEIGHTEDRCOMPASS, GREEDYINSECTOR, RANDOMINSECTOR [133], etc. Moreover, greedy metrics of power- or energy-aware types such as *CPR* [145] and *NADV* [44][146], (See Equations 2.7 and 2.8), can be engaged in the multi-greedy scheme since each of these also have unique performance attributes in different environments. Likewise, methods that exonerate depleted nodes from performing greedy forwarding, such as ALBA [153] and HobyCan [154] can be engaged in the multi-greedy scheme. Excluded nodes are simply removed from the neighbour-table, whether in the conventional or the multi-greedy approach.

(iii) Context knowledge acquisition is a requirement in every multi-greedy forwarding design. It should be noted that conditional knowledge gathering is outside of the multi-greedy forwarding definition. Like other context-aware methods, the multi-greedy method engages with an external context-sensing module(s) from which it obtains the optimization conditional testing data [144]. A multi-greedy design can be realized with pertinent context-awareness schemes as are applicable in geographic hybrid protocols [69][14]. The overhead costs associated with exploration, learning, or acquiring environment knowledge in multi-greedy forwarding should also be tolerable [21][220] as it is in the extant context-aware routing protocols. The characteristics of an environment determines multi-greedy design specifics.

- (iv) The multi-greedy metrics demand an equivalent level of processing resources as it is in the conventional method. In comparison to the conventional approach, the multi-greedy method does not introduce a change to the forwarding algorithm's run-time, since only the conditional if-else statement is involved in switching the greedy metrics. Note that 'system-level context switching' is not involved in the multi-greedy scheme since a geographic protocol uniformly executes with whatever greedy metric is operational at any time (see Figure 3.1).
- (v) A multi-greedy formulation is transient, as it depends on which methods are suitably combined for a packet routing task in the prevailing state of a network. To maintain packet forwarding efficiency, a reformulation becomes necessary when an underlying topology has significantly metamorphosed. Thankfully, the reformulation process is seamless in an operational multi-greedy scheme, which is premised on context awareness and reallocating of the greedy metrics to the adaptive forwarding performances.
- (vi) Fundamentally, the multi-greedy method is operatable in 3-D environments [2][94]. Investigation of the multi-greedy approach was undertaken based on the 2-dimensional space. But converting to the 3-D forwarding in geographic routing is straightforward, which often only requires greedy distance computations that engage location coordinates as x - y - z instead of x - y [111][94][127][242][243][244].

5.3 Future works

The multi-greedy approach to efficient progress packet forwarding that is defined in this thesis provides the basis for implementation designs suitable to different geographic routing environments and requirements. In addition to the two multi-greedy designs shown in chapter 4, the following are identified areas for further investigation.

1) Multi-greedy forwarding using learning algorithms

Research direction in ad hoc networking, like several other disciplines, is currently involved with performance optimization approaches using machine learning and artificial intelligence tools [236]. An area to investigate is the machine learning *classification* of the packet forwarding distance which could characterize topologies at granular levels and achieve better multi-greedy performance results (see section 4.24).

2) MANET topology characterizing using extra features of the packet forwarding distance.

Characterizing MANET topologies using packet forwarding distance's 'length' attribute by node size is an incidental outcome of the multi-greedy design (section 4.2). However, other attributes of a link, such as duration, quality, etc., also influence packet forwarding [194]. Thus, further investigation should be carried out on characterizing MANET topologies using the packet forwarding distance with attributes such as *PIR* [245], *PRR* [88][98], etc.

3) Multi-greedy forwarding using different mobility rates and patterns

An investigation should be carried out on how mobility rates and patterns affect the dissimilarity indices in topology characterizing using packet forwarding distance (see section 4.2). The outcome could provide a way of topology characterizing by average node speed and underlying mobility patterns.

4) Multi-greedy forwarding using real-time protocol performance measures

A preliminary investigation shows the feasibility of multi-greedy forwarding efficiency improvement using real-time protocol performance awareness or feedback. This is similar to the IPFIX packets flow report for context determination, shown in chapter 4. Determining the protocol

performance measures' sampling cycle is critical to efficiency improvement as revealed in the outcomes of the preliminary investigation.

5) Beaconless multi-greedy forwarding

A multi-greedy forwarding method can also be designed to improve beaconless geographic protocols such as GeRaF [134][138], BLR [135], BOSS [139], etc., in which next-hops are elected by participating neighbour nodes. Beaconless routing involves reactive packet forwarding that attempts to minimize the overhead of neighbour-table maintenance. Traditionally, the greedy forwarding criteria in beaconless geographic routing have been based on the GREEDY metric. A multi-greedy approach to beaconless routing shall involve the efficiency leveraging of other methods, such as ELLIPSOID and GREEDY-COMPASS.

6) Characteristic of greedy metrics in multi-greedy optimization

The characteristics of greedy metrics that may implement multi-greedy geographic forwarding can be studied in future works. An example characteristic is the compatibility of greedy metric sets in packet forwarding optimization objectives such as maximizing PDR, minimizing or maximizing hopcount, etc.

7) Per-hop switching in multi-greedy forwarding

The feasibility of multi-greedy switching that is based on individual node contexts, can be proven in future designs. Greedy metric switching at the *macro* level of an entire networking area or segment thereof has been demonstrated in the example designs involving the context measures of IPFIX reports and the *packet forwarding distance* indices. The AMAHR framework [21] was an example MANET investigation regarding the challenges and feasibility of switching protocols at

the *micro* level of nodes' local contexts. However, the geographic routing's inherent local choice-making attribute should support using varied greedy forwarding metrics at intermediate nodes along a packet's path.

In retrospect, the multi-greedy forwarding model that is presented in this thesis focused on optimizing progress-based geographic routing over fluctuating and varying topologies through context-aware adaptive performance. The multi-greedy performance of a routing task involves engaging multiple greedy methods for adaptive assignment and re-assignments to efficient packet forwarding over spatiotemporally varying MANET topologies.

REFERENCES

- [1] C. Lemmon, S. M. Lui, and I. Lee, "Geographic Forwarding and Routing for Ad-hoc Wireless Network: A Survey," in *2009 Fifth International Joint Conference on INC, IMS and IDC*, Aug. 2009, pp. 188–195. doi: 10.1109/NCM.2009.80.
- [2] S. S. Lam and C. Qian, "Geographic Routing in d-Dimensional Spaces with Guaranteed Delivery and Low Stretch," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 663–677, Apr. 2013, doi: 10.1109/TNET.2012.2214056.
- [3] D. D. Perkins, H. D. Hughes, and C. B. Owen, "Factors affecting the performance of ad hoc networks," in *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, 2002, pp. 2048–2052 vol.4. doi: 10.1109/ICC.2002.997208.
- [4] J. Wu and P. Fan, "A Survey on High Mobility Wireless Communications: Challenges, Opportunities and Solutions," *IEEE Access*, vol. 4, pp. 450–476, 2016, doi: 10.1109/ACCESS.2016.2518085.
- [5] G. Oladeji-Atanda, "A greedy next-hop selection module for vehicular ad-hoc network routing using the A* algorithm method," Dec. 2014, Accessed: Jun. 09, 2021. [Online]. Available: <https://ubrisa.ub.bw/handle/10311/2035>
- [6] T. G. McGiffen, S. Beiker, and A. Paulraj, "Motivating Network Deployment: Vehicular Communications," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 22–33, Sep. 2017, doi: 10.1109/MVT.2017.2699699.
- [7] K. Peters, A. Jabbar, E. K. Çetinkaya, and J. P. G. Sterbenz, "A Geographical Routing Protocol for Highly Dynamic Aeronautical Networks," in *2011 IEEE Wireless Communications and Networking Conference*, Cancun, Quintana Roo: IEEE, Mar. 2011, pp. 492–497. doi: 10.1109/WCNC.2011.5779182.
- [8] W. Chen, R. K. Guha, T. J. Kwon, and J. Lee, "A Survey and Challenges in Routing and Data Dissemination in Vehicular Ad-hoc Networks," in *Proceedings of the 2008 IEEE International Conference on Vehicular Electronics and Safety*, Columbus, OH, USA: IEEE, Sep. 2008, pp. 328–333. doi: 10.1109/ICVES.2008.4640900.
- [9] K.-H. Cho and M.-W. Ryu, "A Survey of Greedy Routing Protocols for Vehicular Ad Hoc Networks," *Smart Computing Review*, vol. 2, no. 2, pp. 125–137, Apr. 2012, doi: 10.6029/smartcr.2012.02.003.
- [10] S. Yousefi, M. S. Mlousavi, and Fathy, "Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives," in *IEEE 6th International Conference on ITS Telecommunications Proceedings*, Chengdu, China: IEEE, Jun. 2006.
- [11] B. T. Sharef, R. A. Alsaqour, and Ismail Mahamod, "Vehicular communication ad hoc routing protocols: A survey," *ELSEVIER Science Direct - Journal of Network and Computer Applications*, vol. 40, pp. 363–396, 2013, doi: 10.1016/j.jnca.2013.09.008.
- [12] W. Zafar and B. M. Khan, "Flying Ad-Hoc Networks: Technological and Social Implications," *IEEE Technology and Society Magazine*, vol. 35, no. 2, pp. 67–74, Jun. 2016.
- [13] M. Y. Arafat and S. Moh, "Routing Protocols for Unmanned Aerial Vehicle Networks: A Survey," *IEEE Access*, vol. 7, pp. 99694–99720, 2019, doi: 10.1109/ACCESS.2019.2930813.
- [14] S. Kukliński and G. Wolny, "CARAVAN: Context-AwaRe Architecture for VANET," in *Mobile Ad-Hoc Networks: Applications*, X. Wang, Ed., InTech, 2011, pp. 125–148. [Online]. Available: <http://www.intechopen.com/books/mobile-ad-hoc-networks-applications/caravan-a-context-aware-architecturefor-vanet>
- [15] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, "SHARP: a hybrid adaptive routing protocol for mobile ad hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, Annapolis, MD, USA: ACM, 2003, pp. 303–314. doi: 10.1145/778415.778450.
- [16] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *Journal of Network and Computer Applications*, vol. 35, pp. 253–267, 2012.

- [17] D. Naboulsi and M. Fiore, "On the Instantaneous Topology of a Large-scale Urban Vehicular Network: The Cologne Case," in *Proceedings of the Fourteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, in MobiHoc '13. New York, NY, USA: ACM, 2013, pp. 167–176. doi: 10.1145/2491288.2491312.
- [18] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneş, and M. B. Yagoubi, "A survey on position-based routing protocols for Flying Ad hoc Networks (FANETs)," *Vehicular Communications*, vol. 10, pp. 29–56, Oct. 2017, doi: 10.1016/j.vehcom.2017.10.003.
- [19] J. Hong and D. Zhang, "TARCS: A Topology Change Aware-Based Routing Protocol Choosing Scheme of FANETs," *Electronics*, vol. 8, no. 3, p. 274, Mar. 2019, doi: 10.3390/electronics8030274.
- [20] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-Case Optimal and Average-case Efficient Geometric Ad-hoc Routing," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, in MobiHoc '03. New York, NY, USA: ACM, 2003, pp. 267–278. doi: 10.1145/778415.778447.
- [21] J. Hoebeke, I. Moerman, and P. Demeester, "Adaptive routing for mobile ad hoc networks," *EURASIP Journal on Wireless Communications and Networking*, 2012, doi: 10.1186/1687-1499-2012-126.
- [22] C. Liu and J. Wu, "Adaptive Routing in Dynamic Ad Hoc Networks," in *2008 IEEE Wireless Communications and Networking Conference*, Las Vegas, NV: IEEE, Apr. 2008, pp. 2603–2608. doi: 10.1109/WCNC.2008.457.
- [23] Ting-Chao Hou and Victor Li, "Transmission Range Control in Multihop Packet Radio Networks," *IEEE Transactions on Communications*, vol. 34, no. 1, pp. 38–44, Jan. 1986, doi: 10.1109/TCOM.1986.1096436.
- [24] F. Bai, Narayanan Sadagopan, and A. Helmy, "IMPORTANT: a framework to systematically analyze the Impact of Mobility on Performance of Routing Protocols for Adhoc Networks," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, Mar. 2003, pp. 825–835 vol.2. doi: 10.1109/INFCOM.2003.1208920.
- [25] C. Liu, R. Correa, X. Li, P. Basu, B. T. Loo, and Y. Mao, "Declarative Policy-Based Adaptive Mobile Ad Hoc Networking," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 3, pp. 770–783, Jun. 2012, doi: 10.1109/TNET.2011.2165851.
- [26] S. Zhao and D. Raychaudhuri, "Policy-Based Adaptive Routing in Mobile Ad Hoc Wireless Networks," in *2006 IEEE Sarnoff Symposium*, Mar. 2006, pp. 1–4. doi: 10.1109/SARNOF.2006.4534724.
- [27] Z. J. Haas, "A new routing protocol for the reconfigurable wireless networks," San Diego, CA, U.S.A.: IEEE, Oct. 1997. doi: 10.1109/ICUPC.1997.627227.
- [28] Z. Haas, M. Pearlman, and P. Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks." IETF INTERNET DRAFT, Jul. 2002. Accessed: Jan. 10, 2018. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-manet-zone-zrp-04>
- [29] F. Cadger, K. Curran, J. Santos, and S. Moffett, "A Survey of Geographical Routing in Wireless Ad-Hoc Networks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 621–653, Second 2013, doi: 10.1109/SURV.2012.062612.00109.
- [30] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, Boston, MA, USA: ACM, Aug. 2000, pp. 243–254. doi: 10.1145/345910.345953.
- [31] P.-C. Cheng, K. C. Lee, M. Gerla, and J. Härrri, "GeoDTN+Nav: Geographic DTN Routing with Navigator Prediction for Urban Vehicular Environments," *Mobile Netw Appl*, vol. 15, no. 1, pp. 61–82, Feb. 2010, doi: 10.1007/s11036-009-0181-6.

- [32] M. Chen, V. C. M. Leung, S. Mao, Y. Xiao, and I. Chlamtac, "Hybrid Geographic Routing for Flexible Energy-Delay Tradeoff," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 4976–4988, Nov. 2009, doi: 10.1109/TVT.2009.2025767.
- [33] S. Boussoufa-Lahlah, F. Semchedine, and L. Bouallouche-Medjkoune, "Geographic routing protocols for Vehicular Ad hoc NETWORKS (VANETs): A survey," *Vehicular Communications*, vol. 11, pp. 20–31, Jan. 2018, doi: 10.1016/j.vehcom.2018.01.006.
- [34] J. Bernsen and D. Manivannan, "Greedy Routing Protocols for Vehicular Ad Hoc Networks," in *2008 International Wireless Communications and Mobile Computing Conference*, Aug. 2008, pp. 632–637. doi: 10.1109/IWCMC.2008.109.
- [35] S. Ruhrup, "Theory and Practice of Geographic Routing," in *Ad Hoc and Sensor Wireless Networks: Architectures, Algorithms and Protocols*, Bentham Science, 2009.
- [36] M. Al-shugran, O. Ghazali, S. Hassan, K. Nisar, and A. S. M. Arif, "A qualitative comparison evaluation of the greedy forwarding strategies in Mobile Ad Hoc Network," *ELSEVIER Journal of Network and Computer Applications*, vol. 36, pp. 887–897, 2012, doi: 10.1016/j.jnca.2012.10.008.
- [37] R. Shirani, M. St-Hilaire, T. Kunz, Y. Zhou, J. Li, and L. Lamont, "The Performance of Greedy Geographic Forwarding in Unmanned Aeronautical Ad-hoc Networks," presented at the 2011 Ninth Annual Communication Networks and Services Research Conference, Ottawa, ON: IEEE, May 2011, pp. 161–166. doi: 10.1109/CNSR.2011.31.
- [38] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30–39, Dec. 2001, doi: 10.1109/65.967595.
- [39] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 3rd Edition*, 3rd edition. Cambridge, Mass: The MIT Press, 2009.
- [40] S. Giordano and I. Stojmenovic, "Position Based Routing Algorithms for Ad Hoc Networks: A Taxonomy," in *Ad Hoc Wireless Networking*, in Network Theory and Applications. Springer, Boston, MA, 2004, pp. 103–136. doi: 10.1007/978-1-4613-0223-0_4.
- [41] K. Katsaros, M. Dianati, and L. Le, "Effective Implementation of Location Services for VANETs in Hybrid Network Infrastructures," presented at the 2013 IEEE International Conference on Communications Workshops (ICC), Budapest: IEEE, Jun. 2013, pp. 521–525. doi: 10.1109/ICCW.2013.6649289.
- [42] A. Nayak and I. Stojmenovic, "Geographic Routing in Wireless Sensor and Actuator Networks," in *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, IEEE, 2010, pp. 95–125. doi: 10.1002/9780470570517.ch4.
- [43] T.-C. Hou and V. O. K. Li, "Transmission Range Control in Multihop Packet Radio Networks," *IEEE Transactions on Communications*, vol. COM-34, no. 1, Jan. 1986.
- [44] S. Lee, B. Bhattacharjee, and B. Suman, "A general framework for efficient geographic routing in wireless networks," *ELSEVIER Computer Networks*, vol. 54 (2010), pp. 844–861, doi: 10.1016/j.comnet.2009.09.013.
- [45] and and H. Y. and, "Link stability and route lifetime in ad-hoc wireless networks," in *Proceedings. International Conference on Parallel Processing Workshop*, Aug. 2002, pp. 116–123. doi: 10.1109/ICPPW.2002.1039720.
- [46] D. Son, A. Helmy, and B. Krishnamachari, "The effect of mobility-induced location errors on geographic routing in mobile ad hoc sensor networks: analysis and improvement using mobility prediction," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 233–245, Jul. 2004, doi: 10.1109/TMC.2004.28.
- [47] G. Lim, K. Shin, S. Lee, H. Yoon, and J. S. Ma, "Link stability and route lifetime in ad-hoc wireless networks," in *Proceedings. International Conference on Parallel Processing Workshop*, Aug. 2002, pp. 116–123. doi: 10.1109/ICPPW.2002.1039720.

- [48] V. C. Giruka and M. Singhal, "Angular Routing Protocol for Mobile Ad Hoc Networks," in *25th IEEE International Conference on Distributed Computing Systems Workshops*, Jun. 2005, pp. 551–557. doi: 10.1109/ICDCSW.2005.42.
- [49] T. T. Son, H. L. Minh, G. Sexton, and N. Aslam, "A novel encounter-based metric for mobile ad-hoc networks routing," *Ad Hoc Networks*, vol. 14, pp. 2–14, Mar. 2014, doi: 10.1016/j.adhoc.2013.10.012.
- [50] N. Nikaein, C. Bonnet, and N. Nikaein, "HARP - Hybrid Ad hoc Routing Protocol," presented at the International symposium on telecommunications, Teheran, Iran: Eurecom, Sep. 2001. [Online]. Available: <http://www.eurecom.fr/en/publication/682>
- [51] P. Costa and G. P. Picco, "Semi-Probabilistic Routing For Highly Dynamic Networks," in *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*, in Signals and Communication Technology. Springer, Boston, MA, 2006, pp. 193–213. doi: 10.1007/0-387-30394-4_14.
- [52] A. Moussaoui, F. Semchedine, and A. Boukerram, "A link-state QoS routing protocol based on link stability for Mobile Ad hoc Networks," *Journal of Network and Computer Applications*, vol. 39, no. Supplement C, pp. 117–125, Mar. 2014, doi: 10.1016/j.jnca.2013.05.014.
- [53] F. Theoleyre, R. Tout, and F. Valois, "New metrics to evaluate mobility models properties," in *2007 2nd International Symposium on Wireless Pervasive Computing*, Feb. 2007. doi: 10.1109/ISWPC.2007.342624.
- [54] Y. Ganjali and N. McKeown, "Routing in a highly dynamic topology," in *2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005.*, Sep. 2005, pp. 164–175. doi: 10.1109/SAHCN.2005.1557072.
- [55] S. Corson and J. Macker, "Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation consideration." IETF Network Working Group RFC, Jan. 1999. Accessed: Jan. 09, 2018. [Online]. Available: <https://www.ietf.org/rfc/rfc2501.txt>
- [56] P. Cucka, N. S. Netanyahu, and A. Rosenfeld, "Learning in navigation: goal finding in graphs," in *Studies in Pattern Recognition*, in Series in Machine Perception and Artificial Intelligence, no. Volume 25, vol. Volume 25. WORLD SCIENTIFIC, 1997, pp. 39–56. doi: 10.1142/9789812797728_0003.
- [57] K. Yamazaki and K. Sezaki, "A proposal of geographical routing protocols for location-aware services," *Electronics and Communications in Japan (Part I: Communications)*, vol. 87, no. 4, pp. 26–34, Apr. 2004, doi: 10.1002/ecja.10138.
- [58] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.
- [59] K. Katsaros, M. Dianati, and K. Roscher, "A Position-based Routing Module for Simulation of VANETs in Ns-3," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, in SIMUTOOLS '12. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 345–352. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2263019.2263074>
- [60] C. An, Y. Luo, and A. Timm-Giel, "Switching of Routing Algorithms in Wireless Networks for Fire Fighting," in *Mobile Networks and Management*, A. Timm-Giel, J. Strassner, R. Agüero, S. Sargento, and K. Pentikousis, Eds., in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Berlin, Heidelberg: Springer, 2013, pp. 258–270. doi: 10.1007/978-3-642-37935-2_20.
- [61] X. Xiang, Z. Zhou, and X. Wang, "Self-Adaptive On Demand Geographic Routing Protocols for Mobile Ad Hoc Networks," presented at the 26th IEEE International Conference on Computer Communications, Anchorage, AK: IEEE, pp. 2296–2300. doi: 10.1109/INFCOM.2007.268.

- [62] R. Hofstede *et al.*, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014, doi: 10.1109/COMST.2014.2321898.
- [63] G. Carneiro, P. Fortuna, and M. Ricardo, "FlowMonitor: a network monitoring framework for the network simulator 3 (NS-3)," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, in VALUETOOLS '09. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Oct. 2009, pp. 1–10. doi: 10.4108/ICST.VALUETOOLS2009.7493.
- [64] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Second IEEE Workshop on Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99*, New Orleans, Louisiana: IEEE, Feb. 1999, pp. 90–100. doi: 10.1109/MCSA.1999.749281.
- [65] A. Bujari, C. E. Palazzi, and D. Ronzani, "A Comparison of Stateless Position-based Packet Routing Algorithms for FANETs," *IEEE Transactions on Mobile Computing*, vol. 17, no. 11, pp. 2468–2482, Nov. 2018, doi: 10.1109/TMC.2018.2811490.
- [66] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying Ad-Hoc Networks (FANETs): A survey," *ELSEVIER Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, May 2013, doi: 10.1016/j.adhoc.2012.12.004.
- [67] A. H. Ho, Y. H. Ho, and K. A. Hua, "Handling high mobility in next-generation wireless ad hoc networks," *Int. J. Commun. Syst.*, vol. 23, no. 9–10, pp. 1078–1092, Sep. 2010, doi: 10.1002/dac.1056.
- [68] J. Z. Zhao and R. Govindan, "Understanding Packet Delivery Performance in Dense Wireless Sensor Networks," in *1st international conference on Embedded networked sensor systems*, Los Angeles, California, USA: ACM, Nov. 2003, pp. 1–13. doi: 10.1145/958491.958493.
- [69] B. A. Mahmood and D. Manivannan, "Position Based and Hybrid Routing Protocols for Mobile Ad Hoc Networks: A Survey," *Wireless Pers Commun*, vol. 83, no. 2, pp. 1009–1033, Jul. 2015, doi: 10.1007/s11277-015-2437-8.
- [70] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, in MobiCom '98. New York, NY, USA: ACM, 1998, pp. 85–97. doi: 10.1145/288235.288256.
- [71] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Proceedings. IEEE International Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century.*, Dec. 2001, pp. 62–68. doi: 10.1109/INMIC.2001.995315.
- [72] Z. J. Haas and M. R. Pearlman, "The performance of a new routing protocol for the reconfigurable wireless networks," presented at the 1998 IEEE International Conference on Communications., Atlanta, Georgia, U.S.A.: IEEE, Jun. 1998, pp. 156–160. doi: 10.1109/ICC.1998.682608.
- [73] S. Giannoulis, C. Katsanos, S. Koubias, and G. Papadopoulos, "A hybrid adaptive routing protocol for ad hoc wireless networks," in *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings.*, Sep. 2004, pp. 287–290. doi: 10.1109/WFCS.2004.1377726.
- [74] M. J.-N. and, "A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1415–1425, Aug. 1999, doi: 10.1109/49.779923.
- [75] R. Ghanadan, J. Gu, J. Hsu, G. Sadosuk, P. Khuu, and W. Gallagher, "An Efficient Intranet Networking Solution for Airborne Networks," in *Proceedings of the 2006 IEEE Conference on Military Communications*, in MILCOM'06. Piscataway, NJ, USA: IEEE Press, 2006, pp. 315–321. Accessed: Mar. 20, 2019. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1896579.1896628>
- [76] S.-C. M. Woo and S. Singh, "Scalable Routing Protocol for Ad Hoc Networks," *Wireless Networks*, vol. 7, no. 5, pp. 513–529, Sep. 2001, doi: 10.1023/A:1016726711167.

- [77] H. Takagi and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet Radio," *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 246–257, Mar. 1984, doi: 10.1109/TCOM.1984.1096061.
- [78] L. Blazevic, J.-Y. Le Boudec, and S. Giordano, "A location-based routing method for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 97–110, Mar. 2005, doi: 10.1109/TMC.2005.16.
- [79] M. Musolesi and C. Mascolo, "CAR: Context-Aware Adaptive Routing for Delay-Tolerant Mobile Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 2, pp. 246–260, Feb. 2009, doi: 10.1109/TMC.2008.107.
- [80] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Network*, vol. 8, no. 5, pp. 22–32, Sep. 1994, doi: 10.1109/65.313011.
- [81] A. K. Dey, "Understanding and Using Context," *Personal Ubi Comp*, vol. 5, no. 1, pp. 4–7, Feb. 2001, doi: 10.1007/s007790170019.
- [82] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *Mobile Netw Appl*, vol. 1, no. 2, pp. 183–197, Jun. 1996, doi: 10.1007/BF01193336.
- [83] C. M. S. Figueiredo, E. F. Nakamura, and A. A. F. Loureiro, "A Hybrid Adaptive Routing Algorithm for Event-Driven Wireless Sensor Networks," *Sensors (Basel)*, vol. 9, no. 9, pp. 7287–7307, Sep. 2009, doi: 10.3390/s90907287.
- [84] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, in SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 145–158. doi: 10.1145/1015467.1015484.
- [85] M. Musolesi, S. Hailes, and C. Mascolo, "Adaptive routing for intermittently connected mobile ad hoc networks," in *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, Jun. 2005, pp. 183–189. doi: 10.1109/WOWMOM.2005.17.
- [86] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," in *Proceedings of the conference on Communications architectures, protocols and applications*, in SIGCOMM '94. London, United Kingdom: Association for Computing Machinery, Oct. 1994, pp. 234–244. doi: 10.1145/190314.190336.
- [87] F. Cadger, K. Curran, J. Santos, and S. Moffet, "Location and Mobility-Aware Routing for Improving Multimedia Streaming Performance in MANETs," *Wireless Pers Commun*, vol. 86, no. 3, pp. 1653–1672, Feb. 2016, doi: 10.1007/s11277-015-3012-z.
- [88] M. Z. Zamalloa, K. Seada, B. Krishnamachari, and A. Helmy, "Efficient Geographic Routing over Lossy Links in Wireless Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 4, no. 3, p. 12:1-12:33, Jun. 2008, doi: 10.1145/1362542.1362543.
- [89] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric Ad-hoc Routing: Of Theory and Practice," in *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, in PODC '03. New York, NY, USA: ACM, 2003, pp. 63–72. doi: 10.1145/872035.872044.
- [90] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM)," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, in MobiCom '98. New York, NY, USA: ACM, 1998, pp. 76–84. doi: 10.1145/288235.288254.
- [91] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *Springer Wireless Networks*, vol. 6, no. 4, pp. 307–321, Sep. 2000, doi: 10.1023/A:10191061419.
- [92] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 48–57, Feb. 2001, doi: 10.1109/98.904899.

- [93] K. R. Chowdhury and M. D. Felice, "Search: A routing protocol for mobile cognitive radio ad-hoc networks," *Computer Communications*, vol. 32, no. 18, pp. 1983–1997, Dec. 2009, doi: 10.1016/j.comcom.2009.06.011.
- [94] A. E. Abdallah, T. Fevens, and J. Opatrny, "High delivery rate position-based routing algorithms for 3D ad hoc networks," *Computer Communications*, vol. 31, no. 4, pp. 807–817, Mar. 2008, doi: 10.1016/j.comcom.2007.10.037.
- [95] G. G. Finn, "Routing and Addressing Problems in Large Metropolitan-Scale Internet-works," University of Southern California, Marina del Rey., Reports Research/Technical (143) ISI/RR-87-180, Mar. 1987.
- [96] J. Chen, M. Gerla, and Y.-Z. Lee, "TCP over Geo-routing for High Mobility: Vehicle Grids and Airborne Swarms," in *Ad-Hoc Networking*, K. Al Agha, Ed., in IFIP International Federation for Information Processing. Boston, MA: Springer US, 2006, pp. 1–15. doi: 10.1007/978-0-387-34738-7_1.
- [97] P. K. Sahu, E. H.-K. Wu, J. Sahoo, and M. Gerla, "BAHG: Back-Bone-Assisted Hop Greedy Routing for VANET's City Environments," *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, vol. 14, no. 1, pp. 199–213, Mar. 2013, doi: 10.1109/TITS.2012.2212189.
- [98] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy-efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, in SenSys '04. New York, NY, USA: ACM, 2004, pp. 108–121. doi: 10.1145/1031495.1031509.
- [99] I. Stojmenovic and X. Lin, "Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1023–1032, Oct. 2001, doi: 10.1109/71.963415.
- [100] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 11, pp. 1122–1133, Nov. 2001, doi: 10.1109/71.969123.
- [101] P. Bose and P. Morin, "Online Routing in Triangulations," in *Algorithms and Computation*, in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Dec. 1999, pp. 113–122. doi: 10.1007/3-540-46632-0_12.
- [102] P. Bose *et al.*, "Online routing in convex subdivisions," *Int. J. Comput. Geom. Appl.*, vol. 12, no. 04, pp. 283–295, Aug. 2002, doi: 10.1142/S021819590200089X.
- [103] E. Kranakis, H. Singh, and J. Urrutia, "Compass Routing on Geometric Networks," University of British Columbia, Vancouver, Canada, Aug. 1999. [Online]. Available: <http://www.cccg.ca/proceedings/1999/c46.pdf>
- [104] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, in DIALM '99. New York, NY, USA: ACM, 1999, pp. 48–55. doi: 10.1145/313239.313282.
- [105] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic Routing Made Practical," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, in NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 217–230. Accessed: Feb. 02, 2019. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1251203.1251219>
- [106] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "On the Pitfalls of Geographic Face Routing," in *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, in DIALM-POMC '05. New York, NY, USA: ACM, 2005, pp. 34–43. doi: 10.1145/1080810.1080818.
- [107] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, "Geographic Routing in City Scenarios," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 1, pp. 69–72, Jan. 2005, doi: 10.1145/1055959.1055970.

- [108] C. Liu and J. Wu, "Virtual-Force-Based Geometric Routing Protocol in MANETs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 4, pp. 433–445, Apr. 2009, doi: 10.1109/TPDS.2008.103.
- [109] F. Xi and Z. Liu, "Small World Topology-Aware Geographic Routing in Wireless Sensor Networks," in *2009 WRI International Conference on Communications and Mobile Computing*, Jan. 2009, pp. 116–120. doi: 10.1109/CMC.2009.229.
- [110] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing Without Location Information," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, in MobiCom '03. New York, NY, USA: ACM, 2003, pp. 96–108. doi: 10.1145/938985.938996.
- [111] G. S.-C. Kao, T. Fevens, and J. Opatrny, "Position-based routing on 3-D geometric graphs in mobile ad hoc networks," in *17th Canadian Conference on Computational Geometry (CCCG'05)*, Windsor, Ontario, 2005, pp. 81–91. [Online]. Available: <http://www.cccg.ca/proceedings/2005/>
- [112] C. Liu and J. Wu, "Efficient Geometric Routing in Three Dimensional Ad Hoc Networks," in *IEEE INFOCOM 2009*, Apr. 2009, pp. 2751–2755. doi: 10.1109/INFCOM.2009.5062225.
- [113] Y. Wang, C.-W. Yi, M. Huang, and F. Li, "Three-dimensional greedy routing in large-scale random wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 4, pp. 1331–1344, Jun. 2013, doi: 10.1016/j.adhoc.2010.10.003.
- [114] J. Zhou, Y. Chen, B. Leong, and P. S. Sundaramoorthy, "Practical 3D Geographic Routing for Wireless Sensor Networks," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, in SenSys '10. New York, NY, USA: ACM, 2010, pp. 337–350. doi: 10.1145/1869983.1870016.
- [115] A. E. Abdallah, T. Fevens, and J. Opatrny, "Hybrid Position-Based 3D Routing Algorithms with Partial Flooding," in *2006 Canadian Conference on Electrical and Computer Engineering*, May 2006, pp. 227–230. doi: 10.1109/CCECE.2006.277328.
- [116] A. E. Abdallah, "Smart Partial Flooding Routing Algorithms for 3D Ad Hoc Networks," *Procedia Computer Science*, vol. 94, pp. 264–271, Jan. 2016, doi: 10.1016/j.procs.2016.08.040.
- [117] T. Wang, Y. Cao, Y. Zhou, and P. Li, "A Survey on Geographic Routing Protocols in Delay/Disruption Tolerant Networks," *International Journal of Distributed Sensor Networks*, vol. 12, no. 2, p. 3174670, Feb. 2016, doi: 10.1155/2016/3174670.
- [118] L. H. O. Rios and L. Chaimowicz, "A survey and classification of A* based best-first heuristic search algorithms," in *Advances in Artificial Intelligence - SBIA 2010*, São Bernardo do Campo, Brazil: Springer, Oct. 2010, pp. 253–262.
- [119] I. Stojmenovic, M. Russell, and B. Vukojevic, "Depth first search and location based localized routing and QoS routing in wireless networks," in *Proceedings 2000 International Conference on Parallel Processing*, 2000, pp. 173–180. doi: 10.1109/ICPP.2000.876111.
- [120] R. E. Korf, "Artificial Intelligence Search Algorithms," in *Algorithms and Theory of Computation Handbook*, M. J. Atallah and M. Blanton, Eds., Chapman & Hall/CRC, 2010, pp. 22–22. Accessed: Jan. 04, 2019. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1882723.1882745>
- [121] A. Bertrand and M. Moonen, "Seeing the Bigger Picture: How Nodes Can Learn Their Place Within a Complex Ad Hoc Network Topology," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 71–82, May 2013, doi: 10.1109/MSP.2012.2232713.
- [122] S. Durocher, D. Kirkpatrick, and L. Narayanan, "On Routing with Guaranteed Delivery in Three-Dimensional Ad Hoc Wireless Networks," in *Distributed Computing and Networking*, S. Rao, M. Chatterjee, P. Jayanti, C. S. R. Murthy, and S. K. Saha, Eds., in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 546–557.
- [123] C. S. Chen, Y. Li, and Y.-Q. Song, "An exploration of geographic routing with k-hop based searching in wireless sensor networks," in *2008 Third International Conference on Communications and Networking in China*, Aug. 2008, pp. 376–381. doi: 10.1109/CHINACOM.2008.4685045.

- [124] G. Calinescu, "Computing 2-Hop Neighborhoods in Ad Hoc Wireless Networks," in *Ad-Hoc, Mobile, and Wireless Networks*, Springer, Berlin, Heidelberg, Oct. 2003, pp. 175–186. doi: 10.1007/978-3-540-39611-6_16.
- [125] M. Diop, C. Pham, and O. Thiaré, "2-hop neighborhood information for cover set selection in mission-critical surveillance with Wireless Image Sensor Networks," in *2013 IFIP Wireless Days (WD)*, Nov. 2013, pp. 1–7. doi: 10.1109/WD.2013.6686505.
- [126] M. Kumar, R. Tripathi, and S. Tiwari, "Two-hop information based gradient routing in industrial Wireless Sensor Networks," in *2013 4th International Conference on Computer and Communication Technology (ICCT)*, Sep. 2013, pp. 228–232. doi: 10.1109/ICCT.2013.6749632.
- [127] B. Shah and K.-I. Kim, "A Survey on Three-Dimensional Wireless Ad Hoc and Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 7, p. 616014, Jul. 2014, doi: 10.1155/2014/616014.
- [128] A. E. Abdallah, T. Fevens, and J. Opatrny, "Randomized 3D Position-based Routing Algorithms for Ad-hoc Networks," in *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking Services*, Jul. 2006, pp. 1–8. doi: 10.1109/MOBIQ.2006.340457.
- [129] M. Huang, F. Li, and Y. Wang, "Energy-Efficient Restricted Greedy Routing for Three Dimensional Random Wireless Networks," in *Wireless Algorithms, Systems, and Applications*, G. Pandurangan, V. S. Anil Kumar, G. Ming, Y. Liu, and Y. Li, Eds., in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 95–104.
- [130] Z. Wang, D. Zhang, O. Alfandi, and D. Hogrefe, "Efficient geographical 3D routing for Wireless Sensor Networks in smart spaces," in *2011 Baltic Congress on Future Internet and Communications*, Feb. 2011, pp. 168–172. doi: 10.1109/BCFIC-RIGA.2011.5733230.
- [131] R. Flury and R. Wattenhofer, "Randomized 3D Geographic Routing," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, Apr. 2008, pp. 834–842. doi: 10.1109/INFOCOM.2008.135.
- [132] R. NELSON and L. KLEINROCK, "The Spatial Capacity of a Slotted ALOHA Multihop Packet Radio Network with Capture," *IEEE Transactions on Communications*, vol. 32, no. 8, pp. 684–694, Jun. 1984, doi: 10.1109/TCOM.1984.1096124.
- [133] T. Fevens, I. T. Haque, and L. Narayanan, "Randomized Routing Algorithms in Mobile Ad Hoc Networks," in *Mobile and Wireless Communication Networks*, in IFIP International Federation for Information Processing. Springer, Boston, MA, 2005, pp. 347–357. doi: 10.1007/0-387-23150-1_30.
- [134] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 337–348, Oct. 2003, doi: 10.1109/TMC.2003.1255648.
- [135] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli, "BLR: beacon-less routing algorithm for mobile ad hoc networks," *Computer Communications*, vol. 27, no. 11, pp. 1076–1086, Jul. 2004, doi: 10.1016/j.comcom.2004.01.012.
- [136] J. A. Sánchez, R. Marín-Pérez, and P. M. Ruiz, "Beacon-Less Geographic Routing in Real Wireless Sensor Networks," *J. Comput. Sci. Technol.*, vol. 23, no. 3, pp. 438–450, May 2008, doi: 10.1007/s11390-008-9145-8.
- [137] S. Rührup and I. Stojmenović, "Contention-based georouting with guaranteed delivery, minimal communication overhead, and shorter paths in wireless sensor networks," in *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, Apr. 2010, pp. 1–9. doi: 10.1109/IPDPS.2010.5470408.
- [138] M. Zorzi and R. R. Rao, "Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 349–365, Oct. 2003, doi: 10.1109/TMC.2003.1255650.

- [139] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz, "BOSS: Beacon-less On Demand Strategy for Geographic Routing in Wireless Sensor Networks," in *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, Oct. 2007, pp. 1–10. doi: 10.1109/MOBHOC.2007.4428612.
- [140] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, "Contention-based forwarding for mobile ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 4, pp. 351–369, Nov. 2003, doi: 10.1016/S1570-8705(03)00038-6.
- [141] H. Liu, B. Zhang, H. T. Mouftah, X. Shen, and J. Ma, "Opportunistic routing for wireless ad hoc and sensor networks: Present and future directions," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 103–109, Dec. 2009, doi: 10.1109/MCOM.2009.5350376.
- [142] N. Chakchouk, "A Survey on Opportunistic Routing in Wireless Communication Networks," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2214–2241, Fourthquarter 2015, doi: 10.1109/COMST.2015.2411335.
- [143] P. M. Ruiz, V. Cabrera, J. A. Martinez, and F. J. Ros, "BRAVE: Beacon-less routing algorithm for vehicular environments," in *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*, Nov. 2010, pp. 709–714. doi: 10.1109/MASS.2010.5663798.
- [144] B.-L. Wenning, *Context-Based Routing in Dynamic Networks*. in Advanced Studies Mobile Research Center Bremen. Vieweg+Teubner Verlag, 2010. doi: 10.1007/978-3-8348-9709-1.
- [145] X. Li, N. Mitton, A. Nayak, and I. Stojmenovic, "Localized load-aware geographic routing in wireless ad hoc networks," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 7478–5482. doi: 10.1109/ICC.2012.6363642.
- [146] S. Lee, B. Bhattacharjee, and S. Banerjee, "Efficient Geographic Routing in Multihop Wireless Networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, in MobiHoc '05. New York, NY, USA: ACM, 2005, pp. 230–241. doi: 10.1145/1062689.1062720.
- [147] J. Kuruvila, A. Nayak, and I. Stojmenovic, "Progress and Location Based Localized Power Aware Routing for Ad Hoc and Sensor Wireless Networks," *International Journal of Distributed Sensor Networks*, vol. 2, no. 2, pp. 147–159, Mar. 2006, doi: 10.1080/15501320500259159.
- [148] I. Stojmenovic, "Localized network layer protocols in wireless sensor networks based on optimizing cost over progress ratio," *IEEE Network*, vol. 20, no. 1, pp. 21–27, Jan. 2006, doi: 10.1109/MNET.2006.1580915.
- [149] H. Huang, G. Hu, F. Yu, and Z. Zhang, "Energy-aware interference-sensitive geographic routing in wireless sensor networks," *IET Communications*, vol. 5, no. 18, pp. 2692–2702, Dec. 2011, doi: 10.1049/iet-com.2011.0154.
- [150] C. Gkantsidis, G. Goel, M. Mihail, and A. Saberi, "Towards Topology Aware Networks," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, May 2007, pp. 2591–2595. doi: 10.1109/INFCOM.2007.327.
- [151] Z. Zhao, D. Rosário, T. Braun, E. Cerqueira, H. Xu, and L. Huang, "Topology and Link quality-aware Geographical opportunistic routing in wireless ad-hoc networks," in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Jul. 2013, pp. 1522–1527. doi: 10.1109/IWCMC.2013.6583782.
- [152] T. Hieu, L. Dung, and B.-S. Kim, "Stability-Aware Geographic Routing in Energy Harvesting Wireless Sensor Networks," *Sensors*, vol. 16, no. 5, p. 696, May 2016, doi: 10.3390/s16050696.
- [153] C. Petrioli, M. Nati, P. Casari, M. Zorzi, and S. Basagni, "ALBA-R: Load-Balancing Geographic Routing Around Connectivity Holes in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 529–539, Mar. 2014, doi: 10.1109/TPDS.2013.60.
- [154] J. You, D. Lieckfeldt, F. Reichenbach, and D. Timmermann, "Context-Aware Geographic Routing for Sensor Networks with Routing Holes," in *2009 IEEE Wireless Communications and Networking Conference*, Apr. 2009, pp. 1–6. doi: 10.1109/WCNC.2009.4917638.

- [155] H. Huang, H. Yin, G. Min, J. Zhang, Y. Wu, and X. Zhang, "Energy-Aware Dual-Path Geographic Routing to Bypass Routing Holes in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 6, pp. 1339–1352, Jun. 2018, doi: 10.1109/TMC.2017.2771424.
- [156] A. Bertrand and M. Moonen, "Topology-aware distributed adaptation of Laplacian weights for in-network averaging," in *21st European Signal Processing Conference (EUSIPCO 2013)*, Sep. 2013, pp. 1–5.
- [157] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 164–194, Jun. 2005, doi: 10.1145/1089733.1089736.
- [158] R. Wattenhofer and A. Zollinger, "XTC: a practical topology control algorithm for ad-hoc networks," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, Apr. 2004, pp. 216–. doi: 10.1109/IPDPS.2004.1303248.
- [159] W. Si, B. Scholz, J. Gudmundsson, G. Mao, R. Boreli, and A. Zomaya, "On graphs supporting greedy forwarding for directional wireless networks," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 751–756. doi: 10.1109/ICC.2012.6363880.
- [160] J. Gao and L. Guibas, "Geometric algorithms for sensor networks," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, no. 1958, pp. 27–51, Jan. 2012, doi: 10.1098/rsta.2011.0215.
- [161] G. Xing, C. Lu, R. Pless, and Q. Huang, "On Greedy Geographic Routing Algorithms in Sensing-covered Networks," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, in *MobiHoc '04*. New York, NY, USA: ACM, 2004, pp. 31–42. doi: 10.1145/989459.989465.
- [162] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 16–28, Feb. 2001, doi: 10.1109/98.904895.
- [163] S. R. Das, R. Castaneda, Jiangtao Yan, and R. Sengupta, "Comparative performance evaluation of routing protocols for mobile, ad hoc networks," in *Proceedings 7th International Conference on Computer Communications and Networks (Cat. No.98EX226)*, Oct. 1998, pp. 153–161. doi: 10.1109/ICCCN.1998.998772.
- [164] T. Ducrocq, M. Hauspie, N. Mitton, and S. Pizzi, "On the Impact of Network Topology on Wireless Sensor Networks Performances: Illustration with Geographic Routing," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, May 2014, pp. 719–724. doi: 10.1109/WAINA.2014.118.
- [165] M. Fiore and J. Härrä, "The Networking Shape of Vehicular Mobility," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, in *MobiHoc '08*. New York, NY, USA: ACM, 2008, pp. 261–272. doi: 10.1145/1374618.1374654.
- [166] C. Bettstetter, "On the Minimum Node Degree and Connectivity of a Wireless Multihop Network," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, Lausanne, Switzerland: ACM, Jun. 2002, pp. 80–91. doi: 10.1145/513800.513811.
- [167] C. Bettstetter, "On the Connectivity of Ad Hoc Networks," *Comput J*, vol. 47, no. 4, pp. 432–447, Jan. 2004, doi: 10.1093/comjnl/47.4.432.
- [168] T. K. Madsen, F. H. P. Fitzek, R. Prasad, and G. Schulte, "Connectivity Probability of Wireless Ad Hoc Networks: Definition, Evaluation, Comparison," *Wireless Pers Commun*, vol. 35, no. 1–2, pp. 135–151, Oct. 2005, doi: 10.1007/s11277-005-8745-7.
- [169] R. K. Schmidt, T. Köllmer, T. Leinmüller, B. Böddeker, and G. Schäfer, "Degradation of Transmission Range in VANETs caused by Interference," *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 32, no. 4, pp. 224–234, 2010, doi: 10.1515/piko.2009.0043.
- [170] R. K. Schmidt, B. Kloiber, F. Schüttler, and T. Strang, "Degradation of Communication Range in VANETs Caused by Interference 2.0 - Real-World Experiment," in *Communication Technologies for*

- Vehicles*, T. Strang, A. Festag, A. Vinel, R. Mehmood, C. Rico Garcia, and M. Röckl, Eds., in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 176–188.
- [171] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser, “An analysis of the optimum node density for ad hoc mobile networks,” in *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, Jun. 2001, pp. 857–861 vol.3. doi: 10.1109/ICC.2001.937360.
 - [172] W. Yang, X. Yang, S. Yang, and D. Yang, “A greedy-based stable multi-path routing protocol in mobile ad hoc networks,” *Ad Hoc Networks*, vol. 9, no. 4, pp. 662–674, Jun. 2011, doi: 10.1016/j.adhoc.2010.09.004.
 - [173] F. Fitzek, L. Badia, M. Zorzi, G. Schulte, P. Seeling, and T. Henderson, “Mobility and stability evaluation in wireless multi-hop networks using multi-player games,” in *Proceedings of the 2nd workshop on Network and system support for games*, in NetGames ’03. Redwood City, California: Association for Computing Machinery, May 2003, pp. 77–87. doi: 10.1145/963900.963908.
 - [174] X. Hou, Y. Li, D. Jin, D. O. Wu, and S. Chen, “Modeling the Impact of Mobility on the Connectivity of Vehicular Networks in Large-Scale Urban Environments,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2753–2758, Apr. 2016, doi: 10.1109/TVT.2015.2418574.
 - [175] V. Lenders, J. Wagner, and M. May, “Analyzing the Impact of Mobility in Ad Hoc Networks,” in *Proceedings of the 2Nd International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, in REALMAN ’06. New York, NY, USA: ACM, 2006, pp. 39–46. doi: 10.1145/1132983.1132991.
 - [176] C. Yawut, B. Paillassa, and R. Dhaou, “On Metrics for Mobility Oriented Self Adaptive Protocols,” in *2007 Third International Conference on Wireless and Mobile Communications (ICWMC’07)*, Mar. 2007, pp. 11–11. doi: 10.1109/ICWMC.2007.60.
 - [177] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc network research,” *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002, doi: 10.1002/wcm.72.
 - [178] M. Krunz, A. Muqattash, and S.-J. Lee, “Transmission power control in wireless ad hoc networks: challenges, solutions and open issues,” *IEEE Network*, vol. 18, no. 5, pp. 8–14, Sep. 2004, doi: 10.1109/MNET.2004.1337730.
 - [179] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, “Impact of Radio Irregularity on Wireless Sensor Networks,” in *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, in MobiSys ’04. New York, NY, USA: ACM, 2004, pp. 125–138. doi: 10.1145/990064.990081.
 - [180] J. Kuruvila, A. Nayak, and I. Stojmenovic, “Hop count optimal position based packet routing algorithms for ad hoc wireless networks with a realistic physical layer,” in *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE Cat. No.04EX975)*, Oct. 2004, pp. 398–405. doi: 10.1109/MAHSS.2004.1392179.
 - [181] F. B. Zhan and C. E. Noon, “Shortest Path Algorithms: An Evaluation Using Real Road Networks,” *Transportation Science*, vol. 32, no. 1, pp. 65–73, Feb. 1998, doi: 10.1287/trsc.32.1.65.
 - [182] I. Stojmenovic, A. Nayak, and J. Kuruvila, “Design guidelines for routing protocols in ad hoc and sensor networks with a realistic physical layer,” *IEEE Communications Magazine*, vol. 43, no. 3, pp. 101–106, Mar. 2005, doi: 10.1109/MCOM.2005.1404603.
 - [183] M. Lukic, B. Pavkovic, and Mitton, “Greedy Geographic Routing Algorithms in Real Environment,” presented at the 5th International Conference on Mobile Ad-hoc and Sensor Networks, 2009. MSN ’09., Wu Yi Mountain, Fujian, China: IEEE, Dec. 2009. doi: 10.1109/MSN.2009.11.
 - [184] R. Ramanathan and R. Rosales-Hain, “Topology control of multihop wireless networks using transmit power adjustment,” in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, 2000, pp. 404–413 vol.2. doi: 10.1109/INFCOM.2000.832213.

- [185] J. Gomez and A. T. Campbell, "Variable-Range Transmission Power Control in Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 1, pp. 87–99, Jan. 2007, doi: 10.1109/TMC.2007.250673.
- [186] D. Son, B. Krishnamachari, and J. Heidemann, "Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks," in *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, Oct. 2004, pp. 289–298. doi: 10.1109/SAHCN.2004.1381929.
- [187] L. Cheng, J. Niu, J. Cao, S. K. Das, and Y. Gu, "QoS Aware Geographic Opportunistic Routing in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1864–1875, Jul. 2014, doi: 10.1109/TPDS.2013.240.
- [188] A. Alanazi and K. Elleithy, "Real-Time QoS Routing Protocols in Wireless Multimedia Sensor Networks: Study and Analysis," *Sensors (Basel)*, vol. 15, no. 9, pp. 22209–22233, Sep. 2015, doi: 10.3390/s150922209.
- [189] N. Loulloudes, G. Pallis, and M. D. Dikaiakos, "The Dynamics of Vehicular Networks in Large-Scale Urban Environments," in *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*, Oct. 2015, pp. 192–199. doi: 10.1109/CIC.2015.25.
- [190] M. Umlauf and W. Elmenreich, "Topology Characterization for Position-based Wireless Network Topology Generators," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2019, pp. 1–4. doi: 10.1109/WiMOB.2019.8923522.
- [191] W. Xu et al., "Internet of vehicles in big data era," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 19–35, Jan. 2018, doi: 10.1109/JAS.2017.7510736.
- [192] N. Zhang, P. Yang, J. Ren, D. Chen, L. Yu, and X. Shen, "Synergy of Big Data and 5G Wireless Networks: Opportunities, Approaches, and Challenges," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 12–18, Feb. 2018, doi: 10.1109/MWC.2018.1700193.
- [193] B.-L. Wenning, A. Timm-Giel, and C. Görg, "A Generic Framework for Context-Aware Routing and its Implementation in Wireless Sensor Networks," in *Mobilkommunikation: Technologien und Anwendungen: Vorträge der 14. ITG-Fachtagung vom 13. bis 14. Mai 2009 in Osnabrück*, Ralf Tönjes, P. Roer, and C. Westerkamp, Eds., in ITG-Fachbericht. Osnabrück, Germany: VDE, 2009.
- [194] Z. Zhao, T. Braun, D. Rosário, and E. Cerqueira, "CAOR: Context-aware adaptive opportunistic routing in mobile ad-hoc networks," in *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, May 2014, pp. 1–8. doi: 10.1109/WMNC.2014.6878878.
- [195] S. Knox et al., "Towards scatterbox: a context-aware message forwarding platform," 2007.
- [196] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010. Accessed: May 01, 2019. [Online]. Available: <https://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780199206650.001.0001/acprof-9780199206650>
- [197] A. Frieze and M. Karoński, *Introduction to Random Graphs*, 1 edition. Cambridge, United Kingdom: Cambridge University Press, 2016.
- [198] A.-L. Barabási and M. Pósfai, *Network Science*, 1 edition. Cambridge, United Kingdom: Cambridge University Press, 2016.
- [199] H. Conceição, M. Ferreira, and J. Barros, "On the Urban Connectivity of Vehicular Sensor Networks," in *Distributed Computing in Sensor Systems*, S. E. Nikolettseas, B. S. Chlebus, D. B. Johnson, and B. Krishnamachari, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 112–125. doi: 10.1007/978-3-540-69170-9_8.
- [200] I. Chlamtac and A. Faragó, "A new approach to the design and analysis of peer-to-peer mobile networks," *Wirel. Netw.*, vol. 5, no. 3, pp. 149–156, May 1999, doi: 10.1023/A:1019186624837.

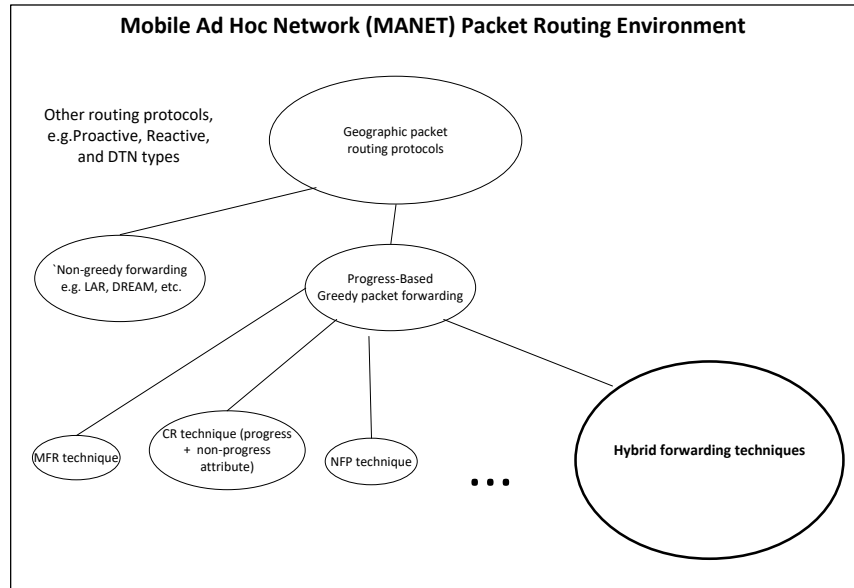
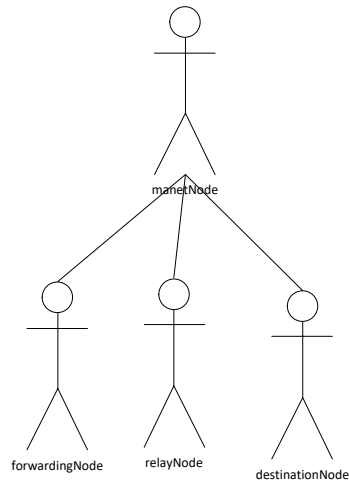
- [201] U. Schilcher, M. Gyarmati, C. Bettstetter, Y. W. Chung, and Y. H. Kim, "Measuring Inhomogeneity in Spatial Distributions," in *VTC Spring 2008 - IEEE Vehicular Technology Conference*, May 2008, pp. 2690–2694. doi: 10.1109/VETECS.2008.589.
- [202] G. Pallis, D. Katsaros, M. D. Dikaiakos, N. Loulloudes, and L. Tassiulas, "On the structure and evolution of vehicular networks," in *2009 IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, Sep. 2009, pp. 1–10. doi: 10.1109/MASCOT.2009.5366230.
- [203] V. Srivastava, A. B. Hilal, M. S. Thompson, J. N. Chattha, A. B. MacKenzie, and L. A. DaSilva, "Characterizing mobile ad hoc networks -: the maniac challenge experiment," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, in WINTech '08. San Francisco, California, USA: Association for Computing Machinery, Sep. 2008, pp. 65–72. doi: 10.1145/1410077.1410090.
- [204] A. Bertrand and M. Moonen, "Distributed computation of the Fiedler vector with application to topology inference in ad hoc networks," *Signal Processing*, vol. 93, no. 5, pp. 1106–1117, May 2013, doi: 10.1016/j.sigpro.2012.12.002.
- [205] A. Jain and B. V. Ramana Reddy, "Eigenvector centrality based cluster size control in randomly deployed wireless sensor networks," *Expert Systems with Applications*, vol. 42, no. 5, pp. 2657–2669, Apr. 2015, doi: 10.1016/j.eswa.2014.11.015.
- [206] H. Gruhn and P. Persson, "Towards a robust algorithm for distributed monitoring of network topology changes," in *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, Jun. 2014, pp. 1–7. doi: 10.1109/MedHocNet.2014.6849098.
- [207] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity for mobile sensor networks," *Automatica*, vol. 46, no. 2, pp. 390–396, Feb. 2010, doi: 10.1016/j.automatica.2009.11.012.
- [208] S. Rosati, K. Kruzelecki, L. Traynard, and B. Rimoldi, "Speed-Aware Routing for UAV Ad-Hoc Networks," presented at the Globecom 2013 - Wireless Networking and Control for Unmanned Autonomous Vehicles, Atlanta, Georgia, U.S.A.: IEEE, Dec. 2013.
- [209] A. Khelil, P. J. Marron, and K. Rothermel, "Contact-based mobility metrics for delay-tolerant ad hoc networking," presented at the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005., Georgia Tech Atlanta, GA, USA, Sep. 2005. doi: 10.1109/MASCOTS.2005.20.
- [210] T. T. Son, H. L. Minh, G. Sexton, N. Aslam, and Z. Ghassemlooy, "Bayesian Model for Mobility Prediction to Support Routing in Mobile Ad-Hoc Networks," presented at the IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), London: IEEE, Sep. 2013, pp. 3186–3190.
- [211] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, San Diego, California, U.S.A.: ACM, Sep. 2003, pp. 134–146. doi: 10.1145/938985.939000.
- [212] R. Arroyo-Valles, A. Guerrero-Curieses, and J. Cid-Sueiro, "Q-Probabilistic Routing in Wireless Sensor Networks," in *Intelligent Sensors, Sensor Networks and Information, 2007. 3rd International Conference on*, Melbourne, Australia: IEEE, Dec. 2007. doi: 10.1109/ISSNIP.2007.4496810.
- [213] W. Su, S.- Lee, and M. Gerla, "Mobility prediction in wireless networks," in *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No.00CH37155)*, Oct. 2000, pp. 491–495 vol.1. doi: 10.1109/MILCOM.2000.905001.
- [214] C. Wu, K. Kumekawa, and T. Kato, "A MANET protocol considering link stability and bandwidth efficiency," in *Proceedings of the International Conference on Ultra Modern Telecommunications*,

- ICUMT 2009, 12-14 October 2009, St. Petersburg, Russia, IEEE, 2009, pp. 1–8. doi: 10.1109/ICUMT.2009.5345608.
- [215] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas, “Characterization of complex networks: A survey of measurements,” *Advances in Physics*, vol. 56, no. 1, pp. 167–242, Jan. 2007, doi: 10.1080/00018730601170527.
- [216] E. M. Airoidi, X. Bai, and K. M. Carley, “Network sampling and classification: An investigation of network model representations,” *Decision Support Systems*, vol. 51, no. 3, pp. 506–518, Jun. 2011, doi: 10.1016/j.dss.2011.02.014.
- [217] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, “Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 243–259, Apr. 2009, doi: 10.1109/TEVC.2008.924428.
- [218] W. Torres-Pomales, “Software Fault Tolerance - A Tutorial.” NASA (.gov), Langley Research Center, Hampton, Virginia, Oct. 2000. Accessed: Jul. 21, 2023. [Online]. Available: <https://ntrs.nasa.gov/api/citations/20000120144/downloads/20000120144.pdf>
- [219] X.-Y. Li, P.-J. Wan, Y. Wang, and C.-W. Yi, “Fault Tolerant Deployment and Topology Control in Wireless Networks,” in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, in MobiHoc ’03. New York, NY, USA: ACM, 2003, pp. 117–128. doi: 10.1145/778415.778431.
- [220] S. Argamon-Engelson, S. Kraus, and S. Sina, “Interleaved versus a priori exploration for repeated navigation in a partially-known graph,” *Int. J. Patt. Recogn. Artif. Intell.*, vol. 13, no. 07, pp. 963–986, Nov. 1999, doi: 10.1142/S0218001499000537.
- [221] P. Velan, J. Medková, T. Jirsík, and P. Čeleda, “Network traffic characterisation using flow-based statistics,” in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2016, pp. 907–912. doi: 10.1109/NOMS.2016.7502924.
- [222] G. F. Riley and T. R. Henderson, “The ns-3 Network Simulator,” in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds., Springer Berlin Heidelberg, 2010, pp. 15–34.
- [223] S. Kühlmorgen, I. Llatser, A. Festag, and G. Fettweis, “Performance Evaluation of ETSI GeoNetworking for Vehicular Ad hoc Networks,” in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, Glasgow, Scotland: IEEE, May 2015. doi: 10.1109/VTCspring.2015.7146003.
- [224] K. Vladi, O. Tetsuya, S. Evjola, B. Leonard, I. Makoto, and U. Kazunori, “Performance Evaluation of a VANET Simulation System Using NS-3 and SUMO,” presented at the IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2015, Gwangju, Korea: IEEE, Mar. 2015, pp. 348–353. doi: 10.1109/WAINA.2015.121.
- [225] C. Sommer and F. Dressler, “Progressing toward Realistic Mobility Models in VANET Simulations,” *IEEE Communications Magazine*, vol. 46, no. 11, pp. 132–137, Nov. 2008.
- [226] S. Rosati, K. Kruzelecki, G. Heitz, D. Floreano, and B. Rimoldi, “Dynamic Routing for Flying Ad Hoc Networks,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 3, pp. 1690–1700, Mar. 2016, doi: 10.1109/TVT.2015.2414819.
- [227] T. X. Brown, S. Doshi, S. Jadhav, D. Henkel, and R.-G. Thekkekunel, “A full-scale wireless ad hoc network test bed,” in *International Symposium on Advance Radio Technologies*, Boulder, CO., 2005, pp. 51–60.
- [228] D. Kotz, C. Newport, and C. Elliott, “The mistaken axioms of wireless-network research,” Dartmouth College Computer Science, Technical Report TR2003-467, Jul. 2003. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.137.634>
- [229] A. Pullin, C. Pattinson, and A.-L. Kor, “Building Realistic Mobility Models for Mobile Ad Hoc Networks,” *Informatics*, vol. 5, no. 2, p. 22, Jun. 2018, doi: 10.3390/informatics5020022.

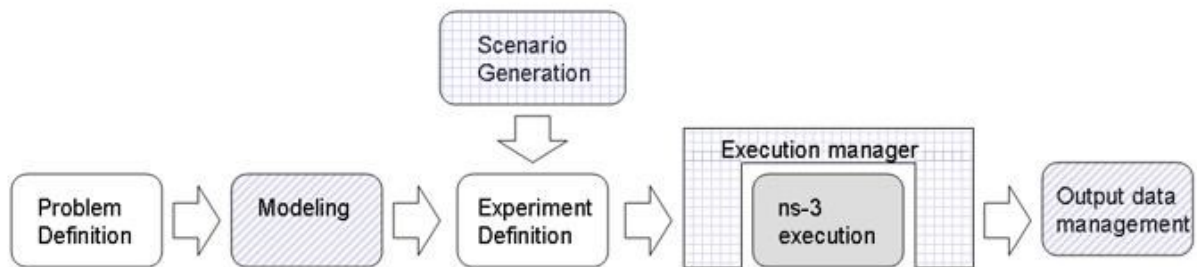
- [230] H. Yang and Z. Liu, "An optimization routing protocol for FANETs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 120, May 2019, doi: 10.1186/s13638-019-1442-0.
- [231] L. Zhang and J. Guo, "A vector-based improved geographic information routing protocol," in *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Jul. 2017, pp. 421–424. doi: 10.1109/ICEIEC.2017.8076596.
- [232] A. Bothe and N. Aschenbruck, "BonnMotion 4 – Taking Mobility Generation to the Next Level," in *2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC)*, Nov. 2020, pp. 1–8. doi: 10.1109/IPCCC50635.2020.9391563.
- [233] A. Silva, N. Reza, and A. Oliveira, "Improvement and Performance Evaluation of GPSR-Based Routing Techniques for Vehicular Ad Hoc Networks," *IEEE Access*, vol. 7, pp. 21722–21733, 2019, doi: 10.1109/ACCESS.2019.2898776.
- [234] CSVNetLab, "PA-GPSR." Jul. 19, 2021. Accessed: Aug. 14, 2021. [Online]. Available: <https://github.com/CSVNetLab/PA-GPSR>
- [235] T. Henderson, "ns-3 Introduction," University of Washington, Jul. 2014. Accessed: Feb. 11, 2019. [Online]. Available: <https://www.nsnam.org/docs/ns-3-overview-july-2014.pdf>
- [236] M. Kulin, T. Kazaz, E. De Poorter, and I. Moerman, "A Survey on Machine Learning-Based Performance Improvement of Wireless Networks: PHY, MAC and Network Layer," *Electronics*, vol. 10, no. 3, Art. no. 3, Jan. 2021, doi: 10.3390/electronics10030318.
- [237] L. da F. Costa *et al.*, "Analyzing and modeling real-world phenomena with complex networks: a survey of applications," *Advances in Physics*, vol. 60, no. 3, pp. 329–412, Jun. 2011, doi: 10.1080/00018732.2011.572452.
- [238] Y. M. G. Costa, D. Bertolini, A. S. Britto, G. D. C. Cavalcanti, and L. E. S. Oliveira, "The dissimilarity approach: a review," *Artif Intell Rev*, vol. 53, no. 4, pp. 2783–2808, Apr. 2020, doi: 10.1007/s10462-019-09746-z.
- [239] R. C. Veltkamp, "Shape matching: similarity measures and algorithms," in *Proceedings International Conference on Shape Modeling and Applications*, May 2001, pp. 188–197. doi: 10.1109/SMA.2001.923389.
- [240] D. Medina, F. Hoffmann, S. Ayaz, and C.-H. Rokitansky, "Topology characterization of high density airspace aeronautical ad hoc networks," in *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, Sep. 2008, pp. 295–304. doi: 10.1109/MAHSS.2008.4660016.
- [241] A. Zadin, T. Fevens, and T. Bdiri, "Impact of Varying Node Velocity and HELLO Interval Duration on Position-based Stable Routing in Mobile Ad Hoc Networks," *Procedia Computer Science*, vol. 94, pp. 353–358, Jan. 2016, doi: 10.1016/j.procs.2016.08.053.
- [242] S. Liu, T. Fevens, and A. E. Abdallah, "Hybrid Position-Based Routing Algorithms for 3D Mobile Ad Hoc Networks," in *2008 The 4th International Conference on Mobile Ad-hoc and Sensor Networks*, Dec. 2008, pp. 177–186. doi: 10.1109/MSN.2008.43.
- [243] Ronzani, "Analysis and Comparison of Position-based Routing Protocols for 3D MANETs," Università degli Studi di Padova, Padua, Italy, 2015. [Online]. Available: http://www.math.unipd.it/~dronzani/docs/md_thesis_ronzani.pdf
- [244] Q. Zhou, W. Gu, J. Li, Q. Sun, and F. Yang, "A Topology Aware Routing Protocol Based ADS-B System for Aeronautical Ad Hoc Networks," in *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, Sep. 2012, pp. 1–4. doi: 10.1109/WiCOM.2012.6478379.
- [245] F. Lyu *et al.*, "Characterizing Urban Vehicle-to-Vehicle Communications for Reliable Safety Applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2586–2602, Jun. 2020, doi: 10.1109/TITS.2019.2920813.

APPENDICES

Appendix A: Use case: Geographic packet routing and forwarding in MANET



Appendix B: Multi-greedy packet forwarding evaluation design using NS-3 [235]



Appendix C: Parts of the C++ Codes Modified in the NS-3 Simulator

The codes show highlighted portions with comments bearing ‘atanda’, which are the lines of specific modifications made in the different modules, including ns3-gpsr-ccp, ns3-gpsr-ptable-cpp, ns3-gpsr-ptable-cpph, ns3-ipv4-flow-probe-cpp, ns3-ipv4-flow-probe-cpp, and ns3-pagpsr-main-cpp:

ns3-gpsr-ccp

```

1  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2  /*
3  *  GPSR
4  *
5  */
6  #define NS_LOG_APPEND_CONTEXT \
7  if (m_ipv4) { std::clog << "[node " << m_ipv4->GetObject<Node> ()->GetId ()
<< "]" "; }
8
9  #include "gpsr.h"
10 #include "ns3/log.h"
11 #include "ns3/boolean.h"
12 #include "ns3/random-variable-stream.h"
13 #include "ns3/inet-socket-address.h"
14 #include "ns3/trace-source-accessor.h"
15 #include "ns3/udp-socket-factory.h"
16 #include "ns3/wifi-net-device.h"
17 #include "ns3/adhoc-wifi-mac.h"
18 #include "ns3/rng-seed-manager.h"
19 #include <algorithm>
20 #include <limits>
21
22
23 #define GPSR_LS_GOD 0
24
25 #define GPSR_LS_RLS 1
26
27
28 float InPackets = 0.0;
29 float OutPackets = 0.0;
30 uint32_t count = 0;
31
32 NS_LOG_COMPONENT_DEFINE ("GpsrRoutingProtocol");
33
34 namespace ns3 {
35 namespace gpsr {
36
37
38
39 struct DeferredRouteOutputTag : public Tag
40 {
41     /// Positive if output device is fixed in RouteOutput
42     uint32_t m_isCallFromL3;
43
44     DeferredRouteOutputTag () : Tag (),
45     m_isCallFromL3 (0)
46     {
47     }
48
49     static TypeId GetTypeId ()
50     {
51     static TypeId tid = TypeId
52     ("ns3::gpsr::DeferredRouteOutputTag").SetParent<Tag> ();
53     return tid;
54     }
55
56     static TypeId GetInstanceTypeId () const
57     {

```

```

57 return GetTypeId ();
58 }
59
60 uint32_t GetSerializedSize () const
61 {
62 return sizeof(uint32_t);
63 }
64
65 void Serialize (TagBuffer i) const
66 {
67 i.WriteU32 (m_isCallFromL3);
68 }
69
70 void Deserialize (TagBuffer i)
71 {
72 m_isCallFromL3 = i.ReadU32 ();
73 }
74
75 void Print (std::ostream &os) const
76 {
77 os << "DeferredRouteOutputTag: m_isCallFromL3 = " << m_isCallFromL3;
78 }
79 };
80
81 //Ptr<UniformRandomVariable> x = CreateObject<UniformRandomVariable> ();
82
83 /***** Miscellaneous constants *****/
84
85 /// Maximum allowed jitter.
86 #define GPSR_MAXJITTER (HelloInterval.GetSeconds () / 2)
87 /// Random number between [(-GPSR_MAXJITTER)-GPSR_MAXJITTER] used to
jitter HELLO packet
transmission.
88
89 // #define JITTER (Seconds (x->GetValue (-GPSR_MAXJITTER, GPSR_MAXJITTER)))
90 // #define FIRST_JITTER (Seconds (x->GetValue (0, GPSR_MAXJITTER))) //first
Hello can not
be in the past, used only on SetIpv4
91
92
93
94 NS_OBJECT_ENSURE_REGISTERED (RoutingProtocol);
95
96 /// UDP Port for GPSR control traffic, not defined by IANA yet
97 const uint32_t RoutingProtocol::GPSR_PORT = 666;
98
99 RoutingProtocol::RoutingProtocol ()
100 : HelloInterval (Seconds (1)),
101 MaxQueueLen (64),
102 MaxQueueTime (Seconds (30)),
103 m_queue (MaxQueueLen, MaxQueueTime),
104 HelloIntervalTimer (Timer::CANCEL_ON_DESTROY),
105 m_delay(Seconds (0))
106 {
107
108 m_neighbors = PositionTable ();
109 }
110

```

```

111 TypeId
112 RoutingProtocol::GetTypeId (void)
113 {
114     static TypeId tid = TypeId ("ns3::gpsr::RoutingProtocol")
115     .SetParent<Ipv4RoutingProtocol> ()
116     .AddConstructor<RoutingProtocol> ()
117     .AddAttribute ("HelloInterval", "HELLO messages emission interval.",
118     TimeValue (Seconds (1)),
119     MakeTimeAccessor (&RoutingProtocol::HelloInterval),
120     MakeTimeChecker ())
121     .AddAttribute ("LocationServiceName", "Indicates wich Location Service is
enabled" ,
122     EnumValue (GPSR_LS_GOD),
123     MakeEnumAccessor (&RoutingProtocol::LocationServiceName),
124     MakeEnumChecker (GPSR_LS_GOD, "GOD",
125     GPSR_LS_RLS, "RLS"))
126     .AddTraceSource ("HelloTx",
127     "Number of hello packets transmitted.",
128     MakeTraceSourceAccessor (&RoutingProtocol::m_hello),
129     "ns3::gpsr::RoutingProtocol::Hello")
130 ;
131     return tid;
132 }
133
134 RoutingProtocol::~~RoutingProtocol ()
135 {
136 }
137
138 void
139 RoutingProtocol::DoDispose ()
140 {
141     m_ipv4 = 0;
142     Ipv4RoutingProtocol::DoDispose ();
143 }
144
145 Ptr<LocationService>
146 RoutingProtocol::GetLS ()
147 {
148     return m_locationService;
149 }
150 void
151 RoutingProtocol::SetLS (Ptr<LocationService> locationService)
152 {
153     m_locationService = locationService;
154 }
155
156 float
157 RoutingProtocol::GetPdr ()
158 {
159     float pdr = InPackets/OutPackets;
160     InPackets = 0;
161     OutPackets = 0;
162     return pdr;
163 }
164
165
166 bool RoutingProtocol::RouteInput (Ptr<const Packet> p, const Ipv4Header
&header, Ptr<

```

```

const NetDevice> idev,
167 UnicastForwardCallback ucb, MulticastForwardCallback
mcb,
168 LocalDeliverCallback lcb, ErrorCallback ecb)
169 {
170
171 NS_LOG_FUNCTION (this << p->GetUid () << header.GetDestination () <<
idev->GetAddress
());
172 if (m_socketAddresses.empty ())
173 {
174 NS_LOG_LOGIC ("No gpsr interfaces");
175 return false;
176 }
177
178 NS_ASSERT (m_ipv4 != 0);
179 NS_ASSERT (p != 0);
180 // Check if input device supports IP
181 NS_ASSERT (m_ipv4->GetInterfaceForDevice (idev) >= 0);
182 int32_t iif = m_ipv4->GetInterfaceForDevice (idev);
183 Ipv4Address dst = header.GetDestination ();
184 Ipv4Address origin = header.GetSource ();
185
186 DeferredRouteOutputTag tag; //FIXME since I have to check if it's in
origin for it to
work it means I'm not taking some tag out...
187 if (p->PeekPacketTag (tag) && IsMyOwnAddress (origin))
188 {
189 Ptr<Packet> packet = p->Copy (); //FIXME ja estou a abusar de tirar tags
190 packet->RemovePacketTag (tag);
191 DeferredRouteOutput (packet, header, ucb, ecb);
192 return true;
193 }
194
195
196
197 if (m_ipv4->IsDestinationAddress (dst, iif))
198 {
199
200 Ptr<Packet> packet = p->Copy ();
201 TypeHeader tHeader (GPSRTYPE_POS);
202 packet->RemoveHeader (tHeader);
203 if (!tHeader.IsValid ())
204 {
205 NS_LOG_DEBUG ("GPSR message " << packet->GetUid () << " with unknown type
received: " << tHeader.Get () << ". Ignored");
206 return false;
207 }
208
209 if (tHeader.Get () == GPSRTYPE_POS)
210 {
211 if (dst == Ipv4Address ("10.0.0.52"))
212 ++InPackets;
213 PositionHeader phdr;
214 packet->RemoveHeader (phdr);
215 }
216
217 if (dst != m_ipv4->GetAddress (1, 0).GetBroadcast ())

```

```

218 {
219 NS_LOG_LOGIC ("Unicast local delivery to " << dst);
220 }
221 else
222 {
223 NS_LOG_LOGIC ("Broadcast local delivery to " << dst);
224 }
225
226 lcb (packet, header, iif);
227 return true;
228 }
229
230
231 return Forwarding (p, header, ucb, ecb);
232 }
233
234
235 void
236 RoutingProtocol::DeferredRouteOutput (Ptr<const Packet> p, const
Ipv4Header & header,
237 UnicastForwardCallback ucb, ErrorCallback ecb)
238 {
239 NS_LOG_FUNCTION (this << p << header);
240 NS_ASSERT (p != 0 && p != Ptr<Packet> ());
241
242 if (m_queue.GetSize () == 0)
243 {
244 CheckQueueTimer.Cancel ();
245 CheckQueueTimer.Schedule (Time ("500ms"));
246 }
247
248 QueueEntry newEntry (p, header, ucb, ecb);
249 bool result = m_queue.Enqueue (newEntry);
250
251
252 m_queuedAddresses.insert (m_queuedAddresses.begin (),
header.GetDestination ());
253 m_queuedAddresses.unique ();
254
255 if (result)
256 {
257 NS_LOG_LOGIC ("Add packet " << p->GetUid () << " to queue. Protocol " <<
(uint16_t)
header.GetProtocol ());
258
259 }
260
261 }
262
263 void
264 RoutingProtocol::CheckQueue ()
265 {
266
267 CheckQueueTimer.Cancel ();
268
269 std::list<Ipv4Address> toRemove;
270

```

```

271 for (std::list<Ipv4Address>::iterator i = m_queuedAddresses.begin (); i
    !=
m_queuedAddresses.end (); ++i)
272 {
273     if (SendPacketFromQueue (*i))
274     {
275         //Insert in a list to remove later
276         toRemove.insert (toRemove.begin (), *i);
277     }
278 }
279
280 //remove all that are on the list
281 for (std::list<Ipv4Address>::iterator i = toRemove.begin (); i !=
toRemove.end (); ++i)
282 {
283     m_queuedAddresses.remove (*i);
284 }
285
286 if (!m_queuedAddresses.empty ()) //Only need to schedule if the queue is
not empty
287 {
288     CheckQueueTimer.Schedule (Time ("500ms"));
289 }
290 }
291
292 bool
293 RoutingProtocol::SendPacketFromQueue (Ipv4Address dst)
294 {
295     NS_LOG_FUNCTION (this);
296     bool recovery = false;
297     QueueEntry queueEntry;
298
299
300     if (m_locationService->IsInSearch (dst))
301     {
302         return false;
303     }
304
305     if (!m_locationService->HasPosition (dst)) // Location-service stoped
looking for the
dst
306     {
307         m_queue.DropPacketWithDst (dst);
308         NS_LOG_LOGIC ("Location Service did not find dst. Drop packet to " <<
dst);
309         return true;
310     }
311
312     Vector myPos;
313
314     Ptr<MobilityModel> MM = m_ipv4->GetObject<MobilityModel> ();
315     myPos.x = MM->GetPosition ().x;
316     myPos.y = MM->GetPosition ().y;
317     Ipv4Address nextHop;
318
319     if(m_neighbors.isNeighbour (dst))
320     {
321         nextHop = dst;

```

```

322 }
323 else{
324 Vector dstPos = m_locationService->GetPosition (dst);
325 nextHop = m_neighbors.BestNeighbor (dstPos, myPos);
326 if (nextHop == Ipv4Address::GetZero ())
327 {
328 NS_LOG_LOGIC ("Fallback to recovery-mode. Packets to " << dst);
329 recovery = true;
330 }
331 if(recovery)
332 {
333
334 Vector Position;
335 Vector previousHop;
336 uint32_t updated;
337
338 while(m_queue.Dequeue (dst, queueEntry))
339 {
340 Ptr<Packet> p = ConstCast<Packet> (queueEntry.GetPacket ());
341 UnicastForwardCallback ucb = queueEntry.GetUnicastForwardCallback ();
342 Ipv4Header header = queueEntry.GetIpv4Header ();
343
344 TypeHeader tHeader (GPSRTYPE_POS);
345 p->RemoveHeader (tHeader);
346 if (!tHeader.IsValid ())
347 {
348 NS_LOG_DEBUG ("GPSR message " << p->GetUid () << " with unknown type
received: " << tHeader.Get () << ". Drop");
349 return false; // drop
350 }
351 if (tHeader.Get () == GPSRTYPE_POS)
352 {
353 PositionHeader hdr;
354 p->RemoveHeader (hdr);
355 Position.x = hdr.GetDstPosx ();
356 Position.y = hdr.GetDstPosy ();
357 updated = hdr.GetUpdated ();
358 }
359
360 PositionHeader posHeader (Position.x, Position.y, updated, myPos.x,
myPos.y,
(uint8_t) 1, Position.x, Position.y);
361 p->AddHeader (posHeader); //enters in recovery with last edge from Dst
362 p->AddHeader (tHeader);
363
364 RecoveryMode(dst, p, ucb, header);
365 }
366 return true;
367 }
368 }
369 Ptr<Ipv4Route> route = Create<Ipv4Route> ();
370 route->SetDestination (dst);
371 route->SetGateway (nextHop);
372
373 // FIXME: Does not work for multiple interfaces
374 route->SetOutputDevice (m_ipv4->GetNetDevice (1));
375
376 while (m_queue.Dequeue (dst, queueEntry))

```

```

377 {
378 DeferredRouteOutputTag tag;
379 Ptr<Packet> p = ConstCast<Packet> (queueEntry.GetPacket ());
380
381 UnicastForwardCallback ucb = queueEntry.GetUnicastForwardCallback ();
382 Ipv4Header header = queueEntry.GetIpv4Header ();
383
384 if (header.GetSource () == Ipv4Address ("102.102.102.102"))
385 {
386 route->SetSource (m_ipv4->GetAddress (1, 0).GetLocal ());
387 header.SetSource (m_ipv4->GetAddress (1, 0).GetLocal ());
388 }
389 else
390 {
391 route->SetSource (header.GetSource ());
392 }
393 ucb (route, p, header);
394 }
395 return true;
396 }
397
398
399 void
400 RoutingProtocol::RecoveryMode(Ipv4Address dst, Ptr<Packet> p,
UnicastForwardCallback ucb,
Ipv4Header header){
401 //std::cout<<"Entered in recovery Mode\n";
402
403
404 //ATANDA
405 //Disable/Enable recovery function. Disable - prevents entry into
recovery mode, i.e.,
aborts continuation to recovery mode
406 bool doNotRecover = true; //to drop packet
407 if (doNotRecover)
408 {
409 //std::cout<<"Recover not, drop\n";
410 return;
411 }
412
413 //std::cout<<"Contihuing in recovery Mode\n";
414 //END - ATANDA
415
416
417
418 Vector Position;
419 Vector previousHop;
420 uint32_t updated;
421 uint64_t positionX;
422 uint64_t positionY;
423 Vector myPos;
424 Vector recPos;
425
426 Ptr<MobilityModel> MM = m_ipv4->GetObject<MobilityModel> ();
427 positionX = MM->GetPosition ().x;
428 positionY = MM->GetPosition ().y;
429 myPos.x = positionX;
430 myPos.y = positionY;

```

```

431
432 TypeHeader tHeader (GPSRTYPE_POS);
433 p->RemoveHeader (tHeader);
434 if (!tHeader.IsValid ())
435 {
436 NS_LOG_DEBUG ("GPSR message " << p->GetUid () << " with unknown type
received: " <<
tHeader.Get () << ". Drop");
437 return; // drop
438 }
439 if (tHeader.Get () == GPSRTYPE_POS)
440 {
441 PositionHeader hdr;
442 p->RemoveHeader (hdr);
443 Position.x = hdr.GetDstPosx ();
444 Position.y = hdr.GetDstPosy ();
445 updated = hdr.GetUpdated ();
446 recPos.x = hdr.GetRecPosx ();
447 recPos.y = hdr.GetRecPosy ();
448 previousHop.x = hdr.GetLastPosx ();
449 previousHop.y = hdr.GetLastPosy ();
450 }
451
452 PositionHeader posHeader (Position.x, Position.y, updated, recPos.x,
recPos.y, (
uint8_t) 1, myPos.x, myPos.y);
453 p->AddHeader (posHeader);
454 p->AddHeader (tHeader);
455
456
457 Ipv4Address nextHop = m_neighbors.BestAngle (previousHop, myPos);
458
459 if (nextHop == Ipv4Address::GetZero ())
460 {
461 return;
462 }
463
464 Ptr<Ipv4Route> route = Create<Ipv4Route> ();
465 route->SetDestination (dst);
466 route->SetGateway (nextHop);
467
468 // FIXME: Does not work for multiple interfaces
469 route->SetOutputDevice (m_ipv4->GetNetDevice (1));
470 route->SetSource (header.GetSource ());
471
472 ucb (route, p, header);
473 return;
474 }
475
476
477 void
478 RoutingProtocol::NotifyInterfaceUp (uint32_t interface)
479 {
480 NS_LOG_FUNCTION (this << m_ipv4->GetAddress (interface, 0).GetLocal ());
481 Ptr<Ipv4L3Protocol> l3 = m_ipv4->GetObject<Ipv4L3Protocol> ();
482 if (l3->GetNAddresses (interface) > 1)
483 {

```

```

484 NS_LOG_WARN ("GPSR does not work with more then one address per each
interface." );
485 }
486 Ipv4InterfaceAddress iface = l3->GetAddress (interface, 0);
487 if (iface.GetLocal () == Ipv4Address ("127.0.0.1"))
488 {
489     return;
490 }
491
492 // Create a socket to listen only on this interface
493 Ptr<Socket> socket = Socket::CreateSocket (GetObject<Node> (),
494 UdpSocketFactory::GetTypeId ());
495 NS_ASSERT (socket != 0);
496 socket->SetRecvCallback (MakeCallback (&RoutingProtocol::RecvGPSR,
this));
497 socket->Bind (InetSocketAddress (Ipv4Address::GetAny (), GPSR_PORT));
498 socket->BindToNetDevice (l3->GetNetDevice (interface));
499 socket->SetAllowBroadcast (true);
500 socket->SetAttribute ("IpTtl", UintegerValue (1));
501 m_socketAddresses.insert (std::make_pair (socket, iface));
502
503
504 // Allow neighbor manager use this interface for layer 2 feedback if
possible
505 Ptr<NetDevice> dev = m_ipv4->GetNetDevice (m_ipv4->GetInterfaceForAddress
(iface.
GetLocal ());
506 Ptr<WifiNetDevice> wifi = dev->GetObject<WifiNetDevice> ();
507 if (wifi == 0)
508 {
509     return;
510 }
511 Ptr<WifiMac> mac = wifi->GetMac ();
512 if (mac == 0)
513 {
514     return;
515 }
516
517 mac->TraceConnectWithoutContext ("TxErrHeader",
m_neighbors.GetTxErrorCallback ());
518
519 }
520
521
522 void
523 RoutingProtocol::RecvGPSR (Ptr<Socket> socket)
524 {
525     NS_LOG_FUNCTION (this << socket);
526     Address sourceAddress;
527     Ptr<Packet> packet = socket->RecvFrom (sourceAddress);
528
529     TypeHeader tHeader (GPSRTYPE_HELLO);
530     packet->RemoveHeader (tHeader);
531     if (!tHeader.IsValid ())
532     {
533         NS_LOG_DEBUG ("GPSR message " << packet->GetUid () << " with unknown type
received: " << tHeader.Get () << ". Ignored");
534         return;

```

```

535 }
536
537 HelloHeader hdr;
538 packet->RemoveHeader (hdr);
539 Vector Position;
540 Position.x = hdr.GetOriginPosX ();
541 Position.y = hdr.GetOriginPosY ();
542 InetAddress inetSourceAddr = InetAddress::ConvertFrom
(sourceAddress);
543 Ipv4Address sender = inetSourceAddr.GetIpv4 ();
544 Ipv4Address receiver = m_socketAddresses[socket].GetLocal ();
545
546 UpdateRouteToNeighbor (sender, receiver, Position);
547
548 }
549
550
551 void
552 RoutingProtocol::UpdateRouteToNeighbor (Ipv4Address sender, Ipv4Address
receiver, Vector
Pos)
553 {
554 m_neighbors.AddEntry (sender, Pos);
555
556 }
557
558
559
560 void
561 RoutingProtocol::NotifyInterfaceDown (uint32_t interface)
562 {
563 NS_LOG_FUNCTION (this << m_ipv4->GetAddress (interface, 0).GetLocal ());
564
565 // Disable layer 2 link state monitoring (if possible)
566 Ptr<Ipv4L3Protocol> l3 = m_ipv4->GetObject<Ipv4L3Protocol> ();
567 Ptr<NetDevice> dev = l3->GetNetDevice (interface);
568 Ptr<WifiNetDevice> wifi = dev->GetObject<WifiNetDevice> ();
569 if (wifi != 0)
570 {
571 Ptr<WifiMac> mac = wifi->GetMac ()->GetObject<AdhocWifiMac> ();
572 if (mac != 0)
573 {
574 mac->TraceDisconnectWithoutContext ("TxErrHeader",
575 m_neighbors.GetTxErrorCallback ());
576 }
577 }
578
579 // Close socket
580 Ptr<Socket> socket = FindSocketWithInterfaceAddress (m_ipv4->GetAddress
(interface, 0
));
581 NS_ASSERT (socket);
582 socket->Close ();
583 m_socketAddresses.erase (socket);
584 if (m_socketAddresses.empty ())
585 {
586 NS_LOG_LOGIC ("No gpsr interfaces");
587 m_neighbors.Clear ();

```

```

588 m_locationService->Clear ();
589 return;
590 }
591 }
592
593
594 Ptr<Socket>
595 RoutingProtocol::FindSocketWithInterfaceAddress (Ipv4InterfaceAddress
addr ) const
596 {
597 NS_LOG_FUNCTION (this << addr);
598 for (std::map<Ptr<Socket>, Ipv4InterfaceAddress>::const_iterator j =
599 m_socketAddresses.begin (); j != m_socketAddresses.end (); ++j)
600 {
601 Ptr<Socket> socket = j->first;
602 Ipv4InterfaceAddress iface = j->second;
603 if (iface == addr)
604 {
605 return socket;
606 }
607 }
608 Ptr<Socket> socket;
609 return socket;
610 }
611
612
613
614 void RoutingProtocol::NotifyAddAddress (uint32_t interface,
Ipv4InterfaceAddress address)
615 {
616 NS_LOG_FUNCTION (this << " interface " << interface << " address " <<
address);
617 Ptr<Ipv4L3Protocol> l3 = m_ipv4->GetObject<Ipv4L3Protocol> ();
618 if (!l3->IsUp (interface))
619 {
620 return;
621 }
622 if (l3->GetNAddresses ((interface) == 1))
623 {
624 Ipv4InterfaceAddress iface = l3->GetAddress (interface, 0);
625 Ptr<Socket> socket = FindSocketWithInterfaceAddress (iface);
626 if (!socket)
627 {
628 if (iface.GetLocal () == Ipv4Address ("127.0.0.1"))
629 {
630 return;
631 }
632 // Create a socket to listen only on this interface
633 Ptr<Socket> socket = Socket::CreateSocket (GetObject<Node> (),
634 UdpSocketFactory::GetTypeId ());
635 NS_ASSERT (socket != 0);
636 socket->SetRecvCallback (MakeCallback (&RoutingProtocol::RecvGPSR,this));
637 // Bind to any IP address so that broadcasts can be received
638 socket->Bind (InetSocketAddress (Ipv4Address::GetAny (), GPSR_PORT));
639 socket->BindToNetDevice (l3->GetNetDevice (interface));
640 socket->SetAllowBroadcast (true);
641 m_socketAddresses.insert (std::make_pair (socket, iface));
642

```

```

643 Ptr<NetDevice> dev = m_ipv4->GetNetDevice (m_ipv4->GetInterfaceForAddress
(
iface.GetLocal ());
644 }
645 }
646 else
647 {
648 NS_LOG_LOGIC ("GPSR does not work with more then one address per each
interface.
Ignore added address");
649 }
650 }
651
652 void
653 RoutingProtocol::NotifyRemoveAddress (uint32_t i, Ipv4InterfaceAddress
address)
654 {
655 NS_LOG_FUNCTION (this);
656 Ptr<Socket> socket = FindSocketWithInterfaceAddress (address);
657 if (socket)
658 {
659
660 m_socketAddresses.erase (socket);
661 Ptr<Ipv4L3Protocol> l3 = m_ipv4->GetObject<Ipv4L3Protocol> ();
662 if (l3->GetNAddresses (i))
663 {
664 Ipv4InterfaceAddress iface = l3->GetAddress (i, 0);
665 // Create a socket to listen only on this interface
666 Ptr<Socket> socket = Socket::CreateSocket (GetObject<Node> (),
667 UdpSocketFactory::GetTypeId ());
668 NS_ASSERT (socket != 0);
669 socket->SetRecvCallback (MakeCallback (&RoutingProtocol::RecvGPSR,
this));
670 // Bind to any IP address so that broadcasts can be received
671 socket->Bind (InetSocketAddress (Ipv4Address::GetAny (), GPSR_PORT));
672 socket->SetAllowBroadcast (true);
673 m_socketAddresses.insert (std::make_pair (socket, iface));
674
675 // Add local broadcast record to the routing table
676 Ptr<NetDevice> dev = m_ipv4->GetNetDevice (m_ipv4->GetInterfaceForAddress
(
iface.GetLocal ());
677
678 }
679 if (m_socketAddresses.empty ())
680 {
681 NS_LOG_LOGIC ("No gpsr interfaces");
682 m_neighbors.Clear ();
683 m_locationService->Clear ();
684 return;
685 }
686 }
687 else
688 {
689 NS_LOG_LOGIC ("Remove address not participating in GPSR operation" );
690 }
691 }
692

```

```

693 void
694 RoutingProtocol::SetIpv4 (Ptr<Ipv4> ipv4)
695 {
696     NS_ASSERT (ipv4 != 0);
697     NS_ASSERT (m_ipv4 == 0);
698
699     m_ipv4 = ipv4;
700
701     float n = float (rand()) / (float (RAND_MAX/GPSR_MAXJITTER) );
702     JITTER = Seconds(n);
703
704     HelloIntervalTimer.SetFunction (&RoutingProtocol::HelloTimerExpire,
705     this);
706     HelloIntervalTimer.Schedule (JITTER);
707
708     //Schedule only when it has packets on queue
709     CheckQueueTimer.SetFunction (&RoutingProtocol::CheckQueue, this);
710
711     Simulator::ScheduleNow (&RoutingProtocol::Start, this);
712 }
713
714 void
715 RoutingProtocol::HelloTimerExpire ()
716 {
717     float n = float (rand()) / (float (RAND_MAX/GPSR_MAXJITTER) );
718     float n2 = -(float (rand()) / (float (RAND_MAX/GPSR_MAXJITTER) ));
719
720     JITTER = Seconds(n+n2);
721
722     SendHello ();
723     HelloIntervalTimer.Cancel ();
724     HelloIntervalTimer.Schedule (HelloInterval + JITTER + m_delay);
725 }
726
727 void
728 RoutingProtocol::SendHello ()
729 {
730     NS_LOG_FUNCTION (this);
731     double positionX;
732     double positionY;
733     //double probToSend;
734
735     //Ptr<UniformRandomVariable> val = CreateObject<UniformRandomVariable>
736     ();
737     //double value = val->GetValue (0, 1);
738
739     //uint64_t density = m_neighbors.GetSize();
740
741     //if (density == 0)
742     //    probToSend = 1;
743     //else
744     //    probToSend = 1/double(density);
745
746     Ptr<MobilityModel> MM = m_ipv4->GetObject<MobilityModel> ();
747
748     positionX = MM->GetPosition ().x;
749     positionY = MM->GetPosition ().y;

```

```

749
750 for (std::map<Ptr<Socket>, Ipv4InterfaceAddress>::const_iterator j =
m_socketAddresses.
begin (); j != m_socketAddresses.end (); ++j)
751 {
752 Ptr<Socket> socket = j->first;
753 Ipv4InterfaceAddress iface = j->second;
754 HelloHeader helloHeader (((uint64_t) positionX),((uint64_t) positionY));
755
756 Ptr<Packet> packet = Create<Packet> ();
757 packet->AddHeader (helloHeader);
758 TypeHeader tHeader (GPSRTYPE_HELLO);
759 packet->AddHeader (tHeader);
760 // Send to all-hosts broadcast if on /32 addr, subnet-directed otherwise
761 Ipv4Address destination;
762 if (iface.GetMask () == Ipv4Mask::GetOnes ())
763 {
764 destination = Ipv4Address ("255.255.255.255");
765 }
766 else
767 {
768 destination = iface.GetBroadcast ();
769 }
770
771 m_hello (++count);
772 //m_delay = Seconds(1-probToSend);
773 socket->SendTo (packet, 0, InetSocketAddress (destination, GPSR_PORT));
774
775
776 }
777 }
778
779 bool
780 RoutingProtocol::IsMyOwnAddress (Ipv4Address src)
781 {
782 NS_LOG_FUNCTION (this << src);
783 for (std::map<Ptr<Socket>, Ipv4InterfaceAddress>::const_iterator j =
784 m_socketAddresses.begin (); j != m_socketAddresses.end (); ++j)
785 {
786 Ipv4InterfaceAddress iface = j->second;
787 if (src == iface.GetLocal ())
788 {
789 return true;
790 }
791 }
792 return false;
793 }
794
795
796
797
798 void
799 RoutingProtocol::Start ()
800 {
801 NS_LOG_FUNCTION (this);
802 m_queuedAddresses.clear ();
803
804 //FIXME ajustar timer, meter valor parametrizavel

```

```

805 Time tableTime ("2s");
806 switch (LocationServiceName)
807 {
808 case GPSR_LS_GOD:
809 NS_LOG_DEBUG ("GodLS in use");
810 m_locationService = CreateObject<GodLocationService> ();
811 break;
812 case GPSR_LS_RLS:
813 NS_LOG_UNCOND ("RLS not yet implemented");
814 break;
815 }
816
817 }
818
819 Ptr<Ipv4Route>
820 RoutingProtocol::LoopbackRoute (const Ipv4Header & hdr, Ptr<NetDevice>
oif)
821 {
822 NS_LOG_FUNCTION (this << hdr);
823 m_lo = m_ipv4->GetNetDevice (0);
824 NS_ASSERT (m_lo != 0);
825 Ptr<Ipv4Route> rt = Create<Ipv4Route> ();
826 rt->SetDestination (hdr.GetDestination ());
827
828 std::map<Ptr<Socket>, Ipv4InterfaceAddress>::const_iterator j =
m_socketAddresses.begin
();
829 if (oif)
830 {
831 // Iterate to find an address on the oif device
832 for (j = m_socketAddresses.begin (); j != m_socketAddresses.end (); ++j)
833 {
834 Ipv4Address addr = j->second.GetLocal ();
835 int32_t interface = m_ipv4->GetInterfaceForAddress (addr);
836 if (oif == m_ipv4->GetNetDevice (static_cast<uint32_t> (interface)))
837 {
838 rt->SetSource (addr);
839 break;
840 }
841 }
842 }
843 else
844 {
845 rt->SetSource (j->second.GetLocal ());
846 }
847 NS_ASSERT_MSG (rt->GetSource () != Ipv4Address (), "Valid GPSR source
address not
found");
848 rt->SetGateway (Ipv4Address ("127.0.0.1"));
849 rt->SetOutputDevice (m_lo);
850 return rt;
851 }
852
853
854 int
855 RoutingProtocol::GetProtocolNumber (void) const
856 {
857 return GPSR_PORT;

```

```

858 }
859
860 void
861 RoutingProtocol::AddHeaders (Ptr<Packet> p, Ipv4Address source,
862 Ipv4Address destination,
863 uint8_t protocol, Ptr<Ipv4Route> route)
864 {
865
866 NS_LOG_FUNCTION (this << " source " << source << " destination " <<
867 destination);
868
869 Vector myPos;
870 Ptr<MobilityModel> MM = m_ipv4->GetObject<MobilityModel> ();
871 myPos.x = MM->GetPosition ().x;
872 myPos.y = MM->GetPosition ().y;
873 /*
874 Ipv4Address nextHop;
875
876 if(m_neighbors.isNeighbour (destination))
877 {
878 nextHop = destination;
879 }
880 else
881 {
882 nextHop = m_neighbors.BestNeighbor (m_locationService->GetPosition
883 (destination),
884 myPos);
885 }
886 */
887 uint16_t positionX = 0;
888 uint16_t positionY = 0;
889 uint32_t hdrTime = 0;
890
891 if(destination != m_ipv4->GetAddress (1, 0).GetBroadcast ())
892 {
893 positionX = m_locationService->GetPosition (destination).x;
894 positionY = m_locationService->GetPosition (destination).y;
895 hdrTime = (uint32_t) m_locationService->GetEntryUpdateTime
896 (destination).GetSeconds
897 ();
898 }
899
900 PositionHeader posHeader (positionX, positionY, hdrTime, (uint64_t)
901 0, (uint64_t) 0, (
902 uint8_t) 0, myPos.x, myPos.y);
903 p->AddHeader (posHeader);
904 TypeHeader tHeader (GPSRTYPE_POS);
905 p->AddHeader (tHeader);
906
907 m_downTarget (p, source, destination, protocol, route);
908
909 }
910
911 bool
912 RoutingProtocol::Forwarding (Ptr<const Packet> packet, const Ipv4Header &
913 header,
914 UnicastForwardCallback ucb, ErrorCallback ecb)
915 {

```

```

906 Ptr<Packet> p = packet->Copy ();
907 NS_LOG_FUNCTION (this);
908 Ipv4Address dst = header.GetDestination ();
909 Ipv4Address origin = header.GetSource ();
910
911 m_neighbors.Purge ();
912
913 uint32_t updated = 0;
914 Vector Position;
915 Vector RecPosition;
916 uint8_t inRec = 0;
917
918 Vector previousHop;
919
920 TypeHeader tHeader (GPSRTYPE_POS);
921 PositionHeader hdr;
922 p->RemoveHeader (tHeader);
923 if (!tHeader.IsValid ())
924 {
925 NS_LOG_DEBUG ("GPSR message " << p->GetUid () << " with unknown type
received: " <<
tHeader.Get () << ". Drop");
926 return false; // drop
927 }
928 if (tHeader.Get () == GPSRTYPE_POS)
929 {
930
931 p->RemoveHeader (hdr);
932 Position.x = hdr.GetDstPosx ();
933 Position.y = hdr.GetDstPosy ();
934 updated = hdr.GetUpdated ();
935 RecPosition.x = hdr.GetRecPosx ();
936 RecPosition.y = hdr.GetRecPosy ();
937 inRec = hdr.GetInRec ();
938 previousHop.x = hdr.GetLastPosx ();
939 previousHop.y = hdr.GetLastPosy ();
940 }
941
942 Vector myPos;
943 Ptr<MobilityModel> MM = m_ipv4->GetObject<MobilityModel> ();
944 myPos.x = MM->GetPosition ().x;
945 myPos.y = MM->GetPosition ().y;
946
947 if(inRec == 1 && CalculateDistance (myPos, Position) < CalculateDistance
(RecPosition,
Position)){
948 inRec = 0;
949 hdr.SetInRec(0);
950
951 NS_LOG_LOGIC ("No longer in Recovery to " << dst << " in " << myPos);
952 }
953
954 if(inRec){
955 p->AddHeader (hdr);
956 p->AddHeader (tHeader); //put headers back so that the RecoveryMode is
compatible
with Forwarding and SendFromQueue
957 RecoveryMode (dst, p, ucb, header);

```

```

958 return true;
959 }
960
961
962
963 uint32_t myUpdated = (uint32_t) m_locationService->GetEntryUpdateTime
(dst).GetSeconds
();
964 if (myUpdated > updated) //check if node has an update to the position of
destination
965 {
966 Position.x = m_locationService->GetPosition (dst).x;
967 Position.y = m_locationService->GetPosition (dst).y;
968 updated = myUpdated;
969 }
970
971
972 Ipv4Address nextHop;
973
974 if(m_neighbors.isNeighbour (dst))
975 {
976
977 nextHop = dst;
978 PositionHeader posHeader (Position.x, Position.y, updated, (uint64_t) 0,
(uint64_t
) 0, (uint8_t) 0, myPos.x, myPos.y);
979 p->AddHeader (posHeader);
980 p->AddHeader (tHeader);
981 Ptr<Ipv4Route> route = Create<Ipv4Route> ();
982 route->SetDestination (dst);
983 route->SetGateway (nextHop);
984
985 // FIXME: Does not work for multiple interfaces
986 route->SetOutputDevice (m_ipv4->GetNetDevice (1));
987 route->SetDestination (header.GetDestination ());
988 ucb (route, p, header);
989 return true;
990
991 }
992 else
993 {
994
995 nextHop = m_neighbors.BestNeighbor (Position, myPos);
996 //std::cout<<"NextHop == "<<nextHop<<"\n";
997
998 if (nextHop != Ipv4Address::GetZero ())
999 {
1000 PositionHeader posHeader (Position.x, Position.y, updated, (uint64_t) 0,
(
uint64_t) 0, (uint8_t) 0, myPos.x, myPos.y);
1001 p->AddHeader (posHeader);
1002 p->AddHeader (tHeader);
1003
1004
1005 Ptr<NetDevice> oif = m_ipv4->GetObject<NetDevice> ();
1006 Ptr<Ipv4Route> route = Create<Ipv4Route> ();
1007 route->SetDestination (dst);
1008 route->SetSource (header.GetSource ());

```

```

1009 route->SetGateway (nextHop);
1010
1011 // FIXME: Does not work for multiple interfaces
1012 route->SetOutputDevice (m_ipv4->GetNetDevice (1));
1013 route->SetDestination (header.GetDestination ());
1014 NS_ASSERT (route != 0);
1015 NS_LOG_DEBUG ("Exist route to " << route->GetDestination () << " from
interface " << route->GetOutputDevice ());
1016
1017
1018 NS_LOG_LOGIC (route->GetOutputDevice () << " forwarding to " << dst << "
from "
<< origin << " through " << route->GetGateway () << " packet " << p->GetUid
());
1019
1020 ucb (route, p, header);
1021 return true;
1022 }
1023 }
1024 hdr.SetInRec(1);
1025 hdr.SetRecPosx (myPos.x);
1026 hdr.SetRecPosy (myPos.y);
1027 hdr.SetLastPosx (Position.x); //when entering Recovery, the first edge
is the Dst
1028 hdr.SetLastPosy (Position.y);
1029
1030
1031 p->AddHeader (hdr);
1032 p->AddHeader (tHeader);
1033 RecoveryMode (dst, p, ucb, header);
1034
1035 NS_LOG_LOGIC ("Entering recovery-mode to " << dst << " in " << m_ipv4-
>GetAddress (1, 0
).GetLocal ());
1036 return true;
1037 }
1038
1039
1040
1041 void
1042 RoutingProtocol::SetDownTarget (IpL4Protocol::DownTargetCallback
callback)
1043 {
1044 m_downTarget = callback;
1045 }
1046
1047
1048 IpL4Protocol::DownTargetCallback
1049 RoutingProtocol::GetDownTarget (void) const
1050 {
1051 return m_downTarget;
1052 }
1053
1054 Ptr<Ipv4Route>
1055 RoutingProtocol::RouteOutput (Ptr<Packet> p, const Ipv4Header &header,
1056 Ptr<NetDevice> oif, Socket::SocketErrno &sockerr)
1057 {
1058 NS_LOG_FUNCTION (this << header << (oif ? oif->GetIfIndex () : 0));

```

```

1059
1060 if (!p)
1061 {
1062     return LoopbackRoute (header, oif); // later
1063 }
1064 if (m_socketAddresses.empty ())
1065 {
1066     sockerr = Socket::ERROR_NOROUTETOHOST;
1067     NS_LOG_LOGIC ("No gpsr interfaces");
1068     Ptr<Ipv4Route> route;
1069     return route;
1070 }
1071 sockerr = Socket::ERROR_NOTERROR;
1072 Ptr<Ipv4Route> route = Create<Ipv4Route> ();
1073 Ipv4Address dst = header.GetDestination ();
1074
1075 Vector dstPos = Vector (1, 0, 0);
1076
1077 if (!(dst == m_ipv4->GetAddress (1, 0).GetBroadcast ()))
1078 {
1079     dstPos = m_locationService->GetPosition (dst);
1080 }
1081
1082 if (CalculateDistance (dstPos, m_locationService->GetInvalidPosition ())
== 0 &&
m_locationService->IsInSearch (dst))
1083 {
1084     DeferredRouteOutputTag tag;
1085     if (!p->PeekPacketTag (tag))
1086     {
1087         p->AddPacketTag (tag);
1088     }
1089     return LoopbackRoute (header, oif);
1090 }
1091
1092 Vector myPos;
1093 Ptr<MobilityModel> MM = m_ipv4->GetObject<MobilityModel> ();
1094 myPos.x = MM->GetPosition ().x;
1095 myPos.y = MM->GetPosition ().y;
1096
1097 Ptr<Packet> p2 = p->Copy ();
1098
1099 TypeHeader tHeader (GPSRTYPE_POS);
1100 PositionHeader hdr;
1101 p2->RemoveHeader (tHeader);
1102
1103 if (tHeader.Get () == GPSRTYPE_POS)
1104 {
1105     if (m_ipv4->GetAddress (1, 0).GetLocal () == Ipv4Address ("10.0.0.51"))
1106         ++OutPackets;
1107 }
1108
1109
1110
1111 Ipv4Address nextHop;
1112
1113 if(m_neighbors.isNeighbour (dst))
1114 {

```

```

1115 nextHop = dst;
1116 }
1117 else
1118 {
1119 nextHop = m_neighbors.BestNeighbor (dstPos, myPos);
1120 }
1121
1122 if (nextHop != Ipv4Address::GetZero ())
1123 {
1124 NS_LOG_DEBUG ("Destination: " << dst);
1125
1126 route->SetDestination (dst);
1127 if (header.GetSource () == Ipv4Address ("102.102.102.102"))
1128 {
1129 route->SetSource (m_ipv4->GetAddress (1, 0).GetLocal ());
1130 }
1131 else
1132 {
1133 route->SetSource (header.GetSource ());
1134 }
1135 route->SetGateway (nextHop);
1136 route->SetOutputDevice (m_ipv4->GetNetDevice (m_ipv4-
>GetInterfaceForAddress (route
->GetSource ()))));
1137 route->SetDestination (header.GetDestination ());
1138 NS_ASSERT (route != 0);
1139 NS_LOG_DEBUG ("Exist route to " << route->GetDestination () << " from
interface "
<< route->GetSource ());
1140 if (oif != 0 && route->GetOutputDevice () != oif)
1141 {
1142 NS_LOG_DEBUG ("Output device doesn't match. Dropped.");
1143 sockerr = Socket::ERROR_NOROUTETOHOST;
1144 return Ptr<Ipv4Route> ();
1145 }
1146 return route;
1147 }
1148 else
1149 {
1150 DeferredRouteOutputTag tag;
1151 if (!p->PeekPacketTag (tag))
1152 {
1153 p->AddPacketTag (tag);
1154 }
1155 return LoopbackRoute (header, oif); //in RouteInput the recovery-mode is
called
1156 }
1157
1158 }
1159 }
1160 }
1161

```

ns3-gpsr-ptable-cpp

```

1 #include "gpsr-htable.h"
2 #include "ns3/simulator.h"
3 #include "ns3/log.h"
4 #include <algorithm>
5 #include "ns3/flow-monitor-module.h"
6
7 #include <iostream> //atanda
8 #include <fstream> // atanda
9
10
11
12 // #include "ns3/ipv4-flow-classifier.h"
13
14 // #include "ns3/flow-classifier.h"
15 // #include "ns3/network-module.h"
16 // #include "ns3/flow-id-tag.h"
17
18
19
20 NS_LOG_COMPONENT_DEFINE ("GpsrTable");
21
22
23 namespace ns3 {
24 namespace gpsr {
25
26
27 /*
28 GPSR position table
29 */
30
31 PositionTable::PositionTable ()
32 {
33 m_txErrorCallback = MakeCallback (&PositionTable::ProcessTxError, this);
34 m_entryLifeTime = Seconds (2); //FIXME fazer isto parametrizavel de acordo
com tempo
de hello
35
36 }
37
38 Time
39 PositionTable::GetEntryUpdateTime (Ipv4Address id)
40 {
41 if (id == Ipv4Address::GetZero ())
42 {
43 return Time (Seconds (0));
44 }
45 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i = m_table.find
(id);
46 return i->second.second;
47 }
48
49 /**
50 * \brief Adds entry in position table
51 */
52 void
53 PositionTable::AddEntry (Ipv4Address id, Vector position)
54 {

```

```

55 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i = m_table.find
(id);
56 if (i != m_table.end () || id.IsEqual (i->first))
57 {
58 m_table.erase (id);
59 m_table.insert (std::make_pair (id, std::make_pair (position,
Simulator::Now ()))));
60 return;
61 }
62
63
64 m_table.insert (std::make_pair (id, std::make_pair (position,
Simulator::Now ()))));
65 }
66
67 /**
68 * \brief Deletes entry in position table
69 */
70 void PositionTable::DeleteEntry (Ipv4Address id)
71 {
72 m_table.erase (id);
73 }
74
75 /**
76 * \brief Gets position from position table
77 * \param id Ipv4Address to get position from
78 * \return Position of that id or NULL if not known
79 */
80 Vector
81 PositionTable::GetPosition (Ipv4Address id)
82 {
83
84 NodeList::Iterator listEnd = NodeList::End ();
85 for (NodeList::Iterator i = NodeList::Begin (); i != listEnd; i++)
86 {
87 Ptr<Node> node = *i;
88 if (node->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal () == id)
89 {
90 return node->GetObject<MobilityModel> ()->GetPosition ();
91 }
92 }
93 return PositionTable::GetInvalidPosition ();
94
95 }
96
97 /**
98 * \brief Checks if a node is a neighbour
99 * \param id Ipv4Address of the node to check
100 * \return True if the node is neighbour, false otherwise
101 */
102 bool
103 PositionTable::isNeighbour (Ipv4Address id)
104 {
105
106 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i =
m_table.find (id);
107 if (i != m_table.end () || id.IsEqual (i->first))
108 {

```

```

109 return true;
110 }
111
112 return false;
113 }
114
115
116 /**
117  * \brief remove entries with expired lifetime
118  */
119 void
120 PositionTable::Purge ()
121 {
122
123 if(m_table.empty ())
124 {
125 return;
126 }
127
128 std::list<Ipv4Address> toErase;
129
130 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i =
m_table.begin ();
131 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator listEnd =
m_table.end ();
132
133 for (; !(i == listEnd); i++)
134 {
135
136 if (m_entryLifeTime + GetEntryUpdateTime (i->first) <= Simulator::Now ())
137 {
138 toErase.insert (toErase.begin (), i->first);
139
140 }
141 }
142 toErase.unique ();
143
144 std::list<Ipv4Address>::iterator end = toErase.end ();
145
146 for (std::list<Ipv4Address>::iterator it = toErase.begin (); it != end;
++it)
147 {
148
149 m_table.erase (*it);
150
151 }
152 }
153
154 /**
155  * \brief clears all entries
156  */
157 void
158 PositionTable::Clear ()
159 {
160 m_table.clear ();
161 }
162

```

```

//-----
-----
163
164 //ATANDA
165 uint32_t nodeDegree = 0; //for a forwarder node
166 uint32_t totalNodeDegreeCount = 0; //global value, increments by one for
each node,
even if it has 0 neighbours. Counts each category of node degree
167 uint32_t progressNodeDegree = 0; //neighbour nodes located in progress
region
of tranmission area
168 uint32_t totalForwardersDegreeCount = 0; //Of nodes that forwarded a
packet at
each case
169 uint32_t highestNodeDegree = 0; //
170
171 uint32_t forwarderCount = 0; //this is to totalForwarders
172 uint32_t totalForwarders = 0;
173 uint32_t totalFailForwarders = 0;
174 uint32_t totalIntoPerimeterRecovery = 0;
175 uint32_t countOfAllAttemptedGreedy = 0;
176 uint32_t countOfNotAttemptedGreedy = 0;
177
//-----
-----
178
179 double bestFoundDistance = 0; //THIS ATTRIBUTE WAS MOVED HERE FROM THE
INSIDE OF
THE METHOD: PositionTable::BestNeighbor (Vector position, Vector nodePos)
180
//-----
-----
181
182 const uint32_t NODE_DEGREES_ARRAY_SIZE = 20;
183 uint32_t nodeDegreeCount[NODE_DEGREES_ARRAY_SIZE] = {0};
184 uint32_t progressNodeDegreeCount[NODE_DEGREES_ARRAY_SIZE] = {0};
185
//-----
-----
186
187 const uint32_t FORWARDERS_ARRAY_SIZE2 = 22;
188 uint32_t forwarderGreedyDistanceCount2[FORWARDERS_ARRAY_SIZE2] = {0};
189 double forwarderGreedyDistanceRange[FORWARDERS_ARRAY_SIZE2] = {0, 10, 20,
30, 40, 50,
60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300, 320};
190
191 double forwarderGreedyDistance = 0; // Distance between current node
holding packet
and a node chosen to relay packet
192 double maxForwarderGreedyDistance = 0;
193 double minForwarderGreedyDistance = 0;
194 double forwarderHomingDistance = 0; //Distance between a node selected to
relay
packet and the destination

```

```

195 long double totalForwarderGreedyDistance = 0;
196 long double totalForwarderHomingDistance = 0;
197
198
199
200 const uint32_t PROGRESS_ARRAY_SIZE3 = 22;
201 uint32_t progressGreedyDistanceCount3[PROGRESS_ARRAY_SIZE3] = {0};
202 double progressGreedyDistanceRange[PROGRESS_ARRAY_SIZE3] = {0, 10, 20,
30, 40, 50, 60
, 70, 80, 90, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300, 320};
203
204 double progressGreedyDistance = 0; //Distance from current node to a
neighbour in
the progress region
205 double maxProgressGreedyDistance = 0;
206 double minProgressGreedyDistance = 0;
207 double progressHomingDistance = 0; //Distance between a progress region
neighbour and destination
208 long double totalProgressGreedyDistance = 0;
209 long double totalProgressHomingDistance = 0;
210 uint32_t progressNeighboursCount = 0; //All neighbours in progress region
counting
211
212
213
214 const uint32_t NON_PROGRESS_ARRAY_SIZE4 = 22;
215 uint32_t nonProgressNodeDistanceCount4[NON_PROGRESS_ARRAY_SIZE4] = {0};
216 double nonProgressNodeDistanceRange[NON_PROGRESS_ARRAY_SIZE4] = {0, 10,
20, 30, 40,
50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300,
320};
217
218 double nonProgressNodeDistance = 0;
219 double maxNonProgressNodeDistance = 0;
220 double minNonProgressNodeDistance = 0;
221 double nonProgressHomingDistance = 0;
222 long double totalNonProgressNodeDistance = 0;
223 long double totalNonProgressHomingDistance = 0;
224 uint32_t nonProgressNeighboursCount = 0;
225
226
227
228
229 Ptr<Ipv4FlowProbe> gpsrrxpackets;
230 uint32_t gpsrRxPackets = 0;
231 uint32_t currentRxPacketsValue = 0;
232 uint32_t previousRxPacketsValue = 0;
233 uint32_t changeInRxPacketsValue = 0;
234 uint32_t rxPacketsTotalGREEDY = 0;

```

```

235 uint32_t rxPacketsTotalELLIPSOID = 0;
236
237 uint32_t rxPacketsInTestWindowGREEDY = 0;
238 uint32_t rxPacketsInTestWindowELLIPSOID = 0;
239
240 uint32_t simSize = 0; //holds set size of simulation nodes from pagpsr-
main.cc
241 double simTotalTime = 0; //holds set time of simulation from pagpsr-
main.cc
242 double simPeriod = 0; ////simPeriod is {0s, 10s, 20s, 30s, ...}
243 bool simPeriodChangeOccured = false;
244 bool checkPacketsDeliveries = false;
245 uint32_t simPeriodChangeCount = 0;
246 bool gpsrRxPacketsValueChanged = false;
247
248 uint32_t modFactor = 0;
249
250 uint32_t countRunsOfGREEDY = 0;
251 uint32_t countRunsOfELLIPSOID = 0;
252
253
254
255 bool runGREEDY = true;
256 bool runELLIPSOID = false; //default
257 bool justRanGREEDY = false;
258 bool justRanELLIPSOID = false;
259
//-----
-----
260
261 //ATANDA
262 void PositionTable::TotalNodeDegreeCount()
263 {
264     for (uint32_t i = 0; i < NODE_DEGREES_ARRAY_SIZE; ++i)
265     {
266         totalNodeDegreeCount = totalNodeDegreeCount + (i * nodeDegreeCount[i]);
267         //std::cout << "nodeDegreeCount[" << i << "] total is " <<
nodeDegreeCount[i] <<
std::endl;
268     }
269 }
270
271 //ATANDA
272 void PositionTable::TotalForwardersDegreeCount()
273 {
274     for (uint32_t i = 0; i < NODE_DEGREES_ARRAY_SIZE; ++i)
275         totalForwardersDegreeCount = totalForwardersDegreeCount + (i *
progressNodeDegreeCount[i]);
276 }
277
//-----
-----
278
279 //ATANDA
280 void PositionTable::ProgressDistanceMeasures()
281 {

```

```

282 totalProgressGreedyDistance = totalProgressGreedyDistance +
progressGreedyDistance;
283 totalProgressHomingDistance = totalProgressHomingDistance +
progressHomingDistance;
284 progressNeighboursCount += 1;
285
286 if (progressGreedyDistance > maxProgressGreedyDistance) //Finds the
maximum
progress distance throughout the routing session
287 { maxProgressGreedyDistance = progressGreedyDistance;}
288
289 if (minProgressGreedyDistance == 0)
290 { minProgressGreedyDistance = progressGreedyDistance;}
291
292 if (progressGreedyDistance < minProgressGreedyDistance)
293 { minProgressGreedyDistance = progressGreedyDistance;}
294
295 for (uint32_t i = 0; i < PROGRESS_ARRAY_SIZE3; i++)
296 {
297 if ((progressGreedyDistance > progressGreedyDistanceRange[i]) && (
progressGreedyDistance <= progressGreedyDistanceRange[i+1]))
298 {
299 progressGreedyDistanceCount3[i+1] += 1;
300 }
301 }
302 }
303
304 //ATANDA
305 void PositionTable::NonProgressDistanceMeasures()
306 {
307 totalNonProgressNodeDistance = totalNonProgressNodeDistance +
nonProgressNodeDistance;
308 totalNonProgressHomingDistance = totalNonProgressHomingDistance +
nonProgressHomingDistance;
309 nonProgressNeighboursCount += 1;
310
311 if (nonProgressNodeDistance > maxNonProgressNodeDistance) //Finds the
maximum
non-progress distance throughout the routing session
312 { maxNonProgressNodeDistance = nonProgressNodeDistance;}
313
314 if (minNonProgressNodeDistance == 0)
315 { minNonProgressNodeDistance = nonProgressNodeDistance;}
316
317 if (nonProgressNodeDistance < minNonProgressNodeDistance)
318 { minNonProgressNodeDistance = nonProgressNodeDistance;}
319
320
321
322 //std::cout << "nonProgressNodeDistance " << nonProgressNodeDistance <<
std::endl;
323 for (uint32_t i = 0; i < NON_PROGRESS_ARRAY_SIZE4; i++)
324 {
325 if ((nonProgressNodeDistance > nonProgressNodeDistanceRange[i]) && (
nonProgressNodeDistance <= nonProgressNodeDistanceRange[i+1]))
326 {
327 nonProgressNodeDistanceCount4[i+1] += 1;
328 }

```

```

329 }
330 }
331
332 //ATANDA
333 void PositionTable::ForwarderDistanceMeasures2()
334 {
335     totalForwarderGreedyDistance = totalForwarderGreedyDistance +
forwarderGreedyDistance;
336     totalForwarderHomingDistance = totalForwarderHomingDistance +
forwarderHomingDistance;
337     forwarderCount += 1; //this may be equivalent to totalForwarders
338
339     if (forwarderGreedyDistance > maxForwarderGreedyDistance) //Finds the
maximum forwarder distance throughout the routing session
340     { maxForwarderGreedyDistance = forwarderGreedyDistance;}
341
342     if (minForwarderGreedyDistance == 0)
343     { minForwarderGreedyDistance = forwarderGreedyDistance;}
344
345     if (forwarderGreedyDistance < minForwarderGreedyDistance)
346     { minForwarderGreedyDistance = forwarderGreedyDistance;}
347
348     for (uint32_t i = 0; i < FORWARDERS_ARRAY_SIZE2; i++)
349     {
350         if ((forwarderGreedyDistance > forwarderGreedyDistanceRange[i]) && (
forwarderGreedyDistance <= forwarderGreedyDistanceRange[i+1]))
351         {
352             forwarderGreedyDistanceCount2[i+1] += 1;
353             //std::cout << "Forwarder Greedy Distance " << forwarderGreedyDistance
<< " Forwarder Greedy Distance Range "<<
forwarderGreedyDistanceRange[i+1] << std::endl;
354         }
355     }
356 }
357
358
//-----
-----
-----
359 //ATANDA
360 //Obtains total simulation time from pagpsr-main.cc at the beginning of
the process
361 uint32_t PositionTable::GetSimSize (uint32_t simSizeA)
362 {
363     std::cout << "simSize is: " << simSizeA << std::endl;
364     return simSize = simSizeA;
365 }
366
//-----
-----
-----
367
368 //ATANDA
369 double PositionTable::GetSimPeriod (double simPeriodA)
370 {
371     simPeriodChangeOccured = true;
372     //std::cout << "simPeriodChangeOccured " << simPeriodA << "s" <<
std::endl;

```

```

373 return simPeriod = simPeriodA;
374 }
375
//-----
-----
376
377 //ATANDA
378 uint32_t PositionTable::ObtainModFactor ()
379 {
380 return modFactor;
381 }
382
//-----
-----
383
384 //ATANDA
385 void PositionTable::CsvNodeDegreesWrite()
386 {
387 std::ofstream MyCsvFile;
388 MyCsvFile.open(
"/home/atanda/ns-allinone-3.23/ns-
3.23/results/gpsr_results/csvResult_NodeDegrees_C
OMPASS_30nto110n_200s.csv", std::ofstream::app);
389 if (MyCsvFile.is_open())
390 {
391 MyCsvFile << "----- SimSize: " << simSize << "
-----" << std::endl;
392 MyCsvFile << "SimSize" << "Node_Degree" <<"," << "Node_Degree_Count" <<
"," <<
"Progress_Count" << std::endl;
393 for (uint32_t i = 0; i < NODE_DEGREES_ARRAY_SIZE; ++i)
394 {
395 MyCsvFile << i << "," << nodeDegreeCount[i] << "," <<
progressNodeDegreeCount
[i] << std::endl;
396 }
397 }
398 else
399 { std::cout << "Unable to open Node Degrees csv output file" <<
std::endl;
400 }
401 MyCsvFile.close();
402 }
403
404
405 //ATANDA
406 void PositionTable::CsvDistancesWrite2()
407 {
408 std::ofstream MyCsvFile2;
409 MyCsvFile2.open(
"/home/atanda/ns-allinone-3.23/ns-
3.23/results/gpsr_results/csvResult_Distances_COM
PASS_30nto110n_200s.csv", std::ofstream::app);
410 if (MyCsvFile2.is_open())
411 {
412 MyCsvFile2 << "----- SimSize: " << simSize << "

```

```

-----" << std::endl;
413 MyCsvFile2 << "Greedy_distance_limit, No_of_forwarders,
No_of_all_progress_candidates, No_of_all_Non-progress_candidates" <<
std::endl;
414 uint32_t j = 0;
415 for (uint32_t i = 0; i < FORWARDERS_ARRAY_SIZE2; ++i) // Same as other
array
SIZE2, SIZE3, and SIZE4
416 {
417 MyCsvFile2 << j << "," << forwarderGreedyDistanceCount2[i] <<"," <<
progressGreedyDistanceCount3[i] <<"," << nonProgressNodeDistanceCount4[i] <<
std::endl;
418 if (i < 10)
419 { j = j + 10; } // to print {0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
420 else
421 { j = j + 20; } // to print {120, 140, 160, 180, 200, 220, 240, 260, 280,
300, 320}
422 }
423 }
424 else
425 {
426 std::cout << "Unable to open Distances csv output file for write" <<
std::endl;
427 }
428 MyCsvFile2.close();
429 }
430
//-----
-----

431
432 //ATANDA
433 void PositionTable::CsvDistancesAveragesWrite3()
434 {
435 std::ofstream MyCsvFile3;
436 MyCsvFile3.open(
"/home/atanda/ns-allinone-3.23/ns-
3.23/results/gpsr_results/csvResult_DistAndDegAve
rages_COMPASS_30nto110n_200s.csv", std::ofstream::app);
437 if (MyCsvFile3.is_open())
438 {
439 //1.Total Failed Forwarders (TFF), 2.Total Forwarders (TF), 3.Average
Node
Degree (AND), 4.Average Forwarders' Node Degree (AFND), 5.Average Forwarder
Greedy Distance (AFGD), 6.Average Forwarder Homing Distance (AFHD),
7.Forwarder
Count (FC), 8.Min Forwarder Greedy Distance (MinFGD), 9.Max Forwarder Greedy
Distance (MaxFGD), 10.Average Progress Greedy Distance (APGD), 11.Average
Progress Homing Distance (APHD), 12.Progress Neighbours Count (PNC), 13.Min
Progress Greedy Distance (MinPGD), 14.Max Progress Greedy Distance (MaxPGD),
15.Average Non-Progress Greedy (AnPND) Distance, 16.Average Non-Progress
Homing
Distance (AnPHD), 17.Non-Progress Neighbours Count (nPNC), 18.Min Non-
Progress
Node Distance (MinnPND), 19.Max Non-Progress Node Distance (MaxnPND)
440 MyCsvFile3 << "----- SimSize: " << simSize << "
-----" << std::endl;

```

```

441 MyCsvFile3 << "1.TFF" << "," << "2.TF" << "," << "3.AND" << "," <<
"4.AFND" <<
", " << "5.AFGD" << "," << "6.AFHD" << "," << "7.FC" << "," << "8.MinFGD" <<
", "
<< "9.MaxFGD" << "," << "10.APGD" << "," << "11.APHD" << "," << "12.PNC" <<
", " << "13.MinPGD" << "," << "14.MaxPGD" << "," << "15.AnPND" << "," <<
"16.AnPHD" << "," << "17.nPNC" << "," << "18.MinnPND" << "," << "19.MaxnPND"
<<
std::endl;
442
443 MyCsvFile3 << totalFailForwarders <<"," << totalForwarders <<"," <<
totalNodeDegreeCount/(totalIntoPerimeterRecovery + totalForwarders +
totalFailForwarders) <<"," << totalForwardersDegreeCount/totalForwarders
<<"," <<
<< totalForwarderGreedyDistance/forwarderCount <<"," <<
totalForwarderHomingDistance/forwarderCount <<"," << forwarderCount <<"," <<
minForwarderGreedyDistance <<"," << maxForwarderGreedyDistance <<"," <<
totalProgressGreedyDistance/progressNeighboursCount <<"," <<
totalProgressHomingDistance/progressNeighboursCount <<"," <<
progressNeighboursCount <<"," << minProgressGreedyDistance <<"," <<
maxProgressGreedyDistance <<"," << totalNonProgressNodeDistance/
nonProgressNeighboursCount <<"," << totalNonProgressHomingDistance/
nonProgressNeighboursCount <<"," << nonProgressNeighboursCount <<"," <<
minNonProgressNodeDistance <<"," << maxNonProgressNodeDistance << std::endl;
444
445 }
446 else
447 {
448 std::cout << "Unable to open Average Distances csv output file for write"
<< std
::endl;
449 }
450 MyCsvFile3.close();
451 }
452
//-----
-----

453
454 //ATANDA
455 void PositionTable::BestNeighbourOutputs ()
456 {
457 TotalNodeDegreeCount();
458 TotalForwardersDegreeCount();
459
460 std::cout << "-----NODE DEGREE MEASURES----- " <<
std::endl
;
461 std::cout << "Total Not Attempted Greedy: " << countOfNotAttemptedGreedy
<< std:::
endl;
462 std::cout << "Total Attempted Greedy: " << countOfAllAttemptedGreedy <<
std::endl
;
463 std::cout << "Total into Perimeter Mode: " << totalIntoPerimeterRecovery
<< std:::
endl;
464 std::cout << "Total Forwarders " << totalForwarders << std::endl;

```

```

465 std::cout << "Total Failed Forwarders: " << totalFailForwarders <<
std::endl;
466
467 std::cout << "The highest node degree is " << highestNodeDegree << "
//IMPORTANT
to check in gpsr-ptable.cc that NODE_DEGREES_ARRAY_SIZE " <<
NODE_DEGREES_ARRAY_SIZE << " >= "<< highestNodeDegree << std::endl;
468 std::cout << "Average node degree " << totalNodeDegreeCount/(
totalIntoPerimeterRecovery + totalForwarders + totalFailForwarders) <<
std::endl;
469 std::cout << "Average forwarders' node degree " <<
totalForwardersDegreeCount/
totalForwarders << std::endl;
470 //std::cout << "Total direct into perimeter recovery: " <<
totalIntoPerimeterRecovery << std::endl; //atanda
471 //std::cout << "Total node degrees: " << totalNodeDegreeCount <<
std::endl;
472 //std::cout << "Total forwarders' Degrees: " <<
totalForwardersDegreeCount <<
std::endl;
473
474 std::cout << "-----FORWARDER DISTANCE MEASURES-----
" <<
std::endl;
475 std::cout << "Average Forwarder Greedy Distance " <<
totalForwarderGreedyDistance
/forwarderCount << std::endl;
476 std::cout << "Average Forwarder Homing Distance " <<
totalForwarderHomingDistance
/forwarderCount << std::endl;
477 //std::cout << "Total Forwarder Greedy Distance " <<
totalForwarderGreedyDistance << std::endl;
478 std::cout << "Forwarder Count " << forwarderCount << std::endl;
479 std::cout << "MinForwarderGreedyDistance " << minForwarderGreedyDistance
<< std::
endl;
480 std::cout << "MaxForwarderGreedyDistance " << maxForwarderGreedyDistance
<< std::
endl;
481
482 std::cout << "-----PROGRESS DISTANCE MEASURES-----
" <<
std::endl;
483 std::cout << "Average Progress Greedy Distance " <<
totalProgressGreedyDistance/
progressNeighboursCount << std::endl;
484 std::cout << "Average Progress Homing Distance " <<
totalProgressHomingDistance/
progressNeighboursCount << std::endl;
485 //std::cout << "Total Progress Forward Distance " <<
totalProgressGreedyDistance
<< std::endl;
486 std::cout << "Progress Neighbours Count " << progressNeighboursCount <<
std::endl
;
487 std::cout << "MinProgressGreedyDistance " << minProgressGreedyDistance <<
std::
endl;

```

```

488 std::cout << "MaxProgressGreedyDistance " << maxProgressGreedyDistance <<
std::
endl;
489
490 std::cout << "-----NON-PROGRESS DISTANCE MEASURES-----
" <<
std::endl;
491 std::cout << "Average Non-Progress Node Distance " <<
totalNonProgressNodeDistance/nonProgressNeighboursCount << std::endl;
492 std::cout << "Average Non-Progress Homing Distance " <<
totalNonProgressHomingDistance/nonProgressNeighboursCount << std::endl;
493 //std::cout << "Total Non-Progress Distance " <<
totalNonProgressNodeDistance <<
std::endl;
494 std::cout << "Non-Progress Neighbours Count " <<
nonProgressNeighboursCount <<
std::endl;
495 std::cout << "Min-Non-ProgressNodeDistance " <<
minNonProgressNodeDistance << std
::endl;
496 std::cout << "Max-Non-ProgressNodeDistance " <<
maxNonProgressNodeDistance << std
::endl;
497
498 std::cout << "-----PACKETS
DELIVERIES----- " << std::endl;
499 std::cout << "rxPacketsTotalGREEDY : " << rxPacketsTotalGREEDY <<
std::endl;
500 std::cout << "rxPacketsTotalELLIPSOID : " << rxPacketsTotalELLIPSOID <<
std::
endl; //ELLIPSOID or other, such as GREEDY-COMPASS
501 std::cout << "rxPacketsInTestWindowGREEDY : " <<
rxPacketsInTestWindowGREEDY <<
std::endl;
502 std::cout << "rxPacketsInTestWindowELLIPSOID : " <<
rxPacketsInTestWindowELLIPSOID << std::endl;
503 std::cout << "The value of gpsrRxPackets is " << gpsrRxPackets <<
std::endl;
//atanda
504
505 std::cout << "RAN GREEDY: " << countRunsOfGREEDY << std::endl;
506 std::cout << "RAN ELLIPSOID: " << countRunsOfELLIPSOID << std::endl;
507 std::cout << "-----
" <<
std::endl;
508 //CsvNodeDegreesWrite(); //write to file what makes up the totals:
../gpsr_results/csvResult
509 //CsvDistancesWrite2();
510 //CsvDistancesAveragesWrite3(); //Note: Another write method,
CsvPacketsAveragesWrite, is located in pagpsr-main.cc
511
512 }
513
514
//-----
-----
515

```

```

516
517
518 /**
519 * \brief Gets next hop according to GPSR protocol
520 * \param position the position of the destination node
521 * \param nodePos the position of the node that has the packet
522 * \return Ipv4Address of the next hop, Ipv4Address::GetZero () if no
neighbour was found
in greedy mode
523 */
524 //BESTNEIGHBOUR: Greedy or NC or MFR or NFP or S* or CD-AD_DISTANCES
525 Ipv4Address
526 PositionTable::BestNeighbor (Vector position, Vector nodePos)
527 {
528 Purge ();
529
530 if (m_table.empty ())
531 {
532 NS_LOG_DEBUG ("BestNeighbor table is empty; Position: " << position);
533 countOfNotAttemptedGreedy += 1;
534 totalFailForwarders += 1; //atanda, 1st point of incrementing
535 return Ipv4Address::GetZero (); //Implies drop packets? atanda
536 } //i.e. table is empty (no neighbours)
537
538 double initialDistance = CalculateDistance (nodePos, position);
//Neighbours with
distance greater than initialDistance are out of advance region range -atanda
539 //std::cout << "The value of initialDistance is " << initialDistance <<
std::endl;
//atanda
540
541 Ipv4Address bestFoundID = m_table.begin ()->first;
542 bestFoundDistance = CalculateDistance (m_table.begin ()->second.first,
position);
//THIS ATTRIBUTE WAS DEFINED HERE IN THE ORIGINAL CODE
543
544 double tempForwarderGreedyDistance = CalculateDistance (m_table.begin ()-
>second.
first, nodePos); //atanda
545 //std::cout << "The value of tempForwarderGreedyDistance is " <<
tempForwarderGreedyDistance << std::endl; //atanda
546 double tempForwarderHomingDistance = bestFoundDistance; //atanda
547
548 //std::cout << "The value of m_table.begin ()->first is " <<
m_table.begin ()->first <<
std::endl; //atanda
549
550
//-----
-----
551 //ATANDA
552 nodeDegree = 0;
553 progressNodeDegree = 0;
554
555 //ATANDA Begin Node Degree finding, i.e. number of neighbours of the
current
node
556 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator j;

```

```

557 for (j = m_table.begin (); !(j == m_table.end ()); j++)
558 {
559 nodeDegree += 1;
560 }
561
//-----
-----
562
563 /*
564 //ATANDA - flowId value - begin
565 Ptr<Ipv4FlowProbe> probe;
566 uint32_t gpsrFlowId = probe->GpsrFlowId (); //GpsrFlowId() defined in
ipv4-flow-probe.cc
567 //std::cout << "The value of gpsrflowId is " << gpsrFlowId << std::endl;
//atanda
568 //ATANDA - flowId value - end
569
570
571 //BEGIN assigning method to use based on PDR performance wrt flowID,
atanda
572 uint32_t greedyChoice = 1; //atanda, default choice
573
574
575 //This piece appears to make no di
576 //if (gpsrFlowId > 0) //The initial flow value reading is 0, i.e. no flow
yet
577 //{
578
579
580 //ATANDA - using network packets flow feedback
581 std::ifstream infile; //!!!!REMEMBER TO CHANGE TO THE RIGHT FILE!!!!
582
infile.open("/home/atanda/ns-allinone-3.23/ns-
3.23/src/gpsr/model/simufile_multi
_GREEDYlorSTAR2_lpr_110n_2200s.txt");
583
584 if (infile.fail())
585 {
586 std::cout << "Could not open greedy-method choice simulation file" <<
std::endl; //atanda
587 }
588
589 uint32_t fileFlowId = 0;
590 uint32_t greedyMethod = 1; //default
591
592 if (!infile.fail())
593 {
594
595 bool notEqualFlowId = true;
596
597 while ((infile >> fileFlowId >> greedyMethod) && (notEqualFlowId))
598 {
599
600 if (gpsrFlowId == fileFlowId)
601 {
602 greedyChoice = greedyMethod;

```

```

603 //std::cout << "gpsrFlowId: " << gpsrFlowId << " fileFlowId: " <<
fileFlowId
<< " greedyMethod: " << greedyMethod << " greedyChoice: " << greedyChoice <<
std::endl; //atanda
604 notEqualFlowId = false;
605 }
606 }
607 infile.close();
608 }
609 //END assigning method to use based on PDR performance wrt flowID, atanda
610 */
611
612
//-----
-----
613
614 /*
615 //atanda flow-based Multi-greedy switching
616 if (greedyChoice == 1)
617 {
618 */
619
//-----
-----
620 /*
621 //nodeDegree-based Multi-greedy switching
622 if ((nodeDegree == 1) || (nodeDegree == 5) || (nodeDegree == 8))
623 {
624 */
625
626
//-----
-----
627
628
629
630 //ATANDA BEGIN PDR sampling and process. Note that either runGreedy or
runEllipsoid
should be initiated to true
631 gpsrRxPacketsValueChanged = false;
632 previousRxPacketsValue = gpsrRxPackets;
633 gpsrRxPackets = gpsrrxpackets->GpsrRxPackets (); //1 packet received, if
any, for
each call
634
635 if (previousRxPacketsValue != gpsrRxPackets)
636 {
637 gpsrRxPacketsValueChanged = true;
638
639 if (justRanGREEDY)
640 rxPacketsTotalGREEDY += 1;
641
642 if (justRanELLIPSOID)
643 rxPacketsTotalELLIPSOID += 1;
644

```

```

645 //std::cout << "gpsrRxPacketsValueChanged" << std::endl; //atanda
646 //std::cout << "The value of gpsrRxPackets is " << gpsrRxPackets <<
std::endl;
//atanda
647 //std::cout << "The value of previousRxPacketsValue is " <<
previousRxPacketsValue << std::endl; //atanda
648 }//Endif previousRxPacketsValue
649
650
651 if (simPeriodChangeOccured)
652 {
653 // std::cout << "A change to simPeriod occurred here" << std::endl;
654 modFactor = 100; //atanda ADJUST cycle of time to check packets
deliveries
point, e.g. after every 20s is set to 2
655 if ((++simPeriodChangeCount % modFactor) == 0)
656 { checkPacketsDeliveries = true;
657 // std::cout << "simPeriodChangeCount " << simPeriodChangeCount <<
std::endl;
658 }
659 else { checkPacketsDeliveries = false; }
660 simPeriodChangeOccured = false;
661 }
662 else
663 {
664 //std::cout << "No change to simPeriod" << std::endl;
665 }//Endif simPeriodChangeOccured
666
667
668 if (!(checkPacketsDeliveries))
669 {
670 rxPacketsInTestWindowGREEDY = 0;
671 rxPacketsInTestWindowELLIPSOID = 0;
672 }
673
674
675 if (checkPacketsDeliveries)
676 {
677 runGREEDY = false;
678 runELLIPSOID = false;
679
680 if ((Simulator::Now () > (Seconds(simPeriod))) && (Simulator::Now () <
Seconds(
simPeriod + 1))) //atanda time-window checkduration ADJUST simPeriod +1, +2,
+3,
etc.
681 {
682 //std::cout << "Seconds(simPeriod)" << Seconds(simPeriod) << std::endl;
683 //std::cout << "Seconds(simPeriod+1)" << Seconds(simPeriod+1) <<
std::endl;
684 runGREEDY = true;
685
686 //std::cout << "running GREEDY test " << std::endl;
687 if (gpsrRxPacketsValueChanged)
688 rxPacketsInTestWindowGREEDY += 1;
689 //std::cout << "rxPacketsInTestWindowGREEDY: " <<
rxPacketsInTestWindowGREEDY << std::endl;
690 }

```

```

691 else if ((Simulator::Now () > (Seconds(simPeriod + 1))) &&
(Simulator::Now () <
Seconds(simPeriod + 2))) //ADJUST simPeriod +1, +2, +3, etc. Ensure previous
simperiod + x is lesser
692 {
693 runELLIPSOID = true;
694
695 //std::cout << "running ELLIPSOID test " << std::endl;
696 if (gpsrRxPacketsValueChanged)
697 rxPacketsInTestWindowELLIPSOID += 1;
698 //std::cout << "rxPacketsInTestWindowELLIPSOID: " <<
rxPacketsInTestWindowELLIPSOID << std::endl;
699 }
700 else
701 {
702 if (rxPacketsInTestWindowGREEDY >= rxPacketsInTestWindowELLIPSOID)
703 {
704 //std::cout << "rxPacketsInTestWindowGREEDY >=
rxPacketsInTestWindowELLIPSOID " << std::endl;
705 //std::cout << "rxPacketsInTestWindowGREEDY " <<
rxPacketsInTestWindowGREEDY << std::endl;
706 //std::cout << "rxPacketsInTestWindowELLIPSOID " <<
rxPacketsInTestWindowELLIPSOID << std::endl;
707
708 runGREEDY = true;
709 }
710 else
711 {
712 //std::cout << "rxPacketsInTestWindowELLIPSOID >
rxPacketsInTestWindowGREEDY " << std::endl;
713 //std::cout << "rxPacketsInTestWindowGREEDY " <<
rxPacketsInTestWindowGREEDY << std::endl;
714 //std::cout << "rxPacketsInTestWindowELLIPSOID " <<
rxPacketsInTestWindowELLIPSOID << std::endl;
715
716 runELLIPSOID = true;
717 }
718 }
719 }// ATANDA END PDR sampling and process
720
//-----
-----
721
722
723 countOfAllAttemptedGreedy += 1; //equivalent to totalForwarders +
totalFailedForwarders
724
725
726 if (runGREEDY)
727 {
728
729
730
731 //std::cout << "running GREEDY " << std::endl;
732 //GREEDY BEGIN This is normal GPSR's
733 countRunsOfGREEDY += 1;
734

```

```

735 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
736 for (i = m_table.begin (); !(i == m_table.end ()); i++)
737 {
738
739 //ATANDA begin
740 //std::cout << "The value of i->first is " << i->first << std::endl;
//atanda
741 if ((CalculateDistance (i->second.first, position)) < initialDistance)
742 {
743 progressGreedyDistance = CalculateDistance (i->second.first, nodePos);
744 progressHomingDistance = bestFoundDistance;
745 ProgressDistanceMeasures();
746 //if (progressGreedyDistance <= 10)
747 //std::cout << "progressGreedyDistance: " << progressGreedyDistance <<
std::endl; //atanda
748 //std::cout << "ProgressHomingDistance: " << progressHomingDistance <<
std::endl; //atanda
749 }
750 else
751 {
752 nonProgressNodeDistance = CalculateDistance (i->second.first, nodePos);
753 nonProgressHomingDistance = CalculateDistance (i->second.first,
position);
754 NonProgressDistanceMeasures();
755
756 //std::cout << "Non-ProgressNodeDistance: " << nonProgressNodeDistance
<< std::endl; //atanda
757 //std::cout << "Non-progressHomingDistance: " <<
nonProgressHomingDistance << std::endl; //atanda
758 }
759 //std::cout << "Sentinel2 " << std::endl; //atanda
760 //ATANDA end
761
762 if (bestFoundDistance > CalculateDistance (i->second.first, position))
763 {
764 bestFoundID = i->first;
765 //std::cout << "The value of bestFoundID is " << bestFoundID <<
std::endl;
//atanda
766 bestFoundDistance = CalculateDistance (i->second.first, position);
767 tempForwarderGreedyDistance = progressGreedyDistance;//atanda
768 tempForwarderHomingDistance = progressHomingDistance;//atanda
769 //std::cout << "tempForwarderGreedyDistance: " <<
tempForwarderGreedyDistance
<< std::endl; //atanda
770 }
771 }
772 justRanELLIPSOID = false; ////atanda This reverses that of ELLIPSOID so
as to set
that of GREEDY at the end below
773 justRanGREEDY = true;//atanda
774 //GREEDY END
775
776
777
778 } //end runGREEDY
779
780

```

```

781
782 else //if (runELLIPSOID)
783 {
784
785
786 /*
787 } //for existDistance
788 else
789 {
790 */
791
792 /*
793 }
794 else if (nodeDegree == 1)
795 {
796 */
797
798 /*
799 if (runELLIPSOID)
800 { */
801
802
803 //ATANDA
804 //S* BEGIN
805
806 countRunsOfELLIPSOID += 1;
807
808
809 //metric of Star's f = g + n
810 double starBestFoundDistance = (CalculateDistance (m_table.begin ()-
>second.first,
nodePos) + CalculateDistance (m_table.begin ()->second.first, position));
811 //std::cout << "The value of initialStarBestFoundDistance is " <<
starBestFoundDistance << std::endl; //atanda
812
813 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
814 for (i = m_table.begin (); !(i == m_table.end ()); i++)
815 {
816 //ATANDA begin
817 //std::cout << "The value of i->first is " << i->first << std::endl;
//atanda
818 if ((CalculateDistance (i->second.first, position)) < initialDistance)
819 {
820 progressGreedyDistance = CalculateDistance (i->second.first, nodePos);
821 progressHomingDistance = bestFoundDistance;
822 ProgressDistanceMeasures();
823 //if (progressGreedyDistance <= 10)
824 //std::cout << "progressGreedyDistance: " << progressGreedyDistance <<
std::endl; //atanda
825 //std::cout << "ProgressHomingDistance: " << progressHomingDistance <<
std::endl; //atanda
826 }
827 else
828 {
829 nonProgressNodeDistance = CalculateDistance (i->second.first, nodePos);
830 nonProgressHomingDistance = CalculateDistance (i->second.first,
position);
831 NonProgressDistanceMeasures();

```

```

832
833 //std::cout << "Non-ProgressNodeDistance: " << nonProgressNodeDistance
<< std::endl; //atanda
834 //std::cout << "Non-progressHomingDistance: " <<
nonProgressHomingDistance << std::endl; //atanda
835 }
836 //std::cout << "Sentinel2 " << std::endl; //atanda
837 //ATANDA end
838
839 //std::cout << "The value of i->first is " << i->first << std::endl;
//atanda
840 //std::cout << "The value of i->first Distance is " << CalculateDistance
(i->second.first, position) << std::endl; //atanda
841 //std::cout << "The value of starFoundDistance is " << (CalculateDistance
(i->second.first, nodePos) + CalculateDistance (i->second.first, position))
<<
std::endl; //atanda
842
843 if (((CalculateDistance (i->second.first, position) < initialDistance))
&& ((
CalculateDistance (i->second.first, position)) + (CalculateDistance (i-
>second.
first, nodePos)) < starBestFoundDistance))
844 {
845 bestFoundID = i->first;
846 // std::cout << "The value of bestFoundID is " << bestFoundID <<
std::endl;
//atanda
847 bestFoundDistance = CalculateDistance (i->second.first, position);
848 starBestFoundDistance = CalculateDistance (i->second.first, nodePos) +
bestFoundDistance;
849 // std::cout << "The value of bestFoundDistance is " << bestFoundDistance
<<
std::endl; //atanda
850 //std::cout << "The value of starBestFoundDistance is " <<
starBestFoundDistance << std::endl; //atanda
851 tempForwarderGreedyDistance = progressGreedyDistance;
852 tempForwarderHomingDistance = progressHomingDistance;
853 //std::cout << "tempForwarderGreedyDistance: " <<
tempForwarderGreedyDistance
<< std::endl; //atanda
854 }
855 }
856 justRanGREEDY = false; //This reverses that of GREEDY so as to set that
of
ELLIPSOID at the end below
857 justRanELLIPSOID = true;
858 //S* END
859
860
861
862 } // END runELLIPSOID
863
864
865 /*
866 } //END for existDistance
867 */
868 /*

```

```

869 }
870 else //nodeDegree
871 {
872 */
873
874 /*
875 }
876 else //if (greedyChoice == 2)
877 {
878 */
879
880
881 /*
882 else
883 {
884 */
885
886 /*
887 //ATANDA
888 //GREEDY-COMPASS (Stojmenovic's) BEGIN - atanda
889
890 // #define PI = 3.14159265;
891 double a, b, c;
892
893 //Calculate baseline values
894 // Compute the aX + bY + C = 0 line equation formed by (x0, y0) and (x1,
y1).
Template obtained from obstacles.cc in EstiNet
895 if (position.y == nodePos.y)
896 {
897 a = 0;
898 b = 1;
899 c = -1 * nodePos.y;
900 if (position.x == nodePos.x)
901 a = b = c = 0;
902 }
903 else if (position.x == nodePos.x)
904 {
905 a = 1;
906 b = 0;
907 c = -1 * nodePos.x;
908 }
909 else
910 {
911 b = -1;
912 a = (position.y - nodePos.y) / (position.x - nodePos.x);
913 c = nodePos.y - nodePos.x * a;
914 } //END Calculate baseline values
915
916
917
918 double slopeDistance = CalculateDistance (m_table.begin ()->second.first,
nodePos);
919 double perpendDistance = fabs(((a * m_table.begin ()->second.first.x) +
(b *
m_table.begin ()->second.first.y) + c) / sqrt((a * a) + (b * b)));
920 double acuteAngle = (asin(perpendDistance / slopeDistance)*180) / M_PI;
921

```

```

922 Ipv4Address aboveBestID = m_table.begin ()->first;
923 Ipv4Address belowBestID = m_table.begin ()->first;
924 double aboveBestAngle = 90;
925 double belowBestAngle = 90;
926 double aboveBestFoundDistance = CalculateDistance (m_table.begin ()-
>second.first,
position);
927 double belowBestFoundDistance = CalculateDistance (m_table.begin ()-
>second.first,
position);
928 double neiToDestDistance;
929
930
931 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
932 for (i = m_table.begin (); !(i == m_table.end ()); i++)
933 {
934
935 slopeDistance = CalculateDistance (i->second.first, nodePos);
936 perpendDistance = (((a * i->second.first.x) + (b * i->second.first.y) +
c) /
sqrt((a * a) + (b * b)));
937 acuteAngle = (asin(fabs(perpendDistance) / slopeDistance)*180) / M_PI;
938 neiToDestDistance = CalculateDistance (i->second.first, position);
939 //Is the perpendDistance negative or positive. Observe removal of the
'fabs'
absolute value function
940 if ((neiToDestDistance < initialDistance) && (perpendDistance < 0))
941 {
942
943 if (acuteAngle < belowBestAngle)
944 {
945 belowBestAngle = acuteAngle;
946 //std::cout << "Below best angle is " << belowBestAngle << std::endl;
//atanda
947 belowBestID = i->first;
948 //std::cout << "Below best ID is " << belowBestID << std::endl; //atanda
949 belowBestFoundDistance = neiToDestDistance;
950 //std::cout << "Below best distance is " << belowBestDistance <<
std::endl; //atanda
951 }
952 progressGreedyDistance = CalculateDistance (i->second.first, nodePos);
953 progressHomingDistance = belowBestFoundDistance; //bestFoundDistance;
954 ProgressDistanceMeasures();
955
956 }
957 else if ((neiToDestDistance < initialDistance) && (perpendDistance >= 0))
958 {
959 if (acuteAngle < aboveBestAngle)
960 {
961 //std::cout << "Above perpendicular distance is " << (((a *
i->second.first.x) + (b * i->second.first.y) + c) / sqrt((a * a) + (b *
b))) << std::endl; //atanda
962 aboveBestAngle = acuteAngle;
963 //std::cout << "Above best angle is " << aboveBestAngle << std::endl;
//atanda
964 aboveBestID = i->first;
965 //std::cout << "Above best ID is " << aboveBestID << std::endl; //atanda
966 aboveBestFoundDistance = neiToDestDistance;

```

```

967 //std::cout << "Above best distance is " << aboveBestFoundDistance <<
std::endl; //atanda
968 }
969 }
970 progressGreedyDistance = CalculateDistance (i->second.first, nodePos);
971 progressHomingDistance = aboveBestFoundDistance; //bestFoundDistance;
972 ProgressDistanceMeasures();
973 }
974 //Greedy part computation -atanda
975 if (aboveBestFoundDistance <= belowBestFoundDistance)
976 {
977 bestFoundID = aboveBestID;
978 bestFoundDistance = aboveBestFoundDistance;
979 }
980 else
981 {
982 bestFoundID = belowBestID;
983 bestFoundDistance = belowBestFoundDistance;
984 }
985
986 //justRanELLIPSOID = false; ////atanda This reverses that of ELLIPSOID so
as to set
that of GREEDY at the end below
987 //justRanGREEDY = true; //atanda
988 justRanGREEDY = false; //This reverses that of GREEDY so as to set that
of ELLIPSOID
at the end below
989 justRanELLIPSOID = true;
990 //std::cout << "Best found ID is " << bestFoundID << std::endl; //atanda
991 //GREEDY-COMPASS END
992 */
993 // } // END runGREEDY-COMPASS
994
995
996 /*
997 }
998 else
999 {
1000 */
1001
1002 /*
1003 } //END nodeDegree if conditions
1004 */
1005
1006 /*
1007 } //END greedyChoice gpsrFlowId 'if conditions', atanda
1008 */
1009
1010 /*
1011 //ATANDA
1012 //RPM - Random Progress Method
1013 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
1014 for (i = m_table.begin (); !(i == m_table.end ()); i++)
1015 {
1016 //std::cout << "The value of i->first is " << i->first << std::endl;
//atanda
1017 //std::cout << "The value of i->first Distance is " << CalculateDistance
(i->second.first, position) << std::endl; //atanda

```

```

1018 if (bestFoundDistance < initialDistance)
1019 {
1020 bestFoundID = i->first;
1021 //std::cout << "The value of bestFoundID is " << bestFoundID <<
std::endl;
//atanda
1022 bestFoundDistance = CalculateDistance (i->second.first, position);
1023 //std::cout << "The value of initialDistance is " << initialDistance <<
std::endl; //atanda
1024 //std::cout << "The value of bestFoundDistance is " << bestFoundDistance
<<
std::endl; //atanda
1025 }
1026 }
1027 //RPM END
1028 */
1029
1030
1031 /*
1032 //ATANDA
1033 //NC BEGIN
1034 double ncBestFoundDistance = CalculateDistance (m_table.begin ()-
>second.first,
nodePos);
1035 //std::cout << "The value of initialNCbestFoundDistance is " <<
ncBestFoundDistance
<< std::endl; //atanda
1036 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
1037 for (i = m_table.begin (); !(i == m_table.end ()); i++)
1038 {
1039
1040 //std::cout << "The value of i->first is " << i->first << std::endl;
//atanda
1041 //std::cout << "The value of i->first Distance is " << CalculateDistance
(i->second.first, position) << std::endl; //atanda
1042 //std::cout << "The value of ncBestFoundDistance is " <<
CalculateDistance
(i->second.first, nodePos) << std::endl; //atanda
1043 if (((CalculateDistance (i->second.first, position) < initialDistance)
&&
(CalculateDistance (i->second.first, nodePos) < ncBestFoundDistance)))
1044 {
1045 bestFoundID = i->first;
1046 //std::cout << "The value of bestFoundID is " << bestFoundID <<
std::endl;
//atanda
1047 bestFoundDistance = CalculateDistance (i->second.first, position);
1048 ncBestFoundDistance = CalculateDistance (i->second.first, nodePos);
1049 //std::cout << "The value of bestFoundDistance is " << bestFoundDistance
<<
std::endl; //atanda
1050 //std::cout << "The value of ncbestFoundDistance is " <<
ncBestFoundDistance
<< std::endl; //atanda
1051 }
1052 }
1053 //NC END
1054 */

```

```

1055
1056 /*
1057 //ATANDA
1058 //MFR BEGIN
1059 //Using the notion that the distance of MFR advance and the advance
point to destination
is proportionate to current node position to next hop's, and next hop's to
destination
1060
1061 // #define PI = 3.14159265;
1062 double a, b, c;
1063
1064 //Calculate baseline values
1065 // Compute the  $aX + bY + C = 0$  line equation formed by (x0, y0) and (x1,
y1).
Template obtained from obstacles.cc in EstiNet
1066 if (position.y == nodePos.y)
1067 {
1068 a = 0;
1069 b = 1;
1070 c = -1 * nodePos.y;
1071 if (position.x == nodePos.x)
1072 a = b = c = 0;
1073 }
1074 else if (position.x == nodePos.x)
1075 {
1076 a = 1;
1077 b = 0;
1078 c = -1 * nodePos.x;
1079 }
1080 else
1081 {
1082 b = -1;
1083 a = (position.y - nodePos.y) / (position.x - nodePos.x);
1084 c = nodePos.y - nodePos.x * a;
1085 } //END Calculate baseline values
1086
1087 double slopeDistance = CalculateDistance (m_table.begin ()-
>second.first, nodePos);
1088 double perpendDistance = fabs (((a * m_table.begin ()->second.first.x) +
(b *
m_table.begin ()->second.first.y) + c) / sqrt((a * a) + (b * b)));
1089 double mfrDistance = sqrt((slopeDistance * slopeDistance) -
(perpendDistance *
perpendDistance));
1090 double mfrBestDistance = mfrDistance;
1091
1092 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
1093 for (i = m_table.begin (); !(i == m_table.end ()); i++)
1094 {
1095 //Again calculate baseline values
1096 // Again compute the  $aX + bY + C = 0$  line equation formed by (x0, y0)
and (x1, y1).
Template obtained from obstacles.cc in EstiNet
1097 if (position.y == nodePos.y)
1098 {
1099 a = 0;
1100 b = 1;

```

```

1101 c = -1 * nodePos.y;
1102 if (position.x == nodePos.x)
1103 a = b = c = 0;
1104 }
1105 else if (position.x == nodePos.x)
1106 {
1107 a = 1;
1108 b = 0;
1109 c = -1 * nodePos.x;
1110 }
1111 else
1112 {
1113 b = -1;
1114 a = (position.y - nodePos.y) / (position.x - nodePos.x);
1115 c = nodePos.y - nodePos.x * a;
1116 } //Again End Calculate baseline values
1117
1118 slopeDistance = CalculateDistance (i->second.first, nodePos);
1119 perpendDistance = fabs (((a * i->second.first.x) + (b * i-
>second.first.y) + c)
/ sqrt((a * a) + (b * b)));
1120 mfrDistance = sqrt((slopeDistance * slopeDistance) - (perpendDistance *
perpendDistance));
1121
1122 if ((CalculateDistance (i->second.first, position) < initialDistance) &&
(mfrDistance > mfrBestDistance))
1123
1124 {
1125 bestFoundID = i->first;
1126 //std::cout << "The value of bestFoundID is " << bestFoundID <<
std::endl; //atanda
1127 mfrBestDistance = mfrDistance;
1128 //std::cout << "The value of bestFoundDistance is " << bestFoundDistance
<<
std::endl; //atanda
1129 //std::cout << "The value of x is " << slopeDistance << std::endl;
//atanda
1130 //std::cout << "The value of y is " << perpendDistance << std::endl;
//atanda
1131 //std::cout << "The value of z is " << mfrDistance << std::endl;
//atanda
1132 bestFoundDistance = CalculateDistance (i->second.first, position);
1133 progressGreedyDistance = mfrBestDistance; //CalculateDistance (i-
>second.first,
nodePos);
1134 progressHomingDistance = bestFoundDistance;
1135 ProgressDistanceMeasures();
1136 }
1137 }
1138 //MFR END
1139 */
1140
1141 /*
1142 //ATANDA
1143 //NFP BEGIN
1144 // #define PI = 3.14159265;
1145 double a, b, c;
1146

```

```

1147 //Calculate baseline values
1148 // Compute the  $aX + bY + C = 0$  line equation formed by (x0, y0) and (x1,
y1).
Template obtained from obstacles.cc in EstiNet
1149 if (position.y == nodePos.y)
1150 {
1151 a = 0;
1152 b = 1;
1153 c = -1 * nodePos.y;
1154 if (position.x == nodePos.x)
1155 a = b = c = 0;
1156 }
1157 else if (position.x == nodePos.x)
1158 {
1159 a = 1;
1160 b = 0;
1161 c = -1 * nodePos.x;
1162 }
1163 else
1164 {
1165 b = -1;
1166 a = (position.y - nodePos.y) / (position.x - nodePos.x);
1167 c = nodePos.y - nodePos.x * a;
1168 } //END Calculate baseline values
1169
1170 double slopeDistance = CalculateDistance (m_table.begin ()-
>second.first, nodePos);
1171 double perpendDistance = fabs ((a * m_table.begin ()->second.first.x) +
(b *
m_table.begin ()->second.first.y) + c) / sqrt((a * a) + (b * b));
1172 double nfpDistance = sqrt((slopeDistance * slopeDistance) -
(perpendDistance *
perpendDistance));
1173 double nfpBestDistance = nfpDistance;
1174
1175 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
1176 for (i = m_table.begin (); !(i == m_table.end ()); i++)
1177 {
1178 //Again calculate baseline values
1179 // Again compute the  $aX + bY + C = 0$  line equation formed by (x0, y0)
and (x1, y1).
Template obtained from obstacles.cc in EstiNet
1180 if (position.y == nodePos.y)
1181 {
1182 a = 0;
1183 b = 1;
1184 c = -1 * nodePos.y;
1185 if (position.x == nodePos.x)
1186 a = b = c = 0;
1187 }
1188 else if (position.x == nodePos.x)
1189 {
1190 a = 1;
1191 b = 0;
1192 c = -1 * nodePos.x;
1193 }
1194 else
1195 {

```

```

1196 b = -1;
1197 a = (position.y - nodePos.y) / (position.x - nodePos.x);
1198 c = nodePos.y - nodePos.x * a;
1199 } //Again End Calculate baseline values
1200
1201 slopeDistance = CalculateDistance (i->second.first, nodePos);
1202 perpendDistance = fabs ((a * i->second.first.x) + (b * i-
>second.first.y) + c)
/ sqrt((a * a) + (b * b));
1203 nfpDistance = sqrt((slopeDistance * slopeDistance) - (perpendDistance *
perpendDistance));
1204
1205 if ((CalculateDistance (i->second.first, position) < initialDistance) &&
(nfpDistance < nfpBestDistance))
1206
1207 {
1208 bestFoundID = i->first;
1209 std::cout << "The value of bestFoundID is " << bestFoundID << std::endl;
//atanda
1210 nfpBestDistance = nfpDistance;
1211 //std::cout << "The value of bestFoundDistance is " << bestFoundDistance
<<
std::endl; //atanda
1212 std::cout << "The value of x is " << slopeDistance << std::endl;
//atanda
1213 std::cout << "The value of y is " << perpendDistance << std::endl;
//atanda
1214 std::cout << "The value of z is " << nfpDistance << std::endl; //atanda
1215 bestFoundDistance = CalculateDistance (i->second.first, position);
1216 progressGreedyDistance = nfpBestDistance; //CalculateDistance (i-
>second.first,
nodePos);
1217 progressHomingDistance = bestFoundDistance;
1218 ProgressDistanceMeasures();
1219 }
1220 }
1221 //NFP END
1222 */
1223
1224
1225 /*
1226 //ATANDA
1227 //COMPASS BEGIN - atanda
1228
1229 // #define PI = 3.14159265;
1230 double a, b, c;
1231
1232 //Calculate baseline values
1233 // Compute the  $aX + bY + C = 0$  line equation formed by (x0, y0) and (x1,
y1).
Template obtained from obstacles.cc in EstiNet
1234 if (position.y == nodePos.y)
1235 {
1236 a = 0;
1237 b = 1;
1238 c = -1 * nodePos.y;
1239 if (position.x == nodePos.x)
1240 a = b = c = 0;

```

```

1241 }
1242 else if (position.x == nodePos.x)
1243 {
1244 a = 1;
1245 b = 0;
1246 c = -1 * nodePos.x;
1247 }
1248 else
1249 {
1250 b = -1;
1251 a = (position.y - nodePos.y) / (position.x - nodePos.x);
1252 c = nodePos.y - nodePos.x * a;
1253 } //END Calculate baseline values
1254
1255
1256
1257 double slopeDistance = CalculateDistance (m_table.begin ()-
>second.first, nodePos);
1258 double perpendDistance = fabs (((a * m_table.begin ()->second.first.x) +
(b *
m_table.begin ()->second.first.y) + c) / sqrt((a * a) + (b * b)));
1259 double compassBestAcuteAngle = (asin(perpendDistance /
slopeDistance)*180) / M_PI;
1260
1261 //std::cout << "The value of initial compassAcuteAngle is " <<
compassBestAcuteAngle
<< std::endl; //atanda
1262 //std::cout << "The value of compass angle is " << compassBestAcuteAngle
<<
std::endl; //atanda
1263
1264 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
1265 for (i = m_table.begin (); !(i == m_table.end ()); i++)
1266 {
1267
1268 slopeDistance = CalculateDistance (i->second.first, nodePos);
1269 perpendDistance = fabs (((a * i->second.first.x) + (b * i-
>second.first.y) + c) /
sqrt((a * a) + (b * b)));
1270 //std::cout << "Slope distance is " << slopeDistance << std::endl;
//atanda
1271 //std::cout << "Perpendicular distance is " << perpendDistance <<
std::endl;
//atanda
1272 //std::cout << "Acute angle is " << (asin(perpendDistance /
slopeDistance)*180) /
M_PI << std::endl; //atanda
1273 //std::cout << "The value of i->first is " << i->first << std::endl;
//atanda
1274 //std::cout << "The value of i->first Distance is " << CalculateDistance
(i->second.first, position) << std::endl; //atanda
1275 //std::cout << "The value of i->first compassAcuteAngle is " <<
((asin(perpendDistance / slopeDistance)*180) / M_PI) << std::endl; //atanda
1276
1277
1278 if ((CalculateDistance (i->second.first, position) < initialDistance) &&
(((asin(perpendDistance / slopeDistance)*180) / M_PI) <
compassBestAcuteAngle))

```

```

1279
1280 {
1281 bestFoundID = i->first;
1282 //std::cout << "The value of bestFoundID is " << bestFoundID <<
std::endl; //atanda
1283 bestFoundDistance = CalculateDistance (i->second.first, position);
1284 compassBestAcuteAngle = ((asin(perpendDistance / slopeDistance)*180) /
M_PI);
1285 //std::cout << "The value of bestFoundDistance is " << bestFoundDistance
<<
std::endl; //atanda
1286 //std::cout << "The value of compassBestAcuteAngle is " <<
compassBestAcuteAngle << std::endl; //atanda
1287 }
1288 }
1289 //COMPASS END
1290 */
1291
1292 /*
1293 //ATANDA
1294 //CD-AD approach BEGIN //Alternative GREEDY algorithm
1295 //In CD-AD (CD minus AD distances) approach, Node C is the current, A is
potential next
hop and D is the destination
1296
1297
1298 double cdadBestDistance = (CalculateDistance (nodePos, position) -
CalculateDistance
(m_table.begin ()->second.first, position));
1299
1300 //std::cout << "The value of initial cdadDistance is " <<
cdadBestDistance <<
std::endl; //atanda
1301
1302
1303 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
1304 for (i = m_table.begin (); !(i == m_table.end ()); i++)
1305 {
1306 nodeDegreeCount += 1; //atanda
1307 //std::cout << "The value of i->first is " << i->first << std::endl;
//atanda
1308 //std::cout << "The value of i->first Distance is " << CalculateDistance
(i->second.first, position) << std::endl; //atanda
1309 //std::cout << "The value of i->first cdadDistance is " <<
(CalculateDistance
(nodePos, position) - CalculateDistance (i->second.first, position)) <<
std::endl; //atanda
1310
1311 if ((CalculateDistance (i->second.first, position) < initialDistance) &&
((CalculateDistance (nodePos, position) - CalculateDistance (i->second.first,
position)) > cdadBestDistance))
1312
1313 {
1314 bestFoundID = i->first;
1315 //std::cout << "The value of bestFoundID is " << bestFoundID <<
std::endl;
//atanda
1316 bestFoundDistance = CalculateDistance (i->second.first, position);

```

```

1317 cdadBestDistance = (CalculateDistance (nodePos, position) -
CalculateDistance
(i->second.first, position));
1318 //std::cout << "The value of bestFoundDistance is " << bestFoundDistance
<<
std::endl; //atanda
1319 //std::cout << "The value of cdadBestDistance is " << cdadBestDistance
<<
std::endl; //atanda
1320 }
1321 }
1322 //CD-AD approach END
1323 */
1324 //ATANDA
1325 //std::cout << "Node Degree is " << nodeDegree << std::endl;
1326 //std::cout << "progressNodeDegree2 is " << progressNodeDegree <<
std::endl;
1327 //std::cout << "BestfoundID is " << bestFoundID << std::endl; //atanda
1328 for (uint32_t i = 0; i < NODE_DEGREES_ARRAY_SIZE; ++i)
1329 {
1330 if (i == nodeDegree)
1331 {
1332 //std::cout << "noDegree is " << nodeDegree << std::endl;
1333 nodeDegreeCount[i] += 1;
1334 if(initialDistance > bestFoundDistance)
1335 progressNodeDegreeCount[i] += 1;
1336 //std::cout << "i is: " << i << " nodeDegreeCount[" << i << "] is " <<
nodeDegreeCount[i] << std::endl;
1337 break;
1338 }
1339 }
1340
1341 //ATANDA
1342 if (nodeDegree > highestNodeDegree)
1343 {
1344 highestNodeDegree = nodeDegree;
1345 }
1346
1347
1348 //std::cout << "The node degree is " << nodeDegree << std::endl;
//atanda
1349 //std::cout << "Local progress region node degree " <<
progressNodeDegree <<
std::endl;
1350
1351 //std::cout << "Total Node Degrees " << totalDegreeCount << std::endl;
1352 //std::cout << "Progress Regions Total Node Degrees " <<
totalProgressRegionDegreeCount << std::endl;
1353
1354
1355 if(initialDistance > bestFoundDistance)
1356 {
1357 //ATANDA begin
1358 //std::cout << "Packet sent from " << bestFoundID << std::endl; //atanda
1359 //std::cout << "gpsrFlowId: " << gpsrFlowId << " greedyChoice: " <<
greedyChoice
<< std::endl; //atanda

```

```

1360 //ATANDA begin - this has to be hear rathan than enclosed in a greedy
method's
neighbours checking iterations
1361
1362
1363 forwarderGreedyDistance = tempForwarderGreedyDistance;//atanda
1364 forwarderHomingDistance = tempForwarderHomingDistance;//atanda
1365 ForwarderDistanceMeasures2();
1366 //std::cout << "forwarderGreedyDistance: " << forwarderGreedyDistance <<
std::endl; //atanda
1367
1368 //if (forwarderGreedyDistance <= 10)
1369 //std::cout << "forwarderGreedyDistance1 :" << forwarderGreedyDistance
<<
std::endl; //atanda
1370
1371 totalForwarders += 1; //atanda
1372 //ATANDA end
1373
1374 return bestFoundID;
1375 }
1376 else
1377 {
1378
1379 totalFailForwarders += 1; //atanda, 2nd point of incrementing
1380 totalIntoPerimeterRecovery += 1; //atanda
1381 //std::cout << "Entering recovery mode" << std::endl; //atanda
1382 return Ipv4Address::GetZero (); //so it enters Recovery-mode
1383 }
1384
1385 gpsrRxPackets = gpsrrxpackets->GpsrRxPackets (); //1 packet received, if
any, for
each call
1386 } //END - BESTNEIGHBOUR
1387
1388
1389
1390 /*
1391 * \brief Gets next hop according to GPSR recovery-mode protocol (right
hand rule)
1392 * \param previousHop the position of the node that sent the packet to
this node
1393 * \param nodePos the position of the destination node
1394 * \return Ipv4Address of the next hop, Ipv4Address::GetZero () if no
neighbour was found
in greedy mode
1395 */
1396 Ipv4Address
1397 PositionTable::BestAngle (Vector previousHop, Vector nodePos)
1398 {
1399 Purge ();
1400
1401 if (m_table.empty ())
1402 {
1403 NS_LOG_DEBUG ("BestNeighbor table is empty; Position: " << nodePos);
1404 return Ipv4Address::GetZero ();
1405 } //if table is empty (no neighbours)
1406

```

```

1407 double tmpAngle;
1408 Ipv4Address bestFoundID = Ipv4Address::GetZero ();
1409 double bestFoundAngle = 360;
1410 std::map<Ipv4Address, std::pair<Vector, Time> >::iterator i;
1411
1412 for (i = m_table.begin (); !(i == m_table.end ()); i++)
1413 {
1414     tmpAngle = GetAngle(nodePos, previousHop, i->second.first);
1415
1416     if (bestFoundAngle > tmpAngle && tmpAngle != 0)
1417     {
1418         bestFoundID = i->first;
1419         bestFoundAngle = tmpAngle;
1420     }
1421 }
1422 }
1423
1424 if(bestFoundID == Ipv4Address::GetZero ()) //only if the only neighbour
is who sent
the packet
1425 {
1426     bestFoundID = m_table.begin ()->first;
1427 }
1428 return bestFoundID;
1429 }
1430
1431
1432 //Gives angle between the vector CentrePos-Refpos to the vector
CentrePos-node
counterclockwise
1433 double
1434 PositionTable::GetAngle (Vector centrePos, Vector refPos, Vector node){
1435     double const PI = 4*atan(1);
1436
1437     std::complex<double> A = std::complex<double>(centrePos.x,centrePos.y);
1438     std::complex<double> B = std::complex<double>(node.x,node.y);
1439     std::complex<double> C = std::complex<double>(refPos.x,refPos.y);
1440     //Change B with C
1441     if you want angles clockwise
1442
1443     std::complex<double> AB; //reference edge
1444     std::complex<double> AC;
1445     std::complex<double> tmp;
1446     std::complex<double> tmpCplx;
1447
1448     std::complex<double> Angle;
1449
1450     AB = B - A;
1451     AB = (real(AB)/norm(AB)) +
(std::complex<double>(0.0,1.0)*(imag(AB)/norm(AB)));
1452
1453     AC = C - A;
1454     AC = (real(AC)/norm(AC)) +
(std::complex<double>(0.0,1.0)*(imag(AC)/norm(AC)));
1455
1456     tmp = log(AC/AB);
1457     tmpCplx = std::complex<double>(0.0,-1.0);
1458     Angle = tmp*tmpCplx;

```

```

1457 Angle *= (180/PI);
1458 if (real(Angle)<0)
1459 Angle = 360+real(Angle);
1460
1461 return real(Angle);
1462
1463 }
1464
1465
1466
1467 /**
1468 * \ProcessTxError
1469 */
1470 void PositionTable::ProcessTxError (WifiMacHeader const & hdr)
1471 {
1472 }
1473
1474
1475
1476 //FIXME ainda preciso disto agr que o LS ja n está aqui???????
1477
1478 /**
1479 * \brief Returns true if is in search for destination
1480 */
1481 bool PositionTable::IsInSearch (Ipv4Address id)
1482 {
1483 return false;
1484 }
1485
1486 bool PositionTable::HasPosition (Ipv4Address id)
1487 {
1488 return true;
1489 }
1490
1491
1492 } // gpsr
1493 } // ns3
1494

```

ns3-gpsr-ptable-cpph

```

1 #ifndef GPSR_PTABLE_H
2 #define GPSR_PTABLE_H
3
4 #include <map>
5 #include <cassert>
6 #include <stdint.h>
7 #include "ns3/ipv4.h"
8 #include "ns3/timer.h"
9 #include <sys/types.h>
10 #include "ns3/node.h"
11 #include "ns3/node-list.h"
12 #include "ns3/mobility-model.h"
13 #include "ns3/vector.h"
14 #include "ns3/wifi-mac-header.h"
15 #include "ns3/random-variable-stream.h"
16 #include <complex>
17
18
19
20 namespace ns3 {
21 namespace gpsr {
22
23 //ATANDA
24 /*
25 struct DataPacketsAverages {
26 uint32_t first, second;
27 };
28 */
29
30 //DataPacketsAverages packetsAverages; //atanda
31
32 /*
33 * \ingroup gpsr
34 * \brief Position table used by GPSR
35 */
36 class PositionTable
37 {
38 public:
39
40
41 //ATANDA
42 void TotalNodeDegreeCount ();
43 void TotalForwardersDegreeCount ();
44
45
46 //ATANDA
47 void BestNeighbourOutputs ();
48 void CsvNodeDegreesWrite ();
49 void CsvDistancesWrite2 ();
50 void CsvDistancesAveragesWrite3 ();
51 void ProgressDistanceMeasures ();
52 void NonProgressDistanceMeasures ();
53 void ForwarderDistanceMeasures2 ();
54 double GetSimTotalTime (double simTotalTimeA);
55 double GetSimPeriod (double simPeriodA);
56 uint32_t GetSimSize (uint32_t simSizeA);
57 uint32_t ObtainModFactor ();
58

```

```

59 /// c-tor
60 PositionTable ();
61
62
63 /**
64 * \brief Gets the last time the entry was updated
65 * \param id Ipv4Address to get time of update from
66 * \return Time of last update to the position
67 */
68 Time GetEntryUpdateTime (Ipv4Address id);
69
70 /**
71 * \brief Adds entry in position table
72 */
73 void AddEntry (Ipv4Address id, Vector position);
74
75 /**
76 * \brief Deletes entry in position table
77 */
78 void DeleteEntry (Ipv4Address id);
79
80 /**
81 * \brief Gets position from position table
82 * \param id Ipv4Address to get position from
83 * \return Position of that id or NULL if not known
84 */
85 Vector GetPosition (Ipv4Address id);
86
87 /**
88 * \brief Checks if a node is a neighbour
89 * \param id Ipv4Address of the node to check
90 * \return True if the node is neighbour, false otherwise
91 */
92 bool isNeighbour (Ipv4Address id);
93
94 /**
95 * \brief remove entries with expired lifetime
96 */
97 void Purge ();
98
99 /**
100 * \brief clears all entries
101 */
102 void Clear ();
103
104 /**
105 * \Get Callback to ProcessTxError
106 */
107 Callback<void, WifiMacHeader const &> GetTxErrorCallback () const
108 {
109     return m_txErrorCallback;
110 }
111
112 /**
113 * \brief Gets next hop according to GPSR protocol
114 * \param position the position of the destination node
115 * \param nodePos the position of the node that has the packet

```

```

116 * \return Ipv4Address of the next hop, Ipv4Address::GetZero () if no
nighbour was
found in greedy mode
117 */
118
119
120
121 Ipv4Address BestNeighbor (Vector position, Vector nodePos);
122
123 bool IsInSearch (Ipv4Address id);
124
125 bool HasPosition (Ipv4Address id);
126
127 static Vector GetInvalidPosition ()
128 {
129     return Vector (-1, -1, 0);
130 }
131
132 uint64_t GetSize(){
133     return m_table.size();
134 }
135
136 /**
137 * \brief Gets next hop according to GPSR recovery-mode protocol (right
hand rule)
138 * \param previousHop the position of the node that sent the packet to
this node
139 * \param nodePos the position of the destination node
140 * \return Ipv4Address of the next hop, Ipv4Address::GetZero () if no
nighbour was
found in greedy mode
141 */
142 Ipv4Address BestAngle (Vector previousHop, Vector nodePos);
143
144 //Gives angle between the vector CentrePos-Refpos to the vector
CentrePos-node
counterclockwise
145 double GetAngle (Vector centrePos, Vector refPos, Vector node);
146
147
148
149 private:
150 Time m_entryLifeTime;
151 std::map<Ipv4Address, std::pair<Vector, Time> > m_table;
152 // TX error callback
153 Callback<void, WifiMacHeader const &> m_txErrorCallback;
154 // Process layer 2 TX error notification
155 void ProcessTxError (WifiMacHeader const&);
156
157
158
159
160 };
161
162 } // gpsr
163 } // ns3
164 #endif /* GPSR_PTABLE_H */
165

```

ns3-ipv4-flow-probe-cpp

```

1  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2  //
3  // Copyright (c) 2009 INESC Porto
4  //
5  // This program is free software; you can redistribute it and/or modify
6  // it under the terms of the GNU General Public License version 2 as
7  // published by the Free Software Foundation;
8  //
9  // This program is distributed in the hope that it will be useful,
10 // but WITHOUT ANY WARRANTY; without even the implied warranty of
11 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 // GNU General Public License for more details.
13 //
14 // You should have received a copy of the GNU General Public License
15 // along with this program; if not, write to the Free Software
16 // Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
17 //
18 // Author: Gustavo J. A. M. Carneiro <gjc@inescporto.pt>
19 // <gjcarneiro@gmail.com>
20 //
21 #include "ns3/ipv4-flow-probe.h"
22 #include "ns3/ipv4-flow-classifier.h"
23 #include "ns3/node.h"
24 #include "ns3/packet.h"
25 #include "ns3/flow-monitor.h"
26 #include "ns3/log.h"
27 #include "ns3/pointer.h"
28 #include "ns3/config.h"
29 #include "ns3/flow-id-tag.h"
30
31 namespace ns3 {
32
33 NS_LOG_COMPONENT_DEFINE ("Ipv4FlowProbe");
34
35 //////////////////////////////////////////////////
36 // Ipv4FlowProbeTag class implementation //
37 //////////////////////////////////////////////////
38
39 /**
40 * \ingroup flow-monitor
41 *
42 * \brief Tag used to allow a fast identification of the packet
43 *
44 * This tag is added by FlowMonitor when a packet is seen for
45 * the first time, and it is then used to classify the packet in
46 * the following hops.
47 */
48 class Ipv4FlowProbeTag : public Tag
49 {
50 public:
51 /**
52 * \brief Get the type ID.
53 * \return the object TypeId
54 */

```

```

55 static TypeId GetTypeId (void);
56 virtual TypeId GetInstanceTypeId (void) const;
57 virtual uint32_t GetSerializedSize (void) const;
58 virtual void Serialize (TagBuffer buf) const;
59 virtual void Deserialize (TagBuffer buf);
60 virtual void Print (std::ostream &os) const;
61 Ipv4FlowProbeTag ();
62 /**
63  * \brief Constructor
64  * \param flowId the flow identifier
65  * \param packetId the packet identifier
66  * \param packetSize the packet size
67 */
68 Ipv4FlowProbeTag (uint32_t flowId, uint32_t packetId, uint32_t
packetSize);
69 /**
70  * \brief Set the flow identifier
71  * \param flowId the flow identifier
72 */
73 void SetFlowId (uint32_t flowId);
74 /**
75  * \brief Set the packet identifier
76  * \param packetId the packet identifier
77 */
78 void SetPacketId (uint32_t packetId);
79 /**
80  * \brief Set the packet size
81  * \param packetSize the packet size
82 */
83 void SetPacketSize (uint32_t packetSize);
84 /**
85  * \brief Set the flow identifier
86  * \returns the flow identifier
87 */
88 uint32_t GetFlowId (void) const;
89 /**
90  * \brief Set the packet identifier
91  * \returns the packet identifier
92 */
93 uint32_t GetPacketId (void) const;
94 /**
95  * \brief Get the packet size
96  * \returns the packet size
97 */
98 uint32_t GetPacketSize (void) const;
99 private:
100 uint32_t m_flowId; //!< flow identifier
101 uint32_t m_packetId; //!< packet identifier
102 uint32_t m_packetSize; //!< packet size
103
104 };
105
106 TypeId
107 Ipv4FlowProbeTag::GetTypeId (void)
108 {
109 static TypeId tid = TypeId ("ns3::Ipv4FlowProbeTag")
110 .SetParent<Tag> ()
111 .SetGroupName ("FlowMonitor")

```

```

112 .AddConstructor<Ipv4FlowProbeTag> ()
113 ;
114 return tid;
115 }
116 TypeId
117 Ipv4FlowProbeTag::GetInstanceTypeId (void) const
118 {
119     return GetTypeId ();
120 }
121 uint32_t
122 Ipv4FlowProbeTag::GetSerializedSize (void) const
123 {
124     return 4 + 4 + 4;
125 }
126 void
127 Ipv4FlowProbeTag::Serialize (TagBuffer buf) const
128 {
129     buf.WriteU32 (m_flowId);
130     buf.WriteU32 (m_packetId);
131     buf.WriteU32 (m_packetSize);
132 }
133 void
134 Ipv4FlowProbeTag::Deserialize (TagBuffer buf)
135 {
136     m_flowId = buf.ReadU32 ();
137     m_packetId = buf.ReadU32 ();
138     m_packetSize = buf.ReadU32 ();
139 }
140 void
141 Ipv4FlowProbeTag::Print (std::ostream &os) const
142 {
143     os << "FlowId=" << m_flowId;
144     os << "PacketId=" << m_packetId;
145     os << "PacketSize=" << m_packetSize;
146 }
147 Ipv4FlowProbeTag::Ipv4FlowProbeTag ()
148 : Tag ()
149 {
150 }
151
152 Ipv4FlowProbeTag::Ipv4FlowProbeTag (uint32_t flowId, uint32_t packetId,
uint32_t
packetSize)
153 : Tag (), m_flowId (flowId), m_packetId (packetId), m_packetSize
(packetSize)
154 {
155 }
156
157 void
158 Ipv4FlowProbeTag::SetFlowId (uint32_t id)
159 {
160     m_flowId = id;
161 }
162 void
163 Ipv4FlowProbeTag::SetPacketId (uint32_t id)
164 {
165     m_packetId = id;
166 }

```

```

167 void
168 Ipv4FlowProbeTag::SetPacketSize (uint32_t size)
169 {
170     m_packetSize = size;
171 }
172 uint32_t
173 Ipv4FlowProbeTag::GetFlowId (void) const
174 {
175     return m_flowId;
176 }
177 uint32_t
178 Ipv4FlowProbeTag::GetPacketId (void) const
179 {
180     return m_packetId;
181 }
182 uint32_t
183 Ipv4FlowProbeTag::GetPacketSize (void) const
184 {
185     return m_packetSize;
186 }
187
188 //////////////////////////////////////
189 // Ipv4FlowProbe class implementation //
190 //////////////////////////////////////
191
192 Ipv4FlowProbe::Ipv4FlowProbe (Ptr<FlowMonitor> monitor,
193 Ptr<Ipv4FlowClassifier> classifier,
194 Ptr<Node> node)
195 : FlowProbe (monitor),
196 m_classifier (classifier)
197 {
198     NS_LOG_FUNCTION (this << node->GetId ());
199
200     m_ipv4 = node->GetObject<Ipv4L3Protocol> ();
201
202     if (!m_ipv4->TraceConnectWithoutContext ("SendOutgoing",
203 MakeCallback (&Ipv4FlowProbe::
204 SendOutgoingLogger, Ptr<Ipv4FlowProbe> (this
205 )))
206 {
207     NS_FATAL_ERROR ("trace fail");
208 }
209
210     if (!m_ipv4->TraceConnectWithoutContext ("UnicastForward",
211 MakeCallback (&Ipv4FlowProbe::ForwardLogger,
212 Ptr<Ipv4FlowProbe> (this)))
213 )
214 {
215     NS_FATAL_ERROR ("trace fail");
216 }
217
218     if (!m_ipv4->TraceConnectWithoutContext ("Drop",
219 MakeCallback (&Ipv4FlowProbe::DropLogger, Ptr<
220 Ipv4FlowProbe> (this)))

```

```

220 {
221 NS_FATAL_ERROR ("trace fail");
222 }
223
224 // code copied from point-to-point-helper.cc
225 std::ostringstream oss;
226 oss << "/NodeList/" << node->GetId () << "/DeviceList/*/TxQueue/Drop";
227 Config::ConnectWithoutContext (oss.str (), MakeCallback (&Ipv4FlowProbe::
QueueDropLogger, Ptr<Ipv4FlowProbe> (this)));
228 }
229
230 Ipv4FlowProbe::~Ipv4FlowProbe ()
231 {
232 }
233
234 void
235 Ipv4FlowProbe::DoDispose ()
236 {
237 m_ipv4 = 0;
238 m_classifier = 0;
239 FlowProbe::DoDispose ();
240 }
241
//ATANDA-----
-----
242 //ATANDA -begin
243 FlowId gpsrFlowId;
244 FlowId
245 Ipv4FlowProbe::GpsrFlowId ()
246 {
247 return gpsrFlowId; //Returns current flowId to gpsr-ptable.cc
248 }
249 //ATANDA end
250
//ATANDA-----
-----
251 //ATANDA -begin
252 uint32_t gpsrRxPackets = 0;
253 uint32_t
254 Ipv4FlowProbe::GpsrRxPackets ()
255 {
256 return gpsrRxPackets; //Retruns cumulative value of total packets
received so far to
gpsr-ptable.cc
257 }
258 //ATANDA end
259
260
//ATANDA-----
-----
261 void
262 Ipv4FlowProbe::SendOutgoingLogger (const Ipv4Header &ipHeader, Ptr<const
Packet>
ipPayload, uint32_t interface)
263 {
264 FlowId flowId;
265 FlowPacketId packetId;
266

```

```

267
268
269 if (!m_ipv4->IsUnicast(ipHeader.GetDestination ()))
270 {
271 // we are not prepared to handle broadcast yet
272 return;
273 }
274
275 if (m_classifier->Classify (ipHeader, ipPayload, &flowId, &packetId))
276 {
277 uint32_t size = (ipPayload->GetSize () + ipHeader.GetSerializedSize ());
278 NS_LOG_DEBUG ("ReportFirstTx ("<<this<<", "<<flowId<<", "<<packetId<<",
"<<size<<
"); "
279 << ipHeader << *ipPayload);
280 m_flowMonitor->ReportFirstTx (this, flowId, packetId, size);
281
282 // tag the packet with the flow id and packet id, so that the packet can
be
identified even
283 // when Ipv4Header is not accessible at some non-IPv4 protocol layer
284 Ipv4FlowProbeTag fTag (flowId, packetId, size);
285 ipPayload->AddByteTag (fTag);
286
287 //ATANDA - gpsrFlowId assignment - begin
288 gpsrFlowId = flowId;
289 GpsrFlowId ();
290 //std::cout << "The value of flowId is " << flowId << std::endl; //atanda
291 //ATANDA - gpsrFlowId assignment - end
292
293 }
294 }
295
296 void
297 Ipv4FlowProbe::ForwardLogger (const Ipv4Header &ipHeader, Ptr<const
Packet> ipPayload,
uint32_t interface)
298 {
299 Ipv4FlowProbeTag fTag;
300 bool found = ipPayload->FindFirstMatchingByteTag (fTag);
301
302 if (found)
303 {
304 FlowId flowId = fTag.GetFlowId ();
305 FlowPacketId packetId = fTag.GetPacketId ();
306
307 uint32_t size = (ipPayload->GetSize () + ipHeader.GetSerializedSize ());
308 NS_LOG_DEBUG ("ReportForwarding ("<<this<<", "<<flowId<<",
"<<packetId<<", "<<size
<<"); "
309 m_flowMonitor->ReportForwarding (this, flowId, packetId, size);
310
311 }
312 }
313
314 void
315 Ipv4FlowProbe::ForwardUpLogger (const Ipv4Header &ipHeader, Ptr<const
Packet> ipPayload,

```

```

uint32_t interface)
316 {
317 Ipv4FlowProbeTag fTag;
318 bool found = ipPayload->FindFirstMatchingByteTag (fTag);
319
320 if (found)
321 {
322 FlowId flowId = fTag.GetFlowId ();
323 FlowPacketId packetId = fTag.GetPacketId ();
324
325 uint32_t size = (ipPayload->GetSize () + ipHeader.GetSerializedSize ());
326 NS_LOG_DEBUG ("ReportLastRx ("<<this<<", "<<flowId<<", "<<packetId<<",
"<<size<<
");");
327 m_flowMonitor->ReportLastRx (this, flowId, packetId, size);
328
329 //ATANDA begin
330 ++gpsrRxPackets;
331 //std::cout << "Packets received according to ipv4-flow-probe " <<
gpsrRxPackets
<< std::endl; //atanda
332 //ATANDA end
333
334 }
335 }
336
337 void
338 Ipv4FlowProbe::DropLogger (const Ipv4Header &ipHeader, Ptr<const Packet>
ipPayload,
339 Ipv4L3Protocol::DropReason reason, Ptr<Ipv4> ipv4, uint32_t
ifIndex)
340 {
341 #if 0
342 switch (reason)
343 {
344 case Ipv4L3Protocol::DROP_NO_ROUTE:
345 break;
346
347 case Ipv4L3Protocol::DROP_TTL_EXPIRED:
348 case Ipv4L3Protocol::DROP_BAD_CHECKSUM:
349 Ipv4Address addri = m_ipv4->GetAddress (ifIndex);
350 Ipv4Mask maski = m_ipv4->GetNetworkMask (ifIndex);
351 Ipv4Address bcast = addri.GetSubnetDirectedBroadcast (maski);
352 if (ipHeader.GetDestination () == bcast) // we don't want broadcast
packets
353 {
354 return;
355 }
356 }
357 #endif
358
359 Ipv4FlowProbeTag fTag;
360 bool found = ipPayload->FindFirstMatchingByteTag (fTag);
361
362 if (found)
363 {
364 FlowId flowId = fTag.GetFlowId ();
365 FlowPacketId packetId = fTag.GetPacketId ();

```

```

366
367 uint32_t size = (ipPayload->GetSize () + ipHeader.GetSerializedSize ());
368 NS_LOG_DEBUG ("Drop ("<<this<< ", "<<flowId<< ", "<<packetId<< ",
"<<size<< ", " <<
reason
369 << ", destIp=" << ipHeader.GetDestination () << "); "
370 << "HDR: " << ipHeader << " PKT: " << *ipPayload);
371
372 DropReason myReason;
373
374
375 switch (reason)
376 {
377 case Ipv4L3Protocol::DROP_TTL_EXPIRED:
378 myReason = DROP_TTL_EXPIRE;
379 NS_LOG_DEBUG ("DROP_TTL_EXPIRE");
380 break;
381 case Ipv4L3Protocol::DROP_NO_ROUTE:
382 myReason = DROP_NO_ROUTE;
383 NS_LOG_DEBUG ("DROP_NO_ROUTE");
384 break;
385 case Ipv4L3Protocol::DROP_BAD_CHECKSUM:
386 myReason = DROP_BAD_CHECKSUM;
387 NS_LOG_DEBUG ("DROP_BAD_CHECKSUM");
388 break;
389 case Ipv4L3Protocol::DROP_INTERFACE_DOWN:
390 myReason = DROP_INTERFACE_DOWN;
391 NS_LOG_DEBUG ("DROP_INTERFACE_DOWN");
392 break;
393 case Ipv4L3Protocol::DROP_ROUTE_ERROR:
394 myReason = DROP_ROUTE_ERROR;
395 NS_LOG_DEBUG ("DROP_ROUTE_ERROR");
396 break;
397 case Ipv4L3Protocol::DROP_FRAGMENT_TIMEOUT:
398 myReason = DROP_FRAGMENT_TIMEOUT;
399 NS_LOG_DEBUG ("DROP_FRAGMENT_TIMEOUT");
400 break;
401
402 default:
403 myReason = DROP_INVALID_REASON;
404 NS_FATAL_ERROR ("Unexpected drop reason code " << reason);
405 }
406
407 m_flowMonitor->ReportDrop (this, flowId, packetId, size, myReason);
408 }
409 }
410
411 void
412 Ipv4FlowProbe::QueueDropLogger (Ptr<const Packet> ipPayload)
413 {
414 Ipv4FlowProbeTag fTag;
415 bool tagFound = ipPayload->FindFirstMatchingByteTag (fTag);
416
417 if (!tagFound)
418 {
419 return;
420 }
421

```

```

422 FlowId flowId = fTag.GetFlowId ();
423 FlowPacketId packetId = fTag.GetPacketId ();
424 uint32_t size = fTag.GetPacketSize ();
425
426 NS_LOG_DEBUG ("Drop ("<<this<<", "<<flowId<<", "<<packetId<<",
"<<size<<", " <<
DROP_QUEUE
427 << "); ");
428
429 m_flowMonitor->ReportDrop (this, flowId, packetId, size, DROP_QUEUE);
430 }
431
432 } // namespace ns3
433
434
435

```

ns3-ipv4-flow-probe-cpph

```

1  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2  //
3  // Copyright (c) 2009 INESC Porto
4  //
5  // This program is free software; you can redistribute it and/or modify
6  // it under the terms of the GNU General Public License version 2 as
7  // published by the Free Software Foundation;
8  //
9  // This program is distributed in the hope that it will be useful,
10 // but WITHOUT ANY WARRANTY; without even the implied warranty of
11 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 // GNU General Public License for more details.
13 //
14 // You should have received a copy of the GNU General Public License
15 // along with this program; if not, write to the Free Software
16 // Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
17 //
18 // Author: Gustavo J. A. M. Carneiro <gjc@inescporto.pt>
19 // <gjcarneiro@gmail.com>
20 //
21 #ifndef IPV4_FLOW_PROBE_H
22 #define IPV4_FLOW_PROBE_H
23
24 #include "ns3/flow-probe.h"
25 #include "ns3/ipv4-flow-classifier.h"
26 #include "ns3/ipv4-l3-protocol.h"
27
28 namespace ns3 {
29
30 class FlowMonitor;
31 class Node;
32
33
34 /// \ingroup flow-monitor
35 /// \brief Class that monitors flows at the IPv4 layer of a Node
36 ///
37 /// For each node in the simulation, one instance of the class
38 /// Ipv4FlowProbe is created to monitor that node. Ipv4FlowProbe
39 /// accomplishes this by connecting callbacks to trace sources in the
40 /// Ipv4L3Protocol interface of the node.
41 class Ipv4FlowProbe : public FlowProbe
42 {
43
44 public:
45 /// \brief Constructor
46 /// \param monitor the FlowMonitor this probe is associated with
47 /// \param classifier the Ipv4FlowClassifier this probe is associated with
48 /// \param node the Node this probe is associated with
49 Ipv4FlowProbe (Ptr<FlowMonitor> monitor, Ptr<Ipv4FlowClassifier>
classifier, Ptr<Node>
node);
50 virtual ~Ipv4FlowProbe ();
51
52 //ATANDA-----
53
54 //ATANDA GpsrFlowId () declaration - begin

```

```

55 FlowId GpsrFlowId ();
56 uint32_t GpsrRxPackets ();
57 //ATANDA GpsrFlowId () declaration - end
58
59 /// \brief enumeration of possible reasons why a packet may be dropped
60 enum DropReason
61 {
62     /// Packet dropped due to missing route to the destination
63     DROP_NO_ROUTE = 0,
64
65     /// Packet dropped due to TTL decremented to zero during IPv4 forwarding
66     DROP_TTL_EXPIRE,
67
68     /// Packet dropped due to invalid checksum in the IPv4 header
69     DROP_BAD_CHECKSUM,
70
71     /// Packet dropped due to queue overflow. Note: only works for
72     /// NetDevices that provide a TxQueue attribute of type Queue
73     /// with a Drop trace source. It currently works with Csma and
74     /// PointToPoint devices, but not with WiFi or WiMax.
75     DROP_QUEUE,
76
77     DROP_INTERFACE_DOWN, /**< Interface is down so can not send packet */
78     DROP_ROUTE_ERROR, /**< Route error */
79     DROP_FRAGMENT_TIMEOUT, /**< Fragment timeout exceeded */
80
81     DROP_INVALID_REASON, /**< Fallback reason (no known reason) */
82 };
83
84 protected:
85
86 virtual void DoDispose (void);
87
88 private:
89     /// Log a packet being sent
90     /// \param ipHeader IP header
91     /// \param ipPayload IP payload
92     /// \param interface outgoing interface
93     void SendOutgoingLogger (const Ipv4Header &ipHeader, Ptr<const Packet>
ipPayload,
uint32_t interface);
94     /// Log a packet being forwarded
95     /// \param ipHeader IP header
96     /// \param ipPayload IP payload
97     /// \param interface incoming interface
98     void ForwardLogger (const Ipv4Header &ipHeader, Ptr<const Packet>
ipPayload, uint32_t
interface);
99
100
101
102     /// Log a packet being received by the destination
103     /// \param ipHeader IP header
104     /// \param ipPayload IP payload
105     /// \param interface incoming interface
106     void ForwardUpLogger (const Ipv4Header &ipHeader, Ptr<const Packet>
ipPayload, uint32_t
interface);

```

```

107 /// Log a packet being dropped
108 /// \param ipHeader IP header
109 /// \param ipPayload IP payload
110 /// \param reason drop reason
111 /// \param ipv4 pointer to the IP object dropping the packet
112 /// \param ifIndex interface index
113 void DropLogger (const Ipv4Header &ipHeader, Ptr<const Packet> ipPayload,
114 Ipv4L3Protocol::DropReason reason, Ptr<Ipv4> ipv4, uint32_t ifIndex);
115 /// Log a packet being dropped by a queue
116 /// \param ipPayload IP payload
117 void QueueDropLogger (Ptr<const Packet> ipPayload);
118
119 Ptr<Ipv4FlowClassifier> m_classifier; //!< the Ipv4FlowClassifier this
probe is
associated with
120 Ptr<Ipv4L3Protocol> m_ipv4; //!< the Ipv4L3Protocol this probe is bound
to
121 };
122
123
124 } // namespace ns3
125
126 #endif /* IPV4_FLOW_PROBE_H */
127

```

ns3-pagpsr-main-cpp

```

1  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2
3  #include "ns3/gpsr-module.h"
4  #include "ns3/pagpsr-module.h"
5  #include "ns3/mmgpsr-module.h"
6  #include "ns3/core-module.h"
7  #include "ns3/network-module.h"
8  #include "ns3/internet-module.h"
9  #include "ns3/mobility-module.h"
10 #include "ns3/point-to-point-module.h"
11 #include "ns3/wifi-module.h"
12 #include "ns3/v4ping-helper.h"
13 #include "ns3/udp-echo-server.h"
14 #include "ns3/udp-echo-client.h"
15 #include "ns3/udp-echo-helper.h"
16 #include <iostream>
17 #include <cmath>
18 #include "ns3/flow-monitor-module.h"
19 #include "ns3/netanim-module.h"
20 #include "string.h"
21 #include "ns3/ocb-wifi-mac.h"
22 #include "ns3/wifi-80211p-helper.h"
23 #include "ns3/wave-mac-helper.h"
24 #include "ns3/applications-module.h"
25 #include <string>
26 #include <sys/stat.h>
27 #include <iomanip>
28 #include <fstream>
29
30 #include "ns3/wave-net-device.h"
31 #include "ns3/wave-mac-helper.h"
32 #include "ns3/wave-helper.h"
33
34
35
36
37 using namespace ns3;
38
39 //ATANDA
40 double simPeriodA = 0; //atanda
41 uint32_t simSizeA = 0; //atanda
42 uint32_t modFactorA = 0; //atanda
43 //atanda
44
45 uint32_t totalHello = 0;
46 uint32_t rx_drop = 0;
47 uint32_t tx_drop = 0;
48
49
50 static void PRxDrop (Ptr<const Packet> m_phyRxDropTrace){
51
52 //std::cout<<"PhyRXDrop at == " << Simulator::Now ().GetSeconds () <<
53 "\n";
54 rx_drop += 1;
55 }
56 static void PTxDrop (Ptr<const Packet> m_phyTxDropTrace){
57

```

```

58 //std::cout<<"PhyTXDrop at == " << Simulator::Now ().GetSeconds () <<
"\n";
59 tx_drop +=1;
60 }
61
62
63 void handler (int arg0)
64 {
65 std::cout << "The simulation is now at: "<<arg0 <<" seconds" << std::endl;
66
67 //ATANDA
68 simPeriodA = arg0;//atanda
69 gpsr::PositionTable obj3;//atanda
70 obj3.GetSimPeriod (simPeriodA); //atanda
71 }
72
73
74 //-----
-----
75 //ATANDA
76 void PrintPositionTableOutputs ()
77 {
78 gpsr::PositionTable obj;
79 obj.BestNeighbourOutputs ();
80 }
81 //-----
-----
82
83 //ATANDA
84 uint32_t SendSimSize ()
85 {
86 gpsr::PositionTable obj4;
87 obj4.GetSimSize (simSizeA);
88 }
89 //-----
-----
90
91
92 //----
93
94 class GpsrExample
95 {
96 public:
97 GpsrExample ();
98 /// Configure script parameters, \return true on successful configuration
99 bool Configure (int argc, char **argv);
100 /// Run simulation
101 void Run ();
102 /// Report results
103 void Report (std::ostream & os);
104
105 void writeToFile(uint32_t lostPackets, uint32_t totalTx, uint32_t
totalRx, double
hopCount,double count, double delay);
106
107 //ATANDA
108 void CsvPacketsAveragesWrite(uint32_t lostPackets, uint32_t totalTx,
uint32_t totalRx,

```

```

double hopCount,double count, double delay);
109
110 private:
111
112 uint32_t size;
113
114 std::string phyMode;
115 /// Width of the Node Grid
116 uint32_t gridWidth;
117 /// Distance between nodes, meters
118 double step;
119 /// Simulation time, seconds
120 double totalTime;
121 /// Write per-device PCAP traces if true
122 bool pcap;
123 // seed to generate random numbers
124 uint32_t seed;
125 std::string path;
126 Time entrylifetime;
127 uint32_t headersize;
128 uint32_t packetsize;
129 bool newfile;
130 std::string algorithm;
131 int sourcenode;
132 int destinationNode;
133 int speed;
134 uint32_t drift;
135 uint32_t nPairs;
136
137
138 NodeContainer AllNodes;
139 NodeContainer nodes;
140 NetDeviceContainer devices;
141 Ipv4InterfaceContainer interfaces;
142 /\}
143
144 private:
145 void CreateNodes ();
146 void CreateDevices ();
147 void InstallInternetStack ();
148 void InstallApplications ();
149
150 };
151
152 int main (int argc, char **argv)
153 {
154 GpsrExample test;
155 if (! test.Configure(argc, argv))
156 NS_FATAL_ERROR ("Configuration failed. Aborted.");
157
158 test.Run ();
159 test.Report (std::cout);
160 return 0;
161 }
162
163 //-----
164 GpsrExample::GpsrExample () :

```

```

165 // Number of Nodes
166 size (50),
167 // Grid Width //Default is 10. Has no effect in my simulations, atanda
168 gridWidth(10),
169 // Distance between nodes //Default is 100, Has no effect in my
simulations, atanda
170 step (100),
171 // Simulation time
172 totalTime (200),
173 // Generate capture files for each node
174 pcap (false), //default is false, atanda
175 //seed to generate random numbers
176 seed (1394),
177 path("outputs/"),
178 packetSize(512),
179 algorithm("gpsr"),
180 newfile(true),
181 speed(15),
182 drift(0), //Has no effect in my simulations, atanda
183 nPairs(1),
184 phyMode ("OfdmRate3MbpsBW10MHz")
185
186 {
187 }
188
189
190 bool
191 GpsrExample::Configure (int argc, char **argv)
192 {
193
194 CommandLine cmd;
195
196 cmd.AddValue ("pcap", "Write PCAP traces.", pcap);
197 cmd.AddValue ("size", "Number of nodes.", size);
198 cmd.AddValue ("time", "Simulation time, s.", totalTime);
199 cmd.AddValue ("step", "Grid step, m", step);
200 cmd.AddValue ("seed", "seed value", seed);
201 cmd.AddValue ("path", "path of results file", path);
202 cmd.AddValue ("conn", "number of connections", nPairs);
203 cmd.AddValue ("algorithm", "routing algorithm", algorithm);
204 cmd.AddValue ("newfile", "create new result file", newfile);
205 cmd.AddValue ("speed", "node speed", speed);
206
207 // disable fragmentation for frames below 2200 bytes
208 Config::SetDefault
("ns3::WifiRemoteStationManager::FragmentationThreshold" ,
StringValue ("2200"));
209 // turn off RTS/CTS for frames below 2200 bytes
210 Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold" ,
StringValue (
"500"));
211 Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode" ,
StringValue (
phyMode));
212 cmd.Parse (argc, argv);
213 return true;
214 }
215

```

```

216 void
217 GpsrExample::writeToFile(uint32_t lostPackets, uint32_t totalTx, uint32_t
totalRx, double
hopCount,double count, double delay){
218
219 struct stat buf;
220 std::string outputfile =
"results/"+algorithm+"_results/pairs"+std::to_string(nPairs
)+"/"+algorithm+std::to_string(size)+"_results.txt";
221 //std::string outputfile =
"results/pairs"+std::to_string(nPairs)+"/"+algorithm+std::to_string(size)+"_r
esults.txt
";
222 //std::string outputfile = "results/teste.txt";
223 int exist = stat(outputfile.c_str(), &buf);
224
225 std::ofstream outfile;
226 outfile.open(outputfile.c_str(), std::ios::app);
227
228 if (outfile.is_open())
229 {
230 std::cout << "Output operation successfully performed1\n" ;
231 }
232 else
233 {
234 std::cout << "Error opening file";
235 }
236
237 if (newfile == true){
238
239 std::ofstream outfile;
240 outfile.open(outputfile.c_str(), std::ios::trunc);
241
242 if (outfile.is_open())
243 {
244 std::cout << "Output operation successfully performed2\n" ;
245 }
246 else
247 {
248 std::cout << "Error opening file";
249 }
250
251 outfile<< "Seed\t"<<"LostPackets\t"<<"totalTx\t"<<"totalRx\t"<<"PDR
(%) \t"<<
"HopCount\t"<<"Delay (ms) \t"<<"PhyRxDrop\t"<<"PhyTxDrop\n";
252 outfile.flush();
253 exist = 1;
254 }
255
256 if (exist == -1){
257 outfile<< "Seed\t"<<"LostPackets\t"<<"totalTx\t"<<"totalRx\t"<<"PDR
(%) \t"<<
"HopCount\t"<<"Delay (ms) \t"<<"PhyRxDrop\t"<<"PhyTxDrop\n";
258 outfile.flush();
259 }
260
261 // write to outfile
262 outfile <<seed<<"\t"; //Lost packets

```

```

263 outfile.flush();
264 outfile <<lostPackets<<"\t"; //Lost packets
265 outfile.flush();
266 outfile <<(double)totalTx<<"\t"; //Total transmitted packets
267 outfile.flush();
268 outfile <<(double)totalRx<<"\t"; //Total received packets
269 outfile.flush();
270
271 if (count == 0){
272
273 outfile <<0<<"\t"; //PDR
274 outfile.flush();
275 outfile <<0<<"\t"; //Mean Hop Count
276 outfile.flush();
277 outfile <<0<<"\t"; //Mean Delay (ms)
278 outfile.flush();
279
280 }else{
281
282 outfile <<std::fixed<<std::setprecision(2)<<
((double)totalRx/(double)totalTx)*100.0
<<"\t"; //PDR
283 outfile.flush();
284 outfile <<std::fixed<<std::setprecision(2)<< hopCount/count<<"\t"; //Mean
Hop Count
285 outfile.flush();
286 outfile <<std::fixed<<std::setprecision(2)<< delay/count * 1000<<"\t";
//Mean Delay
(ms)
287 outfile.flush();
288
289 }
290
291 outfile <<rx_drop<<"\t";
292 outfile.flush();
293 outfile <<tx_drop<<"\n";
294 outfile.flush();
295
296 outfile.close();
297
298
299 }
300
301
302 //-----
-----
-----
303 //ATANDA
304 void
305 GpsrExample::CsvPacketsAveragesWrite(uint32_t lostPackets, uint32_t
totalTx, uint32_t
totalRx, double hopCount,double count, double delay){
306 std::ofstream MyCsvFile;
307 MyCsvFile.open(
"/home/atanda/ns-allinone-3.23/ns-
3.23/results/gpsr_results/PDR_based/packetsAverag
es_krauss_COMPASS_30to110n_200s.csv", std::ofstream::app);
308 if (MyCsvFile.is_open())

```

```

309 {
310 MyCsvFile << "//----- SimSize: " << size << " ----- PDR CHECK TIME
CYCLE : " << modFactorA*10 << " ----- Seed: " << seed << " -----"
<< std::endl;
311 MyCsvFile <<"LostPackets"<<","<<"totalTx"<<","<<"totalRx"<<","<<"PDR(%) "
<<","<<"HopCount"<<","<<"Delay (ms) "<<","<<"PhyRxDrop"<<","<<"PhyTxDrop\n"
;
312 MyCsvFile <<lostPackets<<",";
313 MyCsvFile <<totalTx<<",";
314 MyCsvFile <<totalRx<<",";
315
316 if (count == 0)
317 {
318 MyCsvFile <<0<<","; //PDR
319 MyCsvFile <<0<<","; //Mean Hop Count
320 MyCsvFile <<0<<","; //Mean Delay (ms)
321 }
322 else
323 {
324 MyCsvFile <<std::fixed<<std::setprecision(2)<< ((double)totalRx
/(double)totalTx)*100.0<<","; //PDR
325 MyCsvFile.flush();
326 MyCsvFile <<std::fixed<<std::setprecision(2)<< hopCount/count<<
","; //Mean Hop Count
327 MyCsvFile.flush();
328 MyCsvFile <<std::fixed<<std::setprecision(2)<< delay/count * 1000
<<","; //Mean Delay (ms)
329 }
330
331 MyCsvFile <<rx_drop<<",";
332 MyCsvFile.flush();
333 MyCsvFile <<tx_drop<<"\n";
334 MyCsvFile.flush();
335 }
336 else
337 { std::cout << "Unable to open Packets Averages csv output file" ;
338 }
339 MyCsvFile.close();
340 }
341 //-----
-----

342 //atanda
343
344 void
345 GpsrExample::Run ()
346 {
347
348
349 SeedManager::SetSeed(seed);
350
351 CreateNodes ();
352 CreateDevices ();
353 InstallInternetStack ();
354 InstallApplications ();
355
356 if (algorithm=="gpsr"){
357 std::cout<<"Using GPSR algorithm...\n";

```

```

358 GpsrHelper gpsr;
359 gpsr.Install ();
360
361 }else{
362 if (algorithm=="mmgpsr"){
363 std::cout<<"Using MMGPSR algorithm...\n";
364 MMGpsrHelper mmgpsr;
365 mmgpsr.Install ();
366 }else{
367 std::cout<<"Using PA-GPSR algorithm...\n";
368 PAGpsrHelper pagpsr;
369 pagpsr.Install ();
370
371 }
372 }
373 std::cout << "Starting simulation for " << totalTime << " s ...\n";
374 std::cout << "Starting simulation for speed " << speed << " ms ...\n";
375
376
377 for (int i=1; i<=totalTime; i++){
378 if (i % 10 == 0) // at every 10s
379 {
380 Simulator::Schedule(Seconds(i), &handler, i);
381 }
382 }
383
384
385
386 FlowMonitorHelper flowmon;
387 Ptr<FlowMonitor> monitor = flowmon.InstallAll();
388
389
390 Simulator::Stop (Seconds (totalTime));
391 Simulator::Run ();
392
393 //Print per flow statistics
394 monitor->CheckForLostPackets();
395 Ptr<Ipv4FlowClassifier> classifier =
DynamicCast<Ipv4FlowClassifier>(flowmon.
GetClassifier());
396 std::map<FlowId,FlowMonitor::FlowStats> stats = monitor->GetFlowStats();
397 uint32_t lostPackets = 0;
398 uint32_t totalRx=0;
399 uint32_t totalTx=0;
400 double delay=0;
401 double count=0;
402 double hopCount=0;
403
404
405
406 for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i=
stats.begin();i!=stats.
end();++i)
407 {
408 Ipv4FlowClassifier::FiveTuple t=classifier->FindFlow(i->first);
409
410 if(i->second.rxPackets!=0)
411 {

```

```

412 totalRx+=i->second.rxPackets;
413
414 //std::cout << "Flow No (for non-zero flow : " << i->first
<<"\n";//Atanda
415 //std::cout << "TotalRx " << totalRx <<"\n";//Atanda
416
417 totalTx+=i->second.txPackets;
418 hopCount+=(i->second.timesForwarded/i->second.rxPackets+1); // this
hopCount is
not suitable for PA-GPSR because of packet duplication. For PA-GPSR HopCount
calculation we need to use the IP TTL field instead (not implemented yet).
This
hopCount can be helpful to calculate network yield though.
419
420 //std::cout << "hopcount " << hopCount <<"\n";//Atanda
421
422 delay+=(i->second.delaySum.GetSeconds()/i->second.rxPackets);
423 count++;
424 lostPackets += i->second.lostPackets;
425 }
426 }
427
428 writeToFile(lostPackets, totalTx, totalRx, hopCount, count, delay);
429
430
431
432 monitor->SerializeToXmlFile("gpsr.flowmon",false,true); //2nd and 3rd
parameters are
used respectively to activate/deactivate the histograms and the per-probe
detailed
stats. // Atanda - the default was "false, false"
433
434
435
436
437 //ATANDA
438 simSizeA = size;
439 SendSimSize ();
440 PrintPositionTableOutputs ();
441
442 gpsr::PositionTable obj5;
443 modFactorA = obj5.ObtainModFactor ();
444
445 // CsvPacketsAveragesWrite(lostPackets, totalTx, totalRx, hopCount,
count, delay);
446 // std::cout << "modFactor: " << modFactorA <<"\n";//Atanda
447 //atanda
448
449
450
451 Simulator::Destroy ();
452 }
453
454
455 void
456 GpsrExample::Report (std::ostream &)
457 {
458 }

```

```

459
460 void
461 GpsrExample::CreateNodes ()
462 {
463     std::cout << "Creating " << (unsigned)size << " nodes .." << " with
"<<"\nPairs<<"
pairs..." <<"\n";
464     nodes.Create (size);
465     // Name nodes
466     for (uint32_t i = 0; i < size; ++i)
467     {
468         std::ostringstream os;
469         os << "node-" << i;
470         Names::Add (os.str (), nodes.Get (i));
471     }
472 }
473
474 std::string m_traceFile;
475 m_traceFile =
"results/tclFiles/speed/"+std::to_string(speed)+"/newNs2mobility"+std::
to_string(size)+".tcl";
476 Ns2MobilityHelper mobility= Ns2MobilityHelper (m_traceFile); //for
tracefile
477 mobility.Install ();
478
479 AllNodes.Add(nodes);
480
481 }
482
483
484
485 void
486 GpsrExample::CreateDevices ()
487 {
488
489     Wifi80211pHelper wifi = Wifi80211pHelper::Default ();
490
491     YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
492     YansWifiChannelHelper wifiChannel;
493
494     wifiChannel.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
495     wifiChannel.AddPropagationLoss ("ns3::RangePropagationLossModel",
496     "MaxRange", DoubleValue (250.0));
497     wifiChannel.AddPropagationLoss ("ns3::TwoRayGroundPropagationLossModel",
498     "SystemLoss", DoubleValue(1),
499     "HeightAboveZ", DoubleValue(1.5));
500
501
502 // For range near 250m
503     wifiPhy.Set ("TxPowerStart", DoubleValue(33));
504     wifiPhy.Set ("TxPowerEnd", DoubleValue(33));
505     wifiPhy.Set ("TxPowerLevels", UIntegerValue(1));
506     wifiPhy.Set ("TxGain", DoubleValue(0));
507     wifiPhy.Set ("RxGain", DoubleValue(0));
508     wifiPhy.Set ("EnergyDetectionThreshold", DoubleValue(-61.8));
509     wifiPhy.Set ("CcaModelThreshold", DoubleValue(-64.8));
510

```

```

511 wifiPhy.SetChannel (wifiChannel.Create ());
512
513 NqosWaveMacHelper wifiMac = NqosWaveMacHelper::Default ();
514
515 wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
516 "DataMode",StringValue(phyMode),
517 "ControlMode",StringValue(phyMode));
518 devices = wifi.Install (wifiPhy, wifiMac, AllNodes);
519
520 }
521
522 void
523 GpsrExample::InstallInternetStack ()
524 {
525 InternetStackHelper stack;
526 std::string prefix = "10.0.0.";
527 std::string suffix_dst = std::to_string(destinationNode+1); //the nodes
numbers starts
from zero, so we need to add 1 for the IP addresses
528 std::string suffix_source = std::to_string(sourcenode+1);
529 std::string dst_s = prefix+suffix_dst;
530 std::string source_s = prefix+suffix_source;
531 char const *dst_c = dst_s.c_str();
532 char const *source_c = source_s.c_str();
533
534 if (algorithm=="gpsr"){
535 GpsrHelper gpsr;
536 stack.SetRoutingHelper (gpsr);
537 stack.Install (AllNodes);
538 }else{
539 if(algorithm=="pagpsr"){
540 PAGpsrHelper pagpsr;
541 stack.SetRoutingHelper (pagpsr);
542 stack.Install (AllNodes);
543 }
544 else{
545 MMGpsrHelper mmgpsr;
546 stack.SetRoutingHelper (mmgpsr);
547 stack.Install (AllNodes);
548 }
549 }
550
551
552 Ipv4AddressHelper address;
553 address.SetBase ("10.0.0.0", "255.255.0.0");
554 interfaces = address.Assign (devices);
555
556 }
557 void
558 GpsrExample::InstallApplications ()
559 {
560
561 uint16_t port = 9; // well-known echo port number
562
563 uint32_t maxPacketCount = 10000; // number of packets to transmit
564 Time interPacketInterval = Seconds (0.2); // interval between packet
transmissions
565 UdpEchoServerHelper server1 (port);

```

```

566 ApplicationContainer apps;
567
568 srand(276); //same seed to random pairs be the same at all simulations,
can be any
number (use 276 to reproduce our results)
569
570 int source;
571 int dest;
572 std::vector<std::pair<int,int>> source_dest_pairs;
573 std::vector<std::pair<int,int>>::iterator it;
574 bool exist;
575 float start_app;
576 for (uint32_t i = 0; i < nPairs ; i++){
577 source = 0;
578 dest = 0;
579 exist = true;
580 start_app = static_cast <float> (rand()) / static_cast <float>
(RAND_MAX);
581 while((source == dest) || exist == true){
582
583 source = rand() % size;
584 dest = rand() % size;
585 it = std::find(source_dest_pairs.begin(), source_dest_pairs.end(), std:::
make_pair(source,dest));
586 if (it != source_dest_pairs.end()){
587 exist = true;
588 }else{
589 source_dest_pairs.push_back(std::make_pair(source,dest));
590 exist = false;
591 }
592 }
593 UdpEchoClientHelper client (interfaces.GetAddress (dest), port);
594 client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
595 client.SetAttribute ("Interval", TimeValue (interPacketInterval));
596 client.SetAttribute ("PacketSize", UintegerValue (packetsize));
597 apps = client.Install (nodes.Get (source));
598 apps.Start (Seconds (start_app));
599 apps.Stop (Seconds (totalTime-0.1));
600 }
601
602 Config::ConnectWithoutContext(
"/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/PhyRxDrop" ,
MakeCallback(&PRxDrop));
603 Config::ConnectWithoutContext(
"/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/PhyTxDrop" ,
MakeCallback(&PTxDrop));
604 }
605

```

- END -