Unmanned Aeronautical Ad-hoc Networks: Enhancing the Reactive-Greedy-Reactive Protocol and Introducing a New Mobility Model

by

Jean-Daniel Medjo Me Biomo

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
January 2014

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Unmanned Aeronautical Ad-hoc Networks: Enhancing the Reactive-Greedy-Reactive Protocol and Introducing a New Mobility Model

submitted by **Jean-Daniel Medjo Me Biomo**

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

_____
Professor Thomas Kunz, Thesis Co-supervisor

_____
Professor Marc St-Hilaire, Thesis Co-supervisor

_____
Professor Roshdy Hafez, Chair,
Department of Systems and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
January, 2014

Abstract

Unmanned Aeronautical Ad-hoc Networks (UAANETs) are infrastructure-less and self-organizing networks that are formed by small and medium sized Unmanned Aerial Vehicles (UAVs) that are deployed for a wide range of civilian and military applications.

Having a reliable routing protocol for communication between the UAVs is critical. Our goal in this research is twofold. First, we enhance one of the existing routing protocols, the Reactive-Greedy-Reactive (RGR) protocol. In doing so, we propose the Optimized-RGR. Secondly, we propose the Enhanced Gauss-Markov (EGM) mobility model for UAANET simulations to replace the widely used, but unrealistic, Random Waypoint (RWP) mobility model.

Simulations performed using the OPNET simulator show that Optimized-RGR outperforms RGR. There is a 5.3% increase in Packet Delivery Ratio at a negligible cost in latency. Furthermore, realistic mobility models, including EGM, show a lot of network partitioning. Therefore, this should be taken very seriously when developing a routing protocol for UAANETs.

To my parents François and Jacqueline

Acknowledgements

I thank the Almighty God for providing me with the opportunity and the intellectual ability to do research.

I would like to thank my supervisors, Professors Thomas Kunz and Marc St-Hilaire, for their great guidance and endless support throughout the project. With them, I have learned how to do research and I also have acquired a lot of discipline.

Last, but not least, I would like to thank Yi Li, a former visiting student from China, and Dr. Yifeng Zhou, researcher at Communication Research Center (CRC) Canada, for technical support with the OPNET simulator.

Jean-Daniel Medjo Me Biomo

Table of Contents

List of Tables

List of Figures

List of Acronyms

AODV	Ad-hoc On-demand Distance Vector
BMN	Best-Moving Node
CTS	Clear To Send
DN	Destination Node
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
EGM	Enhanced Gauss-Markov
FN	Forwarding Node
FSM	Finite State Machine
GFG	Greedy-Face-Greedy
GGF	Greedy Geographic Forwarding
GM	Gauss-Markov
GOAFR	Greedy Other Adaptive Face Routing
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GPVFR	Greedy Path Vector Face Routing
HI	HELLO Interval
IC-MANET	Intermittently-Connected MANET
IN	Intermediate Node
LAR	Location-Aided Routing
LAROD	Location Aware Routing for Opportunistic Delay-tolerant networks
LoDiS	Location Dissemination Service
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
MAODV	Modified AODV
MN	Mobile Node
mpRGR	mobility prediction RGR
O-AODV	Optimized AODV
OLSR	Optimized Link State Routing
PDR	Packet Delivery Ratio

RD	Random Direction
RERR	Route Error
RGF	Randomized Geographic Forwarding
RGR	Reactive-Greedy-Reactive
RREP	Route Reply
RREQ	Route Request
RRI	Robust Route Index
RSF	Route Stability Factor
RWP	Random Waypoint
sf1mpRGR	Scoped flooding 1 + mobility prediction RGR
sf2mpRGR	Scoped flooding 2 + mobility prediction RGR
SN	Source Node
ST	Smooth-Turn
UAANET	Unmanned Aeronautical Ad Hoc Network
UAV	Unmanned Aerial Vehicles
ZRP	Zone Routing Protocol

Chapter 1 : Introduction

Unmanned Aeronautical Ad-hoc Networks (UAANETs) [1] are a type of Mobile Ad-hoc Networks (MANETs) [3], which are infrastructure-less and self-organizing networks. The specificity of UAANETs is that they are exclusively airborne and are formed by small and medium sized Unmanned Aerial Vehicles (UAVs) [2] that can be deployed for a wide range of civilian and military applications. Those applications include, but are not limited to: rescuing or searching missions in the event of natural disasters such as tsunamis, hurricanes, earthquakes etc.; the establishment and maintenance of temporary Internet or telephone networks to allow communication in/with such devastated areas; tracking missions for the purpose of enemy surveillance on battlefield, border control, etc. In this context, having a routing protocol that allows efficient communication between the UAVs is critical for reliability and missions' delay.

Designing routing protocols for UAANETs is very challenging due to the highly changing network topology that ensues from the high mobility of UAVs combined with their limited transmission ranges. In MANETs, thus in UAANETs as well, the routing protocols can be classified into two groups [4]: topology-based protocols and position-based protocols. Topology-based protocols are routing protocols where the information about the links in the network is used in order to establish and maintain routes. Among these topology-based protocols, we further distinguish proactive (e.g. Destination Sequenced Distance Vector (DSDV) [5], Optimized Link State Routing Protocol (OLSR) [6], etc.), reactive (e.g. Ad-hoc On-demand Distance Vector (AODV) [7], Dynamic Source Routing (DSR) [8], etc.) and hybrid (e.g. Zone Routing Protocol (ZRP) [9]) protocols. In the group of position-based protocols, we have protocols that do not rely on link states. Instead, only the nodes' physical location information is essential. Those protocols are also called geographic routing protocols, and the main one is Greedy Geographic Forwarding (GGF) [14]. The idea is to forward the data packet to the neighbor whose location is closer to the destination than that of the forwarding node (FN).

Reactive-Greedy-Reactive (RGR) [1] is a routing protocol designed for UAANETs. RGR encompasses both the characteristics of topology-based protocols and position-based protocols. RGR is, basically, a combination of AODV and GGF with no recovery strategy. RGR, as its name suggests, operates in 2 modes that alternate: the Reactive/AODV mode and the Greedy/GGF mode. In brief, in RGR, a switch to the GGF mode is performed whenever a forwarding node encounters a broken link to the next hop in AODV mode. In the AODV mode, during route discovery/construction, freshness and length (in hops) are the criteria for route selection. However, due to the dynamics of UAANETs, routes (selected by the aforementioned criteria) are being declared invalid very frequently. This route breakage frequency can be lowered if we make sure that the routes that are selected are those with some level of reliability/stability. Therefore, adding a reliability criterion in the route selection/construction process of RGR is a focus of this work.

In GGF, when the FN has no neighbor whose location is closer to the destination than it is, the packet is dropped and GGF is said to have failed. There exists in the literature a lot of recovery strategies to address this GGF failure, and those strategies result in different flavors of a geographic routing protocol. The strategies aim to salvage those packets that would have otherwise been dropped when GGF fails. However, many of these strategies have issues, ranging from high overhead, high complexity, and inapplicability in UAANETs. Even though we focus on 2D UAANETs in this research (mostly for their relative simplicity), we keep in mind that real UAANETs are in 3D. Therefore, every mechanism or enhancement that we propose ought to be easily extendable to 3D. Now, some GGF failure recovery strategies, such as the planar graph based ones, are upfront not extendable to 3D, which is why they are deemed inapplicable to UAANETs. All the aforementioned shortcomings are the reason why a new recovery strategy to tackle GGF failure in the context of UAANETs is also a focus of this work.

The movement pattern of UAVs in UAANETs depends on the type of application that they are being used for. Surveys presented in [11] showed that mobility models have a significant impact on the performance of network routing protocols. Therefore, the choice of a mobility model is critical. It is even more critical when we take into account the physical constraints (mechanical and aerodynamic) of UAVs. UAVs tend to maintain the same speed and change direction by making turns with large radii [10], which is not the case for ground vehicles that can

afford to make sudden stops, sharp turns, etc. In this work, we propose a mobility model that captures realistic movement patterns of UAVs.

1.1 Research Objectives

The goal of this thesis is twofold. First, we propose two enhancements to the existing RGR routing protocol. In fact, one enhancement is specific to topology-based protocols in general and applies to the existing RGR protocol in particular. The other enhancement is for Geographic routing in general and also applies to RGR as well. Secondly, we propose a realistic mobility model for simulation in UAANETs. The performance of the existing RGR protocol, mainly in terms of Packet Delivery Ratio (PDR), suggests that there is still a lot of room for improvement. Since RGR has 2 modes of operation (i.e. Reactive and Greedy Geographic), it is only logical to seek enhancements in both modes yielding the two enhancements that we propose. On the other hand, the existing RGR as well as a number of other routing protocols in MANETs and UAANETs have been tested under an unrealistic mobility model: the Random Waypoint (RWP) mobility model. RWP leads to sudden stops and sharp turns which are not possible in the context of UAANETs due to UAVs' physical and aerodynamic limitations; therefore the need for a realistic mobility model in order to draw more reliable conclusions on the performance of routing protocols in general, and RGR in particular.

1.2 Thesis Contributions

The first part of the thesis consists of enhancing the existing RGR protocol. This part is twofold as we address the two modes of RGR: the Reactive mode and the GGF mode. First, a route stability/reliability criterion is introduced in the selection/construction of routes during the Reactive mode of RGR. In the existing RGR protocol, freshness and length (in terms of hops) are the ultimate criteria for the selection of a route for data packet delivery. The actual state of the links constituting this route is not taken into account. Given the high mobility of the UAVs and their limited transmission range, some links constituting a fresh-enough or short-enough route

might be at the brink of breakage at the moment that the route is selected. This would result in a very short-lived route. Therefore, we propose to consider stable/reliable enough links for route construction before their freshness and length is even taken into consideration. This results in more stable routes being created or selected during the Reactive mode of RGR. We call the new version of RGR, featuring this link stability criterion, Modified-RGR. Simulation results show, compared to RGR, an increase in PDR coupled with a slight drop in control overhead at no extra cost in terms of end-to-end delay.

Next, we propose a recovery strategy for GGF failure. Our proposal is motivated by the shortcomings of existing strategies: their high complexity, the high overhead that they induce, and their non-extendibility to 3D, which makes them unrealistic for real-life deployments of UAANETs. GGF consists of forwarding a data packet to the neighbor (one-hop) that is closer to the Destination Node (DN) than the current forwarding node. When no such neighbor exists, the packet is dropped and GGF is said to have failed. Our strategy consists of forwarding the packet to the Best-Moving Node (BMN) when GGF fails. We define the BMN as the node that shows the best move toward the DN. This should result in an increase in PDR since packets are given a second chance to be forwarded when GGF fails. We integrate this recovery strategy into the GGF mode of RGR (in fact Modified-RGR) with the intention of further enhancing it, resulting in a newer version of RGR: Optimized-RGR. Simulations showed an increase in PDR (compared to Modified-RGR) at the cost of slightly higher end-to-end delay, and at no additional cost whatsoever in terms of control overhead.

Finally, in the last part of the thesis, we propose the Enhanced Gauss-Markov (EGM) mobility model in order to replicate realistic movement patterns of UAVs. The motivation here is that the RWP mobility model that has been widely used thus far, including in our simulations, is not very realistic as it allows for the nodes to turn sharply and to stop suddenly; which cannot be the case in the context of real UAANETs. We implemented EGM in OPNET and compared it with other mobility models. Simulations showed that, compared to RWP, EGM causes a significant number of network partitions, which has a negative impact on the routing protocol's performance. Smooth-Turn (ST) [10], [12], [13], an existing realistic model that we also implemented in OPNET, also showed with acuity the same problem of network partitioning. All this therefore suggests that for sparse deployments of UAVs, network partitions are hard to avoid

and therefore should be considered in the design of a routing protocol, which is quite different from the approach in many MANET routing protocols.

In summary, we present two major contributions: a twofold contribution (Route stability and GGF failure recovery strategy) to the RGR protocol and a contribution in mobility models. At first glance, the two contributions are not directly related. But in reality, whichever contribution we make with respect to a routing protocol for UAANET can only be reliably tested if a realistic mobility model is used. In that regard, our two contributions are complementary in some sense.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 reviews related work in MANETs and UAANETs spanning from routing protocols to mobility models. In Chapter 3, the two enhancements to the existing RGR protocol are proposed and discussed. The Enhanced Gauss-Markov mobility model is proposed in Chapter 4. All OPNET simulation results are presented in Chapter 5. Finally, Chapter 6 concludes the thesis and points out avenues for future work.

Chapter 2 : Related Work / Literature Review

2.1 Introduction

As already stated, our ultimate goal in this thesis is to enhance RGR and to propose a new mobility model for UAANETs. Our proposition for a new mobility model comes from the fact that the currently used RWP model has some unrealistic features. In this chapter, we present the existing RGR routing protocol and we go over the literature on some routing protocol enhancement ideas and mobility models. More precisely, a review on routing protocols is presented in Section 2.2 whereas Section 2.3 provides a review on existing mobility models.

2.2 Literature Review on Routing Protocols

The RGR protocol, as its name (Reactive-Greedy-Reactive) suggests, has two modes of operation: the Reactive mode and the Greedy mode. Each of these modes operates based on or similar to an existing routing protocol. In this section, we first review the protocols on which the modes of RGR are based on. Then we present RGR itself. Finally, we review some improvements that exist in the literature with respect to the protocols constituting the basis of RGR.

2.2.1 The Constituting Protocols of RGR

As already mentioned, RGR has two modes of operation. The Reactive mode is based on the AODV protocol, whereas the Greedy mode is based on the GGF protocol.

a. The AODV Protocol

AODV is a reactive routing protocol designed for MANETs. It is said to be reactive because routes are not established and maintained in advance of the transmission of eventual data packets. Instead, routes are established when a data packet is to be transmitted over the network, thus the "on-demand" part of the name. Those on-demand established routes are invalidated and eventually erased if a certain period of time elapses without them being used for data transmission. The protocol has two phases: route discovery and route maintenance. When a Source Node (SN) has a data packet (say from its application layer) to transmit to a destination node, it looks up its routing table to determine whether it has a route to that DN. If it does, then the data packet is sent through that route. Conversely, if it does not, then it tries to establish a route. It does so by broadcasting route requests (RREQs) to all its neighbours who happen to be Intermediate Nodes (IN) unless they are the requested DN. Sequence numbers are assigned so that any node immediately discards duplicate copies of the same RREQ upon reception. When a node receives an RREQ, if it is the DN, it sends a route reply (RREP) back to the SN using the route established by the hops (nodes) the RREQ had gone through. If the IN is not the DN, but has a route to the DN, it also sends an RREP back to the SN and, optionally, sends a Gratuitous RREP to the actual DN, in order to "inform" it of the newly created route all the way from the SN. In the event the IN is neither the destination nor has a route to the latter, it just rebroadcasts the RREQ to continue the search for a route. The ability to rebroadcast an RREQ is also decided by the distance (in terms of hops) traveled by the latter. In order to cap the protocol overhead, there is a limit to the number of hops an RREQ can go through. Beyond that limit, the RREQ is dropped. When the SN receives the RREP, it now has a route to the DN. It updates its routing table and sends the data packet through the next hop on that route. That is the route discovery process. However, when the data packet is being routed, it can happen that a next hop is unreachable (broken link) due to the mobility of the nodes. The IN where this occurs sends out a route error (RERR) message to the SN and drops the data packet if local repair is not enabled. When local repair is enabled, the IN holds on to the data packet while it tries to repair the route locally by sending out new RREQs in order to establish a new route to the destination. Note that for RGR, local repair is not enabled. It is replaced by GGF that we present later. However, RERRs are sent out even when local repair per se is not enabled. That is the route maintenance

part: a generated RERR propagates towards all sources having a route via the failed link, and erases all broken routes on the way. A source, upon receiving an RERR, initiates a new route discovery if it still needs the route. One important feature of AODV allows nodes to broadcast HELLO messages at regular time intervals in order to advertise themselves and let their neighbors and neighbors of their neighbors etc. establish routes to them. A node broadcasts a HELLO message if it meets the following two conditions: i) it is part of an active route and ii) the predefined time period has elapsed without it sending a broadcast message. There are two types of broadcast messages: RREQs and HELLOs. An active route is a route to a destination that is marked as valid in the routing table of the node. Therefore, if a given node has at least one valid entry in its routing table (maybe from previously receiving HELLOs from neighbors), then it is considered part of an active route.

The other mode (greedy) of RGR is based on the GGF protocol that we briefly present in the next sub-section.

b. The GGF Protocol

GGF is a geographic routing protocol whose idea is to forward the data packet to the neighbor that has a location closer to the destination than that of the forwarding node. Note that closeness to destination here is measured in terms of Euclidian distance. In order to calculate the distances, the FN obviously needs the location information of the destination and of the neighbors. Fortunately, this is not a limitation as, in UAANETs, the nodes are equipped with GPS (Global Positioning System); making the location information available at no extra cost. For the geographic routing protocols, this location information is made available through a location service. In the literature [1], [45], location services are classified in three groups: flooding-based, quorum-based and home-based. In a flooding-based service for instance, a node can disseminate its location periodically. This is called a proactive service. In a flooding-based reactive service, when a node does not have the updated information of a target, a search message is flooded into the network.

GGF fails when a packet arrives at a node that has no neighbor that is closer to the destination than it is. In this situation, the forwarding node is declared to be a *void node*. In the

occurrence of a *void node* situation, the data packet is simply dropped unless there is a recovery strategy in place.

The two protocols presented above (AODV and GGF) were combined together to a certain extent in order to obtain the RGR protocol that we present next.

2.2.2 The RGR Protocol

The RGR protocol is a routing protocol based on AODV and GGF. RGR works like AODV until a forwarding node faces a broken link. By forwarding node, we mean a node that has received a data packet and is trying to send it to the packet's next hop (in the case of control message packets, we call such nodes Intermediate Nodes). In AODV, when the next hop is unreachable (broken link), we have two options: if local repair is not enabled, the packet is dropped. Otherwise, when local repair is enabled, the FN holds on to the packet and broadcasts an RREQ in order to "repair" the broken link. Eventually, it will send the data packet using the newly re-established or repaired route. In the case of RGR, when the FN faces a broken link, instead of performing local repair like in AODV, it switches to the GGF mode. The GGF mode here works as follows: the FN calculates its own distance to the DN alongside the distance to the same DN of all its current neighbors. If there is a neighbor that is closer to the DN than the FN, the FN forwards the data packet to that neighbor. If there is no such a neighbor, the data packet is dropped altogether. When the neighbor node receives the data packet, it checks if it has a route to the DN established by means of the reactive part of the protocol (AODV functionality). If it does, the packet is forwarded using that route; and if it does not, then it switches to GGF too and so on and so forth. This scheme gives the data packet a second chance to be transmitted without incurring additional overhead costs (due to new local route discovery as in AODV with local repair). In RGR, unlike in GGF protocol in general, the nodes' location information is piggybacked onto control messages (from the Reactive mode) in order to be disseminated to other nodes for distance calculation purposes (in the GGF mode). Therefore, no location service is needed.

A few enhancements for the original RGR were also developed, resulting in protocol variants called mpRGR, sf1mpRGR, and sf2mpRGR.

a. mpRGR

In mpRGR, mobility prediction (mp) is added [15]. To accomplish this, the speed and the direction of the nodes are piggybacked in the control messages. When a node receives location information of another node in a control packet, it can predict where that node currently is. In fact, the information in the packet is not real-time; there is a timestamp that indicates when that location information was recorded. Based on that timestamp, speed, and direction, the current location can be approximated as follows [15]:

$$X_{predict} = X_{last} + V \times \sin(\theta) \times (current\_time - timestamp) \quad (2.1)$$

$$Y_{predict} = Y_{last} + V \times \cos(\theta) \times (current\_time - timestamp) \quad (2.2)$$

where $X_{predict}$ and $Y_{predict}$ are the coordinates of the predicted/approximated current location of the node of interest. X_{last} and Y_{last} are the last known location coordinates of the node of interest, recorded at time *timestamp*. At that time *timestamp*, V and θ, the speed and direction respectively of the node, were also recorded.

When there is a data packet to forward, the forwarding node predicts the current location of the next hop. If it is found out of transmission range, then the route is invalidated right away. For comparison, in the original RGR protocol, a route is invalidated after missing three consecutive HELLO messages, which implies that a certain period of time (up to 3 times the HELLO message interval (3 x 1 second)) can elapse between a link breaking and the protocol detecting such a link break. Therefore, by the time the switch to GGF is made, many data packets have already been dropped; whereas in mpRGR, there is no such delay and therefore we have more switches to GGF in mpRGR than in original RGR, and also a higher PDR. Mobility prediction also brings more accuracy in the process of selecting the neighbour to forward the data packet to during GGF.

In the next two versions of RGR, in addition to mobility prediction (mp), scoped flooding (sf) is added. Scoped flooding concerns the flooding of RREQs in the network. The idea, similar to LAR (Location-Aided Routing) [16], is to limit the flooding of RREQs only to the part of the network where the destination is (expected to be) located; and that will be the search area. An intermediate node will re-broadcast a received RREQ only if it is part of the search area. If not, the RREQ will be dropped. The search area is modelled by means of Euclidian distance to destination as follows: when an IN receives an RREQ, it calculates its own distance to the destination. If that distance is greater than the one of the RREQ's previous hop to the same destination, then the IN is considered to be out of the search area. For this to be possible, the RREQ packet should have a field that contains the distance to the destination of the previous hop. Before an IN re-broadcasts an RREQ, it sets its own distance to the destination in that field of the RREQ packet if needed. In the two versions of RGR that are presented below, the conditions in which distances are set in the RREQ as well as the decision of whether to rebroadcast or to drop the RREQ are slightly different.

b. sf1mpRGR

In the sf1mpRGR scheme [15], when a source node initiates a route discovery for the first time, it does not have the intended destination's location information, nor does it expect any other intermediate node in the network to. Therefore, it puts invalid coordinates of the destination as well as an invalid distance (a value of -1 or 0 for instance) to the destination in the appropriate fields of the RREQ before broadcasting it. All the INs receiving that RREQ with invalid location information also just rebroadcast the RREQ. That is called blind flooding of RREQs. The SN will wait until it receives an RREP from the destination node. When that reception occurs, the SN then creates a valid route entry to that DN in its routing tables. If another route discovery is required for that same DN (because the route entry has been invalidated), the SN sets the predicted location information (based on the previously learnt location information) of the DN in the RREQ as well as its calculated distance to the DN. Upon reception of an RREQ with valid DN coordinates and distance, an IN will calculate its own distance to the DN based on those RREQ-received coordinates. If its distance is greater than the distance included in the received RREQ, then it drops the RREQ. Otherwise, it replaces the

distance in the RREQ by the newly calculated one and rebroadcasts the RREQ. Basically, scoped flooding here is realised based on the SN knowledge of the DN's predicted coordinates (prediction made based on last recorded DN's coordinates). If the SN does not receive an RREP before a predefined timeout, it will re-issue another RREQ (second retry) and sets its distance to 20% higher than the one in the previously issued RREQ (case where that first RREQ had valid DN coordinates). It can retry up to 5 times (with a 20% increase of the distance each time compared to the previous), and if it still has not received an RREP, it will resort to blind flooding.

c. sf2mpRGR

sf2mpRGR [15] works like sf1mpRGR to a certain degree. The difference here is that the INs' knowledge of the DN location is now taken into account. When an IN receives an RREQ, it now first checks whether or not it has the location information of the DN in its table. If it does not, it just rebroadcasts the RREQ (blind flooding). And when it does, it calculates its distance to the DN and checks the distance to the DN included in the received RREQ. If the distance included in the RREQ is invalid (denoting that the SN and all the previous INs do not know the DN coordinates), then the IN sets it to the newly calculated one before rebroadcasting the RREQ. This also happens if the RREQ-received distance is valid but greater than the newly calculated distance. On the other hand, if the distance included in the received RREQ is valid and smaller than the newly calculated distance, then the IN discards the RREQ because it realizes that it is not part of the modeled search area. In the event that the source unsuccessfully waits for a RREP for a timeout period, it will re-issue a RREQ with a distance value of 0.

The results in [15] showed comparisons in terms of performance among the protocols. The protocols considered were the following: original RGR, its variants (mpRGR, sf1mpRGR, sf2mpRGR), and AODV (with local repair). The performance was evaluated according to three metrics: the packet delivery ratio, the routing overhead, and the end-to-end delay. The original RGR protocol showed a higher PDR than AODV (with local repair). And that is mainly because, as we saw in RGR, data packets do get a second chance to be transmitted using a fundamentally different approach (geographic forwarding) when a link breaks. An even higher PDR was

achieved by the versions of RGR that feature mobility prediction (mpRGR, sf1mpRGR and sf2mpRGR). Mobility prediction makes it possible to approximate the real-time coordinates of our neighbours and the DN. This results in a better selection of the node to forward the packet to in GGF. Not to mention that, with mobility prediction, broken links are detected immediately and therefore data packets are not lost due to wrongly marked valid links to next hops. As a result, more packets get to the DN with these versions of RGR. Furthermore, in terms of the routing overhead (that is the sum of RREQs, RREPs, RERRs, and HELLO messages per second), the two versions of RGR that feature scoped flooding showed the best result (lowest overhead). One explanation to that is that scoping the flooding of RREQs dramatically reduces the number of RREQs that constitute the overhead. Finally, in terms of end-to-end delay, all the protocols considered here have the same latency more or less.

The general conclusion to draw from these results in [15] is that, so far, sf1mpRGR and sf2mpRGR are the versions of RGR to move forward with as they present the highest PDR and the lowest routing overhead at no additional cost in terms of end-to-end delay. As we can choose one or the other, let us choose sf1mpRGR for further improvements. Therefore, in the remainder of this document, what we will refer to as RGR is the actual sf1mpRGR version of the original RGR seen above. The improvements that we propose are discussed in Chapter 3, but we first review some related work on the items that constitute the reactive and the greedy modes of RGR.

2.2.3 Review on Potential Improvements

a. Improvements on AODV

MODIFIED-AODV is proposed in [17] as an improvement to conventional AODV. The main modification to original AODV is made during the route discovery phase. The key idea is finding a robust route by making use of a metric called the Robust Route Index (RRI) which is computed as the weighted sum of path hop-length, and average speed between the individual nodes with nodal delay identifiers such as congestion identifiers. In the algorithm during route discovery, each node forwards the RREQ packet with the highest RRI among multiple RREQ packets received. Unlike in AODV, here each node waits for a predefined amount of time in

order to collect several RREQ packets and then selects the one that provides the highest robustness level while offering the shortest route among all such received RREQ packets. The authors conducted simulations varying the number of nodes from 20 to 100. The results showed an improvement to both the packet delivery ratio and the end-to-end delay while paying a cost of an increase in overhead compared to the original AODV protocol. The authors simply present the increase in overhead as a side effect paid to achieve robust delivery and high performance. No clear explanation/discussion is provided though for the exact origin of that surprising additional overhead. It is note-worthy to mention that the nodes' speeds varied from 2 m/s to 10 m/s, which is a relatively low level of mobility. This solution might not be an interesting one to include in the RGR protocol, since RGR is designed for UAANETs which are essentially networks with high mobility; plus, we might not be willing to incur additional routing overhead costs.

In [18], the authors present another modified AODV protocol (MAODV). It modifies the RREQ and the RREP packet formats of the route discovery. The way it works is as follows: every time an intermediate node receives an RREQ/RREP, it appends its own address to it before forwarding it (when forwarding is needed). Consequently, the routing table is populated and during subsequent route requests, the probability of routes being present in the routing table is higher. Thus, the number of route discovery cycles decreases as compared to the basic AODV and therefore the efficiency increases. At least, that is the claim of the authors. The simulations were conducted by varying two parameters: pause time and number of nodes. In fact, in conventional AODV whenever a node receives a control packet it creates an entry in its routing table for the node (previous hop) it received that packet from. Therefore, we do not believe that MAODV is really an improvement to AODV. The simulation results [18] confirmed our belief (doubts) as there is no difference in terms of packets received for instance between the two protocols, at least for a number of nodes similar to what we would expect in the UAANET scenarios of our RGR protocol (around 20-30 nodes).

Another Modified AODV routing protocol (MAODV) is proposed in [19]. Here the idea is to take route stability into account in order to establish a more stable path between source and destination. For this purpose, changes are made to HELLO and RREQ message formats to record sending time and Route Stability Factor (RSF) respectively. The new field added to HELLO

messages records the sending time. Now the receiver of the HELLO message makes use of that sending time along with its reception time to calculate the delay of the HELLO message. Then the average delay of the HELLO messages between two given nodes is calculated. That average delay is then used to calculate a route stability factor that is appended to the RREQ before forwarding it. The stability factor reflects the delay fluctuation of HELLO messages. The smaller the stability factor, the more fixed is the distance between the communicating nodes and the more stable is the path between them. Upon reception of the RREQ, the destination selects the path with the smallest sum of stability factors to send the RREP. The destination node has to wait a while in order to receive more RREQ and gather more statistics for the selection of the most stable path to the source. The simulation, under the condition of increasing node mobility and increasing traffic load, showed that MAODV outperforms conventional AODV in terms of packet delivery ratio, normalized route overhead, and route discovery frequency. In the paper though, they did not compare the delay. Logically, the waiting time for route stability computations should incur some delays. The authors claim the increase in delay, in comparison with conventional AODV, to be negligible; which is plausible since using more stable routes reduces the probability of broken links and retrials. Thus, the time they lose in different computations, they are essentially gaining it back with more "one-shot" transmissions. Another aspect they did not address explicitly in the paper is the case when an intermediate node does have a route to the destination. In that case, it should generate an RREP and send it based on route stability. Overall, even though, at first glance, this modified AODV idea presents a feature that might be of interest in exploring in the reactive part of RGR, the improvements presented are quite weak, especially for the PDR where the improvement is around 0.5% or less. This is not very surprising since the very idea of using HELLO messages delay as a foundation to measure route stability is not very promising. As a matter of fact, since the radio signals travel at the speed of light, the difference between any two HELLOs delay is hardly perceptible unless the traveled distance is really high, which will almost never be the case, given that a transmission range of 250 m was used.

Zhao Qiang and Zhu Hongbo introduced the Optimized AODV (O-AODV) in [20]. This version of AODV is also based on route stability. The novelty in this protocol lies in a new mechanism to process RREQs. The concept of *reliable distance* is introduced. In brief, the decision to drop or forward a RREQ packet is contingent to whether or not a defined condition

on the *reliable distance* is met. The authors claim the routes to be more stable; and this should result in less data packets being dropped compared to conventional AODV. As a matter of fact, simulations showed a 2% PDR increase over conventional AODV. Moreover, unlike most of the previously discussed strategies, O-AODV does not add to the routing overhead, nor does it explicitly add to the end-to-end delay. In fact, this strategy showed a drop in both overhead and end-to-end delay while exhibiting the highest increase in PDR of all the reviewed strategies so far; which makes it our best candidate to integrate into RGR in order to enhance the latter. More details on this O-AODV strategy are presented in Chapter 3 where we present the enhancements to the RGR protocol.

b. Improvements on GGF

The major problem with GGF is, as already pointed out, when we encounter a *void node* situation. When that happens, GGF is said to have failed and the packet is dropped unless there is a recovery strategy in place. In [4], Maghsoudlou et al. review the existing recovery strategies and propose a recovery strategy of their own. The reviewed recovery strategies include: Face Routing [21], face routing variations, and other strategies divided into 5 groups. The proposed recovery strategy in [4] is the Randomized Geographic Forwarding (RGF). The following paragraphs explain some of these strategies.

Based on geographic planar graphs, Face Routing consists of forwarding a packet along the interiors of a sequence of adjacent faces that are intersected by the straight line *SD* connecting the source node *S* with the destination node *D* (Figure 2.1). The adjacent faces must provide progress towards the destination node *D* as does the sequence *F*1*, F*2*, F*3*, F*4 shown in Figure 2.1. Face traversal is done in a localized way by applying the right hand rule (or left hand rule), i.e. a packet is forwarded along the next edge clockwise (counter-clockwise) from the edge it arrived from [22]. Note that Figure 2.1 only shows the procedure of Face Routing in general and not exactly in the context of it (Face Routing) being used as a recovery strategy to a *void node* situation in GGF. We would have had Face Routing being executed in a *void node* situation had nodes v_1 and v_4 (that are the only neighbors to source node *S*) been stretched to the position

highlighted in red for example. Note that in the graph, nodes that are not neighbors are not connected by a direct line between them. *S* and v_3 for example are not neighbors.

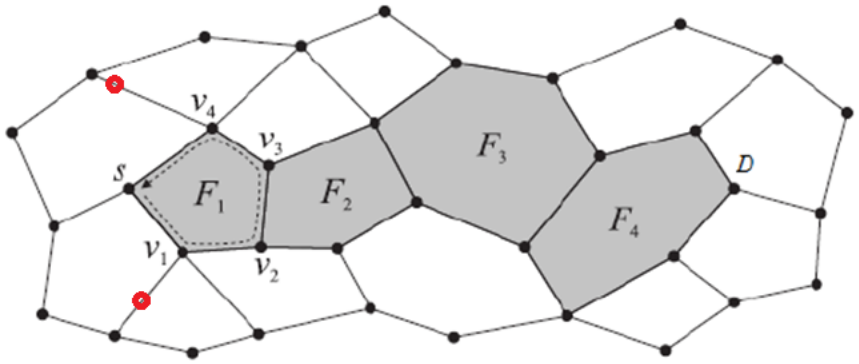

Figure 2.1: Explored sequence of faces towards the destination [23]

As discussed in [4], there is a time in the face routing algorithm to make a decision as to when a face traversal must be interrupted and what new face must be explored next. And this decision has been implemented differently by different variations of face routing. These variations are: Greedy-Face-Greedy (GFG) [24], Compass Routing II (Face-2) [25], Greedy Perimeter Stateless Routing (GPSR) [26], Greedy Other Adaptive Face Routing (GOAFR and GOAFR+) [27], and Greedy Path Vector Face Routing (GPVFR) [28].

The other strategies reviewed in [4] are divided into the following 5 groups. The first group is called Geometric Void Handling. Here, the geometric properties of nodes are exploited to identify topological structures called holes [29]. The second group is Flooding-based Void Handling where flooding, the simple form of communication, is used when a packet gets stuck at a *void node*. In the third category, Cost-Based Void Handling resolves the problem of voids based on the cost assigned to the network nodes. Heuristic Void Handling and Hybrid Void Handling are the other two groups.

From the perspective of UAANETs, the reviewed strategies/protocols present a few shortcomings. First of all, they are studied in a static network. There is no clear understanding of their behaviour when the topology of the network changes due to mobility, while a packet is being forwarded. Besides, planar graph based techniques are not applicable to 3D networks.

Finally, in addition to high overhead, some of the reviewed protocols have high complexity (they are difficult to implement), or require extra resources or complex processing. In an effort to design a light-weight, low-cost geographic routing protocol that works well in both 2D and 3D coordinate systems and in the presence of node mobility in a 3D space, the authors in [4] proposed Randomized Geographic Forwarding. In a nutshell, RGF proposes to overcome GGF failure by randomly picking one neighbor to forward the data packet to. The neighbors are either equally likely to be selected, or they are assigned different likelihood weights (probabilities) based on their distance to the destination node.

As we have now reviewed some routing protocols related to the work presented in this thesis, we are now going to go through the literature on existing MANET mobility models as proposing a mobility model for UAANETs/ MANETs is the other major part of our work.

2.3 Literature Review on Mobility Models for MANETs

Mobility models are a major part in the evaluation of routing protocols for airborne networks. Routing protocols are designed for specific types of applications. In order for a routing protocol to be effective and reliable, it needs to be tested with a realistic enough mobility model that captures well the behavior of the network in the application it is built for. The authors in [11], [30] showed through surveys that mobility models have a significant impact in the performance of the networking protocols. The choice of a mobility model is therefore critical. It is even more critical when we take into account the physical constraints (mechanical and aerodynamic) of UAVs. We already know that UAVs tend to maintain the same heading speed and change direction through making turns with large radii [10]; which is not the case for ground vehicles that can afford to make sudden stops, sharp turns, etc. Many mobility models already exist in the literature. Figure 2.2 shows a classification of the different mobility models available for MANETs.

Mobility models can be classified into four groups (Figure 2.2) based on their specific mobility characteristics. By mobility characteristics we mean, for example, how a node's movement at a given time interval is related or not to its previous movement at a previous time

interval. The classification has four groups: i) Random Models, ii) Models with Temporal Dependency, iii) Models with Spatial Dependency, and iv) Models with Geographic Restriction. The Random Models category includes models where the Mobile Nodes (MNs) move randomly and freely without or with very few restrictions. In these models, the destination, the speed, and the direction are chosen randomly and independently from other nodes and previous movement of the current node. The group of Models with Temporal Dependency contains those models where the current velocity of a mobile node depends on its previous velocity. The notion of time slots is introduced, and the velocities of an MN at different time slots are said to be "correlated" [30]. This temporal dependency stems from the fact that in real life, the movement of an UAV is constrained and limited by physical laws of acceleration, velocity, and the rate of change of direction. In the Models with Spatial Dependency, a node's movement is influenced by the movement of other nodes. The velocities of different nodes are correlated in space; for example in a freeway where the speed of a vehicle cannot exceed the speed of the vehicle ahead of it, otherwise we will have a collision. Finally, in Models with Geographic Restriction, paths and obstacles are integrated.

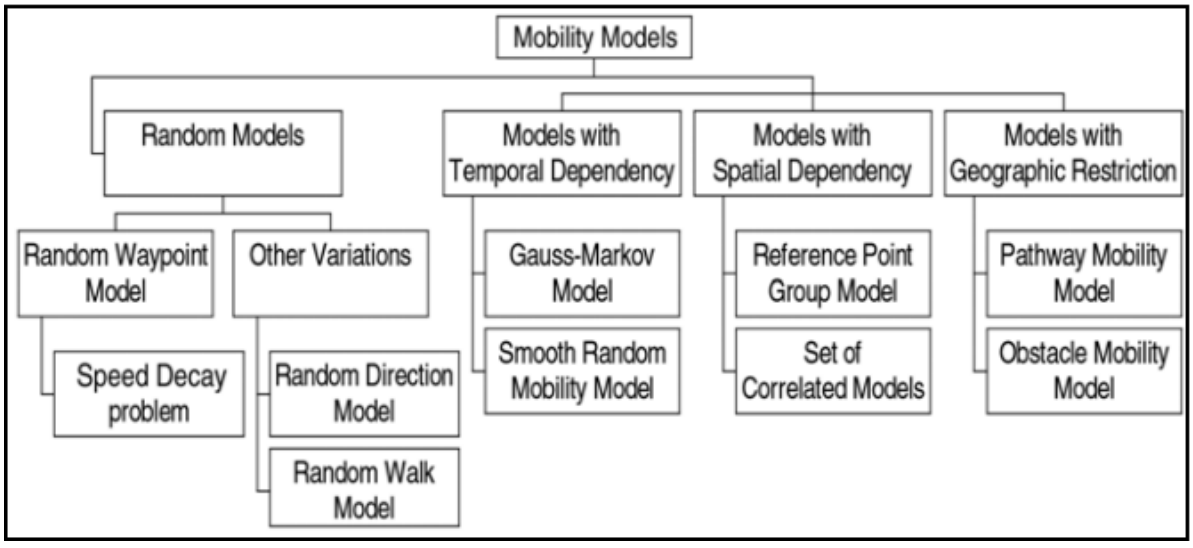

Figure 2.2: The categories of mobility models in MANETs [30]

In the next subsections, we review four existing mobility models. Some of the mobility models may fit in more than one of the groups presented above.

2.3.1 The Random Waypoint (RWP) Model

The RWP model was proposed in [31], [32]. It is now frequently used for simulations in MANETs mainly because of its relative simplicity and wide availability in simulators like ns-2[33], ns-3[34], and OPNET [35]. The way it works is as follows. A node randomly picks a location within the simulation area and moves to that location in a straight line, using a randomly chosen speed. Upon arrival at that location, the node pauses and picks another location and speed. When the pause time is set to 0, the node never stops until the simulation is over; it keeps randomly picking a new location to move to without pausing. In one way or another, the node is subject to sudden stops, sudden accelerations, and sudden speed changes. The same applies to direction, where a node can suddenly make a 180 degree turn. Figure 2.3 shows the simulation trajectory trace of an MN under RWP in a 1000 m X 1000 m area.

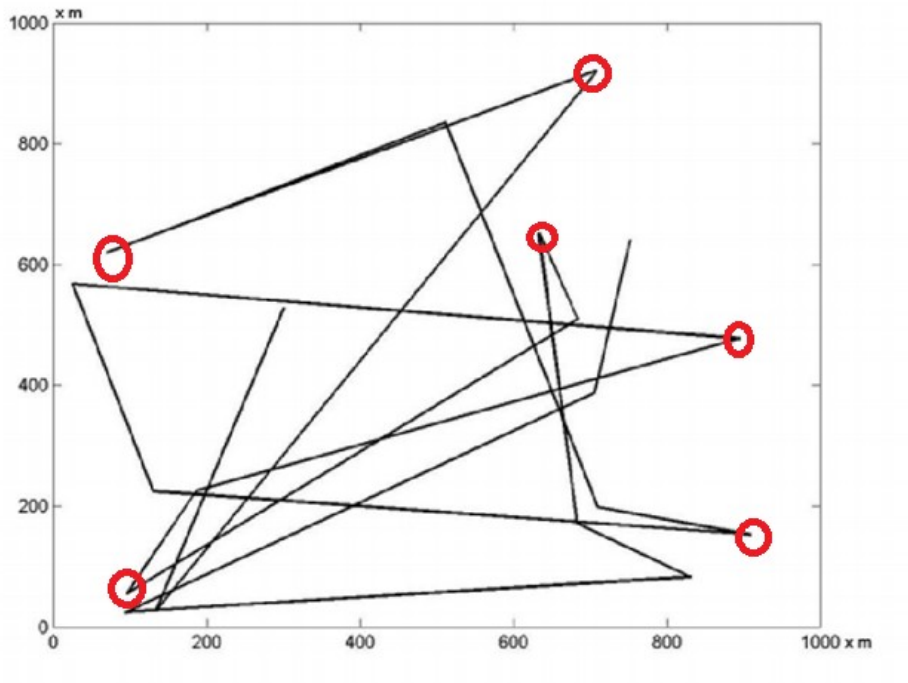

Figure 2.3: Trajectory of a UAV under Random waypoint [36]

As mentioned earlier, airborne vehicles have certain mechanical and aerodynamic constraints that prevent them from making sharp turns like the ones (some of them) highlighted in red circles in Figure 2.3. Obviously, despite the widespread usage of this mobility model, it is definitely not very realistic.

2.3.2 The Random Direction (RD) Model

The RD model is a variant of the RWP model. RD [30] was proposed in order to overcome a certain limitation of RWP. In fact, in RWP, the nodes tend to cluster around the center of the region and move away from the boundaries. This creates a non-uniform spatial node distribution and density wave problem stemming from the fact that the distribution of movement angle is not uniform in RWP as shown in [36]-[38]. In order to solve this, RD works as follows: instead of picking a random position like in RWP, a mobile node randomly (uniform distribution) chooses a direction by which to move. It moves in that direction until it hits a boundary of the simulation region. It then stops for a pause time before choosing another direction. Similar to RWP, if pause time is set to zero, the mobile node never pauses until the simulation is over. In a variant of RD, the MNs stop and choose a new direction a little before they reach the boundary. Figure 2.4 illustrates that. As we can see, the direction changes (some of the most radical of them are highlighted in red circles in Figure 2.4) occur only at the vicinity of a boundary. The figure presents the trajectories of many MNs, each represented by a different color. In [40], the authors claimed that RD results in less fluctuation in node density than RWP.

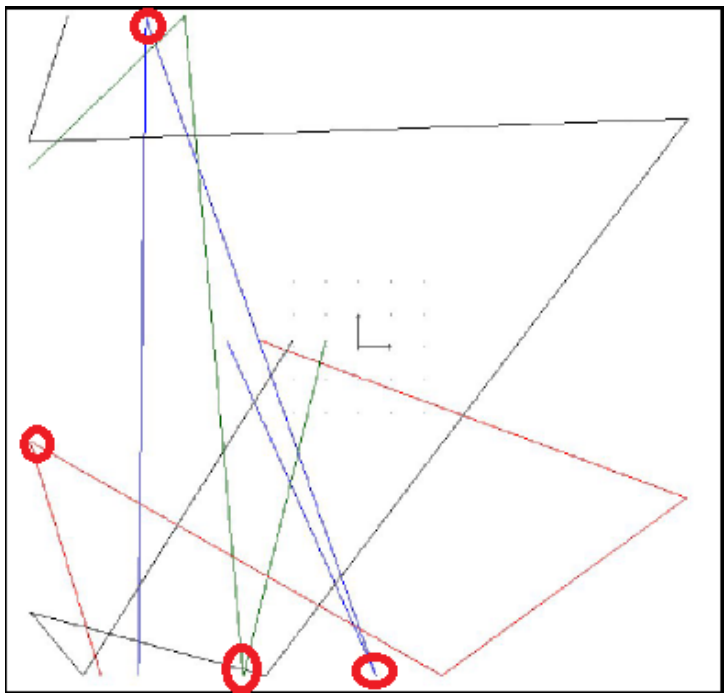

Figure 2.4: Random Direction [39]

2.3.3 The Smooth-Turn (ST) Model

ST is proposed in [10], [12], [13]. ST is similar to RD when it comes to the flexibility of trajectories. The idea is as follows: a MN randomly selects a point along the line perpendicular to its heading direction and circles around that point for a time period. When the time period is over, the MN chooses another center the same way as previously and circles around it for another time period most likely different from the previous time period. The perpendicularity is there to ensure the smoothness of the trajectories. In a nutshell, the MN continuously moves in circular arcs. The choice of the centers is random as well as the choice of the duration of the movement along the corresponding circular arc. However, the quality of randomness is different. For the time periods, the distribution is exponential in order to make them memory-less i.e. the timing of the center change does not depend on the duration for which the MN has maintained its current center. Those time periods during which a node is circling around a given center are also referred to as waiting times. On the other hand, for the selection of the centers, the authors

modelled the inverse length of the turning radius to be a Gaussian distribution. That way, they ensure that straight lines and slight turns are favored over very sharp curvy turns. With a very large radius, the circular arc looks like a straight line. At the boundary, they adopt a "reflection" model that simply consists of mirroring the out-of-region trajectory against that boundary as depicted in Figure 2.5. Note also the green points that represent the centers of the circular arcs. Figure 2.6 shows a UAV simulation trajectory that we generated ourselves using the above-described ST mobility model. The turning centers are once again represented as green points. Note that those centers do not necessarily fall inside the simulation area, especially since large radii are chosen very often.

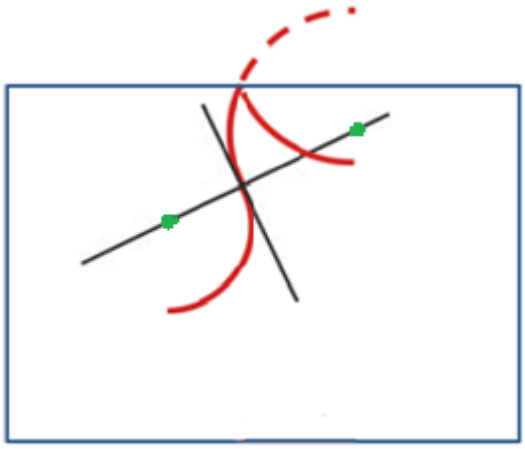

Figure 2.5: ST Model: Reflection at the boundary [12]

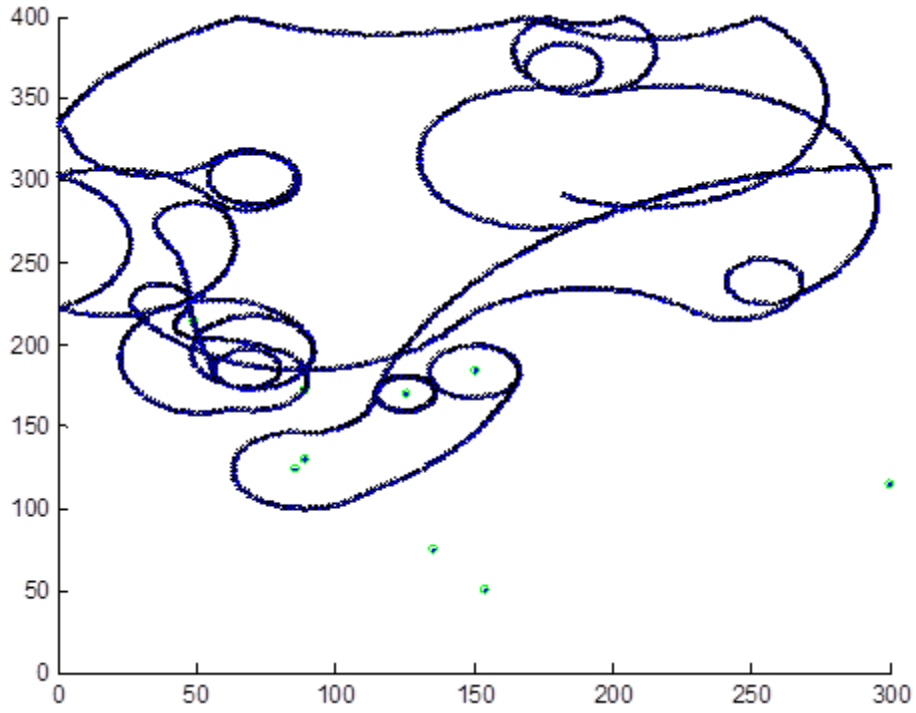

Figure 2.6: ST model trajectory

2.3.4 The Gauss-Markov (GM) Model

GM too, like ST, is motivated by the need to have a model that is closer to reality in the sense that a node, for instance, would accelerate, decelerate, or turn progressively. The model was proposed by Liang and Haas [41]. The current movement of a node (speed and direction) is related to the previous movement. The model is therefore said to feature temporal dependency (see Figure 2.2). At a pre-set instant *t*, the direction and speed of a given node are calculated. The MN moves with that direction and speed for a constant time interval of *T*. After *T*, the speed and direction are calculated again (at time *t*+*T*, then *t*+2*T*, then *t*+3*T*, etc.). The movement (speed + direction) calculated at a defined instant is related to the previously calculated/used movement. The relation is as follows [13], [36], [41]:

$$s_t = \alpha s_{t-1} + (1-\alpha)\overline{s} + \sqrt{1-\alpha^2}s_{x_{t-1}} \tag{2.3}$$

$$d_t = \alpha d_{t-1} + (1-\alpha)\overline{d} + \sqrt{1-\alpha^2}d_{x_{t-1}} \tag{2.4}$$

where s_t and d_t are respectively the speed and direction of the node at period time *t*. Likewise, s_{t-1} and d_{t-1} correspond to time period *t*-1. α, which varies in the range [0, 1], is a memory level parameter that reflects the degree of randomness. When $\alpha = 0$, the model is memoryless, whereas $\alpha = 1$ indicates the highest level of memory and the movement at time slot *t* is exactly the same as at the previous time slot *t*-1. $\overline{s}$ and $\overline{d}$ represent the mean speed and mean direction respectively when $t \to \infty$. Finally, $s_{x_{t-1}}$ and $d_{x_{t-1}}$ are random variables from Gaussian distributions denoted by $N(0, \sigma^2)$; σ being the standard deviation (speed or direction) when $t \to \infty$. Figure 2.7 shows an example of a GM trajectory in a 2D area.

The problem now resides in the behaviour of the model at the boundaries. In the scenario they chose in [42], they did not make things clear in that regard. In [30], Bai and Helmy suggest a direction change at the boundary without specifying exactly how this is done. In [43], Amoussou et al. propose a 180° turn. When the next position of the MN is calculated by means of the calculated speed and direction, if that position falls out of the region, then the MN pauses and makes a 180° turn before choosing another direction from a certain interval. This strategy is problematic at more than one count. The 180° sharp turn goes against one of the main reason why the GM model was proposed in the first place; which was to avoid sharp turns. Also, the very temporal dependency of GM is lost here since the next direction has nothing to do with the previous one in this case. Even though this sharp turn only happens at the boundaries, it is still too extreme. Given that the MNs are expected to move towards the boundaries quite often, it is necessary to have a strategy that influences the MN's movement so that we minimize the chances of ending up in a situation where the calculated next position falls out of the region. This should be done while keeping a taste of temporal dependency of the GM model essence. In line with this need, Alenazi et al. [31] propose a strategy to force the MNs away from the simulation boundaries. They define a buffer zone as shown in Figure 2.8.

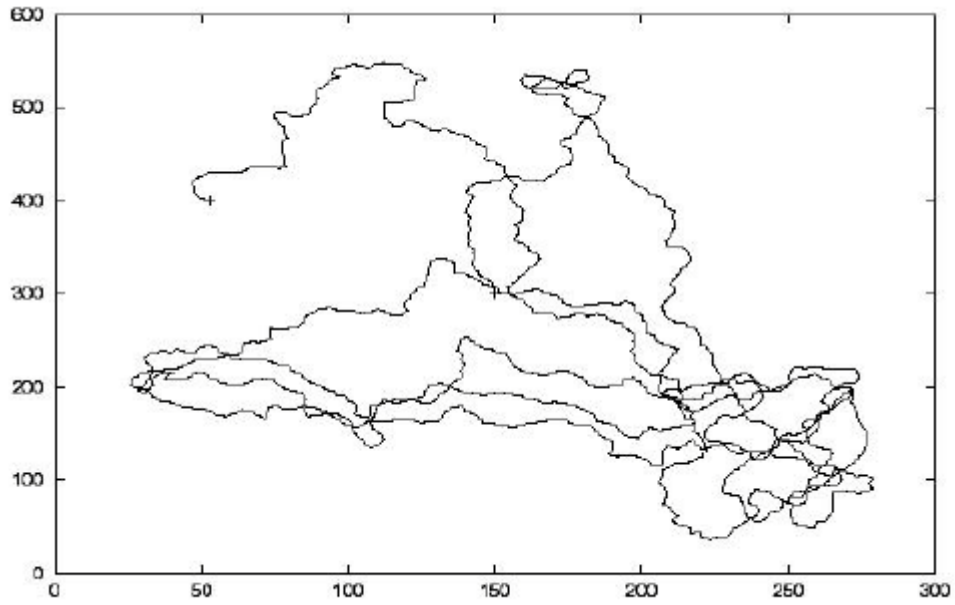

Figure 2.7: Mobile node trajectory with Gauss-Markov mobility model [42]

When the MN enters that zone, the mean direction presented in Equation 2.4 is changed so that the node is progressively pushed toward the center of the region. Yet, they did not specify what the value of that mean direction was under normal circumstances (inner zone shown in Figure 2.8). In fact, as far as we are concerned, the very notion of mean direction is ill-defined (Equation 2.4). We can decide on a mean speed within a certain range, yet it is a little unclear to define a suitable mean direction. What would be the value of the mean direction under normal conditions? 0°, 45°, 90°, etc.? It is difficult to make that decision. For example, if we set a mean direction of 45°, the MNs will have a tendency of moving along the first diagonal of the region (rectangular region), even though this can be dampened by playing with the α factor.

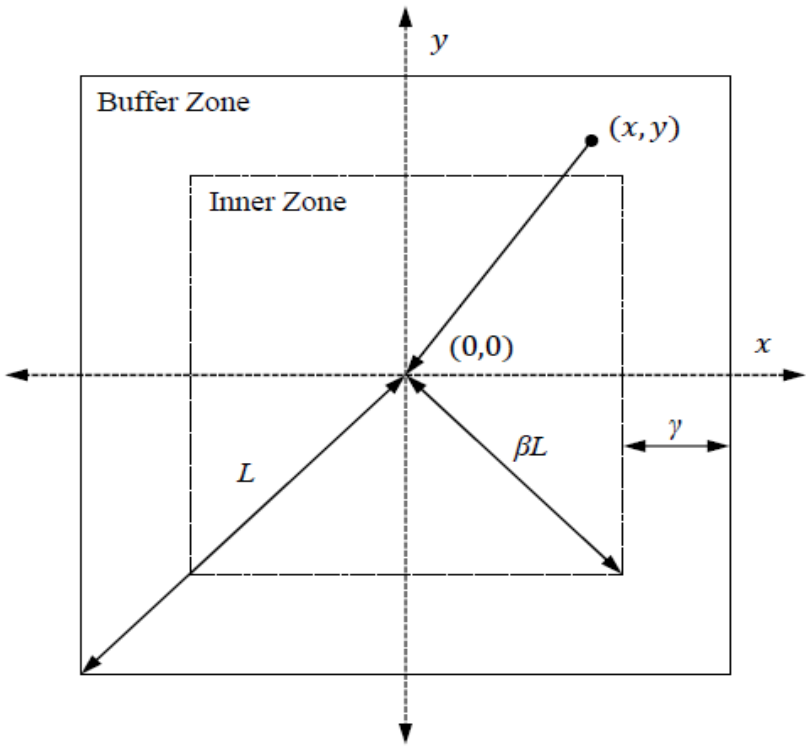

Figure 2.8: Buffer zone for GM [31]

In conclusion, most mobility models that exist suffer one or more of the following shortcomings. They are unrealistic in the sense that they feature movement patterns (such as sudden stops or sharp turns for example) that are definitely not possible with real UAVs due to mechanical and aerodynamic limitations. They make use of notions that are not very clear such as the notion of mean direction in the GM model. They lack a realistic strategy for the boundaries, with some of them recommending a sharp 180° turn when a boundary is hit. They do not seem to cover the entire region very well, even though not all missions require the full coverage of a given region. All these shortcomings motivated us to propose/develop a new mobility model for UAANETs simulations. This mobility model is presented in Chapter 4.

2.4 Summary

In this chapter, we have reviewed the RGR protocol that we are proposing to enhance in this research. Targeting the aforementioned enhancement, we have reviewed some ideas that had been proposed to improve the AODV protocol that constitutes the Reactive part of RGR. We have also reviewed some ideas that target the improvement of the GGF protocol that constitutes the Greedy part of RGR. Most of the explored ideas present shortcomings of some sort that we have pointed out. All this has led us to propose some new ideas with the goal of enhancing RGR both in its Reactive part and in its Greedy Forwarding part. This will be covered in Chapter 3. We also went through mobility models as it appears that the widely used RWP model is not very realistic. We have seen some problems with the mobility models reviewed in the literature and we are proposing a new model in Chapter 4 to address those problems as much as possible.

Chapter 3 : Enhancements to Reactive-Greedy-Reactive Routing Protocol

3.1 Introduction

Both AODV and RGR were discussed in the literature review presented in Chapter 2. As we saw, the RGR routing protocol was specifically designed for UAANETs. The Route Discovery part of the protocol is inherited from AODV. Potential routes are characterized by their freshness and their length. Freshness is represented by a sequence number, whereas the length is represented by a hop count. Whenever an IN has a route to a given destination in its routing table, if it receives an RREP that is destined to another node, and its content informs that IN of another route to the same destination, the IN will update its routing table if the new route is fresher. If the freshness is the same, the update will only occur if the new route is shorter in hop-length. More precisely, the node updates the route table entry if at least one of the following occurs [7]:

- The sequence number is marked as invalid in the route table entry for this destination.
- The destination sequence number in the RREP is greater than the node's copy of the sequence number.
- The sequence numbers are the same but the route is marked as inactive.
- The sequence numbers are the same and the new hop count is smaller than the hop count in the route table entry.

Note that when a route to a destination is created upon reception of a HELLO packet or a RREP issued by (not forwarded through) that destination, the created route entry has its sequence number marked as valid. On the other hand, the created route entry has its sequence number

marked as invalid if the route (generally one-hop) to that destination was created upon reception by the node of a control packet (such as a RREQ for example) from that very destination (acting as the previous hop). Also note that declaring sequence numbers as valid/invalid is different from actually declaring route as valid/invalid. For example, a route can be valid with its associated destination sequence number declared as invalid (see above). The declaring of sequence numbers as valid/invalid is done in order to distinguish how a route is created or learnt of, whereas the declaring of routes as valid/invalid has to do with broken links, timeout expiration, etc.

In summary, the intent is to replace an existing route with a fresher and/or shorter one. The fact that some links in that new route might be at the brink of breaking is ignored. This results in (potentially) very short-lived routes and considerable route breakage frequency. This route breakage frequency can be lowered if we ensure that the routes that are created and put in the routing tables are more reliable/stable. Therefore, adding a reliability criterion in the route selection/construction process of RGR should produce more stable routes; not only fresher and/or shorter ones. We then expect less frequent route breakage (which can be measured by the number of Route Errors), lower route discovery overheads and ultimately a higher PDR. Section 3.2.1 describes how the stability criterion was added to the protocol.

In UAANETs as well as in MANETs, geographic routing is widely used. Geographic routing relies on GGF. GGF fails when a packet arrives at a node that has no neighbor closer to the destination than it is. The node in this situation is referred to as a *void node*. In Chapter 2, we reviewed a few strategies (from the literature) that salvage packets in *void node* situations. Now, from the perspective of UAANETs, the reviewed strategies/protocols present a few shortcomings. First of all, they are studied in a static network. There is no clear understanding of their behaviour when the topology of the network changes due to mobility while a packet is being forwarded. Besides, planar graph based techniques are not applicable to 3D networks. Note that even though we focus on 2D in this work (mostly for the sake of simplicity), we propose enhancements, algorithms, or mechanisms that are seamlessly extendable to 3D since, ultimately, real missions are conducted in 3D scenarios/environments. Therefore, techniques such as the planar graph based ones that are not extendable to 3D are problematic for us. Finally, in addition to high overhead, some of the reviewed protocols have high complexity in that they are difficult to implement, or require extra resources or complex processing.

Similar to the work in [4], we propose a light-weight, low-cost recovery strategy. Yet, instead of randomly picking a neighbor, we propose to make a more informed/deterministic decision. We distance ourselves from randomly selecting a neighbor mainly because we want to make sure that the selected neighbor that the packet is forwarded to is our best guess at actually moving the packet toward the destination. Section 3.2.2 introduces three strategies to salvage packets in void node situations that are applicable to UAANETs.

3.2 RGR Enhancements

In this section, we present the two enhancements we brought to RGR. The first one is adding the stability criterion into the route construction process. This results in a version of RGR we named Modified-RGR. The second enhancement is the strategy to recover from a *void node* situation in Geographic routing in general. We then integrate this strategy into the GGF part of Modified-RGR; we call this version of RGR Optimized-RGR.

3.2.1 Introduction of a Stability Criterion

For the purpose of constructing more stable routes, we could have made use of some ideas presented in Chapter 2, especially two of them: the use of RRI presented in [17] and the use of RSF presented in [19]. The main problem with RRI is that it incurs considerable additional overhead as shown in the experiments presented in [17]. On the other hand, the problem with RSF is that the demonstrated improvements are quite weak, especially for the PDR where the improvement is around 0.5% or less. Finally, we opt for the concept of *reliable distance* (also briefly reviewed in Chapter 2) in order to define our stability criterion. This concept seems more promising as we see next.

As already mentioned, the concept of *reliable distance* was introduced by Qiang and Hongbo in [20] and it confers more stability to the constructed routes. In RGR, unlike AODV for instance, the RREQ packet piggybacks some additional information about the last node that sent it. It can be the originator of the RREQ or an intermediate node. This additional information

includes: the position, the direction, the velocity, and the timestamp at the time when the RREQ was sent. The link stability is taken into account for route construction when processing RREQs using the algorithm proposed in [20]. The algorithm has three steps. The node receiving the RREQ is referred to as *D* and the node the RREQ is received from is referred to as *S*. t_0 is the time when the RREQ is sent by *S*, and t_1 is the time when the RREQ is received by *D*. The *reliable distance* is represented by *r*. This concept is introduced to measure the reliability of the link state between *S* and *D*. The three steps described below (see Figure 3.1 for an illustration).

Step 1: Node *D* determines *S'*, the position of *S* at time t_1 (current time) using the information contained in the RREQ packet. It is assumed that direction and velocity do not change between t_0 and t_1. The position is calculated using Equations (3.1) and (3.2).

$$x_{t_1} = x_{t_0} + v \times \Delta t \times \cos\alpha \tag{3.1}$$

$$y_{t_1} = y_{t_0} + v \times \Delta t \times \sin\alpha \tag{3.2}$$

where $\Delta t = t_1 - t_0$.

Step 2: Node *D* determines its own position at time t_0. This position at time t_0 is also referred to as position *D* (same name as the node's name). The position of *D* at the current time t_1 is denoted by *D'*. Node *D* is able to determine its own position (at t_0 or t_1) by means of its own GPS information about speed and direction. At this point, the initial distance between the nodes *S* and *D* (*d=SD*) at time t_0 and the final distance between them (*d'=S'D'*) at time t_1 can be calculated.

Step 3: If *d' > d* and *d > r*, then it is determined that the two nodes are moving in opposite directions and are already far apart, therefore the RREQ is discarded. Otherwise, we proceed as in the existing RGR protocol with RREQ forwarding or RREP generation among others. *r*, the *reliable distance,* is calculated using Equation (3.3).

$$r = R - w \times V'_{DS} \tag{3.3}$$

where *R* is the node's transmission range, *w* is a constant, and V'_{DS} is the projection onto *S'D'* of the relative velocity between *S* and *D*. See Figure 3.1 for an illustration.

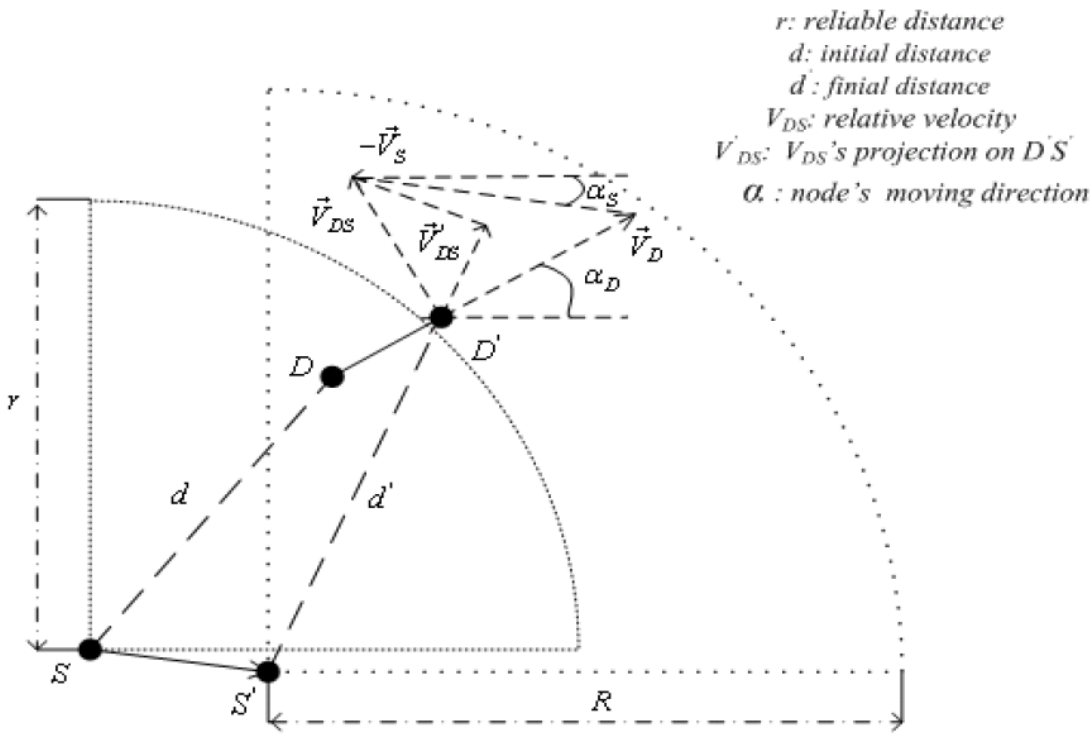

Figure 3.1: Reliable route construction in OAODV [20]

The *reliable distance* should be carefully selected. In fact, too short a *reliable distance* can lead to many RREQs being discarded, and therefore more route discovery retries, which would add to the overhead. This is part of the reason why it is not a constant value but varies according to the relative speed of the nodes. The resulting routes are more stable, therefore we expect to have fewer data packet drops; and this should yield higher PDR than RGR. In addition, this mechanism does not explicitly add to the end-to-end delay and therefore, we do not expect an increase. By implementing this mechanism within the existing RGR protocol, we obtain a version of the latter that we call Modified-RGR.

3.2.2 Recovery Strategy for GGF

In order to rescue data packets and give them more chances to be forwarded when GGF has failed, we are presenting three strategies. Depending on their performance (in simulation), we

will pick the best one and make it our final Recovery Strategy. The choice will be based on simulation results because we do not have enough ground to prefer one strategy over the other two at this point. All three strategies have in common that they feature holding on to the packet at some point by the forwarding node. The motivation behind holding on to the packet is to allow the nodes to move around a little bit with potentially some new nodes entering into the local neighborhood that have a better chance to reach the Destination Node. This applies also to current neighbors that might move to more advantageous positions after the hold-on period. The difference between the three strategies lies in the choice of the neighbor to forward the packet to. At this point, the strategies do not feature a loop prevention mechanism, but it is expected that loops, if they ever occur, will not last as the hold-on period is meant to allow a small change in the configuration of the forwarding node's neighborhood. As we mentioned, by the time the hold-on period is over, there might be additional or better candidates to forward the packet to.

a. Strategy 1: Retry GGF

In Strategy 1, after holding on to the packet, the forwarding node simply retries GGF by looking for a neighbor that would be closer to the destination than it (the FN) is. The idea in this strategy is: instead of dropping the packet when GGF fails, just wait and retry once. If GGF fails again after the hold-on period, the packet is dropped. The limit to one hold-on period comes from the fact that we want to avoid generating large delays for packets that make it to the destination by means of this recovery strategy. The hold-on period has been chosen to be 2 seconds. This value has been selected after trying a few values of the same order. With the nodes moving at [50-60] m/s, in 2 seconds all the nodes will have covered 100 to 120 meters each if no sudden change of direction occurs. Therefore, there is a high probability that the FN might have neighbors (new or existing) that have moved closer to the DN. And as the HELLO interval is typically 1 second, the node may have learned about such new neighbors. As for the new positions of existing neighbors, even in the absence of updates via HELLO messages, a node will use mobility prediction to estimate the neighbor's new position after this period. When a suitable neighbor is found in the second attempt, the packets that would have been dropped at GGF failure (when no recovery strategy exists) now end up being forwarded and may eventually reach the destination; which will increase the PDR. The increase in PDR is expected to come at the

cost of higher average end-to-end delay since we had explicitly delayed the additional packets for a hold-on period at least once in their (multiple-hops) route.

b. Strategy 2: Forward to the Furthest Neighbor

In Strategy 2, after holding on to the packet when GGF fails the first time, the FN, like in Strategy 1, first retries GGF, and if it fails again, then the packet is instantly forwarded to its furthest neighbor. The distance to the DN does not matter anymore at this point. For example, if the FN has a neighbor at 100 m from itself, another one at 200 m, and a last one at 600 m, the data packet will be forwarded to the neighbor at 600 m notwithstanding the distances to the DN. The idea of forwarding the packets to the furthest neighbor when GGF fails the second time is motivated by the following. Since the FN has no valid route to the destination nor does it have a neighbor that is closer to the DN than itself, the dynamics of the network have probably caused the FN to end up in the "wrong area", and the way to escape from it faster is to move the packet as far away as possible from the current location; and the furthest neighbor seems to be the logical choice for that purpose. For this strategy, we selected the hold-on period to be 1 second. Like in Strategy 1, this value is the result of testing a few values that allow the nodes to travel reasonable distances before forwarding retrial. In this strategy, we also expect to achieve a higher PDR than in Modified-RGR since we are giving the data packet more chances to make it to the destination. However, this should also come at the expense of an increased end-to-end delay. When there is no furthest neighbor to forward the packet to after the hold-on period, the packet is dropped. As in Strategy 1, limiting the number of hold-on per FN to 1 is motivated by the need to avoid huge packet delays. Note that, instead of dropping the packet after the second GGF failure as in Strategy 1, the packet is now given a third chance through the furthest node even though we use a hold-on time of 1 second instead of 2 seconds as in Strategy 1. In fact, as pointed out earlier, the value of "1 second" was chosen after trying a few other values including "2 seconds", and we found that Strategy 2 performs better with "1 second" than with "2 seconds"; just as we determined before that Strategy 1 performs better with "2 seconds" than with "1 second". In brief, Strategy 2 performs better with "1 second" than with "2 seconds", and Strategy 1 performs best with "2 seconds". Now, knowing that Strategy 2 with "2 seconds"

(which is further outperformed by "1 second") offers one more chance to the packet than Strategy 1, we can already expect Strategy 2 to outperform Strategy 1 in terms of PDR.

c. Strategy 3: Forward to the Best-Moving Node

The Best-Moving Node is the node, among the FN and all its neighbors, that is deemed to be the best candidate node moving towards the destination. In order to determine the BMN, the FN predicts its own coordinates, the coordinates of all its neighbors, and the coordinates of the DN 10 seconds from the current time. The predictions are made using the current speed and direction of the nodes. With these coordinates, the Euclidian distances from the FN and from the neighbors to the DN are calculated. The node associated with the smallest distance is dubbed the BMN. The BMN can be a neighbor or the FN itself. The choice of a 10 seconds prediction interval is a tradeoff in the sense that longer periods of time may result in better predictions in terms of who is moving to where. However, they are also more uncertain as we assume that neither speed nor direction will change during the prediction interval. Experimenting with a few values demonstrated that a value of 10 seconds provided consistently good results.

In Strategy 3, when GGF fails, the FN determines the BMN and forwards the data packet to it. In the case where the BMN happens to be the FN itself, the FN holds on to the packet for 1 second and then tries GGF again. If GGF fails again, the FN searches for the BMN again in order to forward the packet. If the BMN happens to be the FN for the second time, then the packet is dropped instead of holding on to it at the same FN for a second consecutive time. The reason for dropping the packet is the same as in Strategies 1 and 2: avoiding racking up a huge amount of packet delay. The 1 second value here is also the result of testing a few different values, and "1 second" gave the best results. Note that the 1 second hold-on time is unrelated to the 10 seconds prediction time. The hold-on period, as already mentioned, deals with allowing nodes to actually change location in hope for more advantageous positions toward the destination. On the other hand, the prediction time is a virtual view or an indication of where the nodes might be headed.

All three strategies were implemented on top of Modified-RGR to build three propositions of Optimized-RGR and the results are discussed in Chapter 5. Based on those results, we will pick the best of the three propositions and make it the final Optimized-RGR protocol. The strategy

used in that final version of Optimized-RGR will be, by the same token, our proposed recovery strategy for GGF failure that can be applied to any Geographic routing protocol independent of the RGR family of protocols.

3.3 Overview of the Implementation in OPNET

RGR was already available in OPNET. This includes the node model, process model, proto-C and C code of the model, finite state machine (FSM) model, etc. Our work consisted of changing and adding necessary C code (functions) and changing the FSM in order to implement the enhancements presented in the previous sections. Changing the FSM, for example, was necessary when implementing the hold-on periods. The results of the implementation and simulations are presented in Chapter 5.

Chapter 4 : The Enhanced Gauss-Markov Mobility Model

4.1 Introduction

Routing protocols are designed assuming certain application-specific network characteristics. In order for a routing protocol to be effective and reliable, it needs to be evaluated with a realistic mobility model. The Random Waypoint mobility model, widely used, allows a node to stop suddenly and turn sharply, and therefore, fails to capture the movement pattern of actual airborne vehicles. In this chapter, we propose the Enhanced Gauss-Markov mobility model, a realistic model for networks of UAVs (UAANETs) based on the Gauss-Markov mobility model [41]. EGM features mechanisms to eliminate/limit sudden stops and sharp turns within the simulation region. It is worth noting that in the GM model, the very notion of mean direction is poorly defined. We can decide on a mean speed within a certain range, but it is not obvious what the mean direction under normal conditions should be. For example, if we set a mean direction of 45°, the MNs will have a tendency of moving along the first diagonal of the region. For this reason, in our EGM model, as part of the mechanisms to eliminate/limit sudden stops and sharp turns, we introduce the notion of *direction deviation* instead. Finally, our model, unlike many others, explicitly features a mechanism that ensures smooth trajectories at the boundaries. In the next section, we describe how our proposed Enhanced Gauss-Markov mobility model works. At the end, we also show the trajectory of a UAV when our model is used.

4.2 Description of the Model

The EGM mobility model is based on the GM mobility model. The novelty here is that we compute the direction slightly differently from GM. Also, and most importantly, we implement a

mechanism for boundary avoidance. The model works as follows. At the beginning of the simulation, we randomly assign a direction and speed to each node. The speed is randomly picked from a uniform distribution from a range of [50, 60] m/s, which is typical for UAVs (see [2]). The direction is picked from a random uniform distribution in the range of [0°, 90°] as initially all nodes are at the bottom-left-hand corner of the region (see Figure 4.1). Alternatively, if the nodes were to start, say, in the centre of the simulation area, we could pick an initial direction in the range [0°, 360°]. Note that the origin of angles is the vertical line and that they are read (incremented) clockwise as illustrated in Figure 4.2. In this discussion, we consider the bearing of the UAV and the angle as interchangeable.

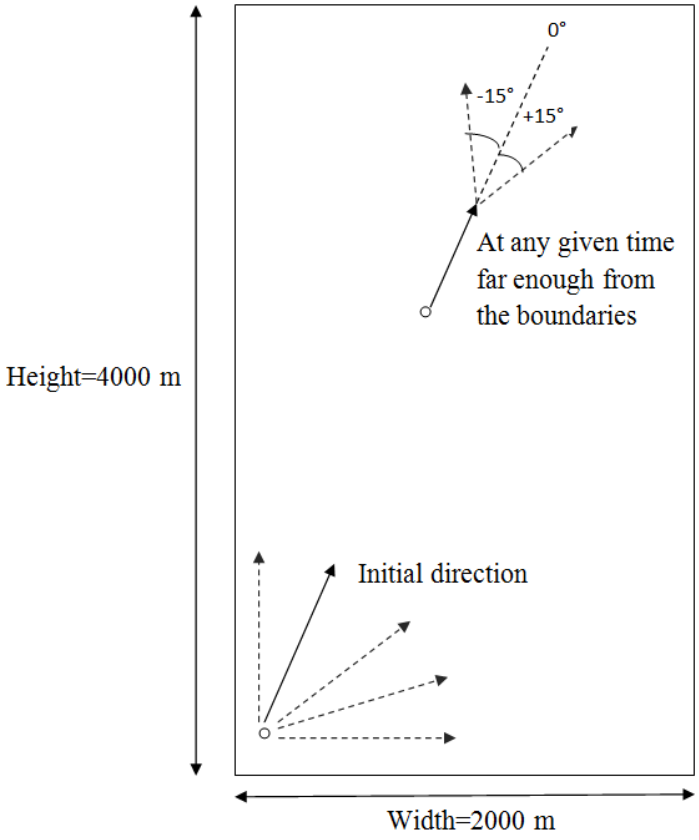

Figure 4.1: Direction deviation

In the GM model, the next speed and direction are calculated. In our model, we calculate the speed as in the GM model, but for the direction, we introduce the notion of direction deviation. We calculate the speed and the direction deviation as follows:

$$s_t = \alpha \times s_{t-1} + (1-\alpha)\overline{s} + \sqrt{1-\alpha^2}s_{x_{t-1}} \quad (4.1)$$

$$d\_dev_t = \alpha \times d\_dev_{t-1} + (1-\alpha)\overline{d\_dev} + \sqrt{1-\alpha^2}d_{dev_{x_{t-1}}} \quad (4.2)$$

where s_t and d_dev_t are respectively the speed and the direction deviation of the node at period time *t*. Likewise, s_{t-1} and d_dev_{t-1} correspond to the speed and direction deviation at period time *t-1*. α, which varies in the range [0, 1], reflects the degree of randomness. When $\alpha=0$, the model is memory-less, whereas $\alpha=1$ indicates the highest level of memory and the movement at time slot *t* is exactly the same as at the previous time slot *t-1*. $\overline{s}$ and $\overline{d\_dev}$ represent the mean speed and the mean direction deviation respectively when $t \to \infty$. Finally, $s_{x_{t-1}}$and $d\_dev_{x_{t-1}}$ are random variables from Gaussian distributions denoted by $N(0, \sigma^2)$; σ being the standard deviation when $t \to \infty$. After we calculate the next direction deviation, we then calculate the next direction as follows:

$$d_t = d_{t-1} + d\_dev_t \quad (4.3)$$

At the beginning of the simulation, the speed of an MN is chosen randomly (uniform distribution) from the range [50, 60] m/s. We do the same for the direction deviation, except here the range is [-10°, +10°]. We chose that range arbitrarily bearing in mind that we did not want to have too big a deviation. The order of 10° seems reasonable. One could also have chosen 15° or even 20° for that matter. In fact, we are going to see below that subsequent direction deviations under normal conditions (far enough from boundaries) will be expected within the [-15°, +15°] range. At the end of the day, these intervals or orders of intervals for direction deviations are chosen to ensure gradual deviations and avoid sharp turns, and can be set based on the physical characteristics of different UAVs that will ultimately determine how tight a turning radius they are capable of.

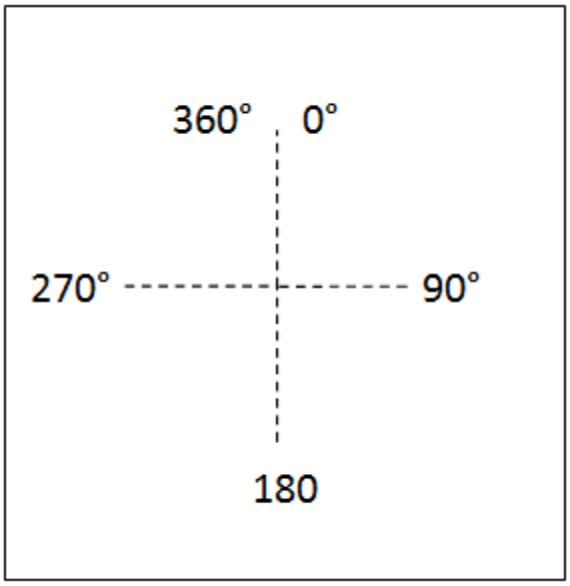

Figure 4.2: Direction angle

As we can see in Figure 4.1, at the end of a time period, the next computed deviation will most likely fall somewhere between [-15°, +15°]. The explanation for that is as follows. At the beginning of the simulation, the initial direction deviation is chosen randomly from the interval [-10°, +10°]. Therefore, when Equation (4.2) is used for the first time, d_dev_{t-1} is a value from [-10°, +10°], $\overline{d\_dev}$ is equal to 0 (center of the [-10°, +10°] interval), and $d\_dev_{x_{t-1}}$ is a Gaussian variable that falls into the interval [-10°, +10°] more than 99% of the time. Finally, when fractions (respectively α, $1-\alpha$, and $\sqrt{1-\alpha^2}$) of these three values are added together as in Equation (4.2), the end result is likely to fall inside the interval [-15°, +15°]. As an illustration for this, take one extreme case where d_dev_{t-1}, and $d\_dev_{x_{t-1}}$ are both equal to -10 (pretty unlikely case). With $\alpha = 0.86$, Equation (4.2) results in a direction deviation of about -14°.

At the end of a time period, we compute the next direction deviation when we are far enough from the boundaries as follows (see Equation (4.2)): we keep the mean direction deviation $\overline{d\_dev}$ at 0. We know that $\overline{d\_dev}$ is the mean direction deviation when $t \to \infty$. Since we want the calculated deviation to fall in the range [-15°, +15°], it follows that the mean value is 0, the center of that interval. We generate the variable $d\_dev_{x_{t-1}}$ from a Gaussian distribution of mean 0 and variance 6.2. The mean of the Gaussian distribution is always assumed to be 0. The

variance is chosen so that the Gaussian variable falls into the range [-10°, +10°]. In fact, it will fall in that interval more than 99% of the time. Only occasionally will it be outside that range. Then again, this Gaussian part of the equation is just a Gaussian noise that is dampened by a factor of $\sqrt{1-\alpha^2}$. This noise is actually there to add a controllable degree of randomness to the outcome of the equation. α is set at the beginning of the simulation to be equal to 0.86. We picked this value arbitrarily. The closer it is to 1, the more the previous deviation has an impact on the currently calculated one. Finally, we know d_dev_{t-1} and d_{t-1}, the previous direction deviation and the previous direction respectively. Thus, we have everything we need to apply Equations (4.2) and (4.3). To find the next speed, we resort to Equation (4.1). The mean speed $\bar{s}$ is set to 55 m/s. This is because we want speed values in the interval [50, 60] m/s and 55 is the center of that interval. The variable $s_{x_{t-1}}$ is generated from a Gaussian distribution of mean 0 and variance 1.54 in order for the Gaussian variable to fall in the range [-5, +5] m/s more than 99% of the time. Again, that Gaussian variable is just a Gaussian noise that is attenuated by a factor of $\sqrt{1-\alpha^2}$. The range [-5, +5] is chosen because the final speed interval has to be [50, 60], and in the extreme case where α is 0, we still want to stay in that interval when applying Equation (4.1). Yet, since we are dealing with random variables, occasionally the calculated speed can fall outside the [50, 60] range. When that happens, we force it to the closest value: 50 or 60. Note that we did not have to do this for the direction deviations because the intervals are not similarly constrained and we can afford a few values outside the specified interval. At the end of the day, the intervals for direction deviations are there just to ensure gradual deviations and avoid sharp turns. If the UAV characteristics were to enforce hard limits on the maximum turning radius, however, we could then similarly enforce max and min direction deviations.

Once we have determined the next speed and next direction, we can calculate the next (x, y) position as well:

$$x_{next} = x_{current} + s \times \sin(direction) \times time\_period \tag{4.4}$$

$$y_{next} = y_{current} + s \times \cos(direction) \times time\_period \tag{4.5}$$

As long as that next position falls within the boundaries of our simulation region, we continue computing the subsequent positions the very same way. However, when a next position threatens to fall outside the region, we need to take steps to ensure that the node does not move

outside the region. As a first step, we implemented a boundary-avoidance mechanism. As soon as a node gets within a certain distance margin *d=250 m* to a boundary, we change the mean direction deviation (see Equation (4.2)) from 0 to a value that forces the node to move back towards the center of the region progressively instead of going towards the boundary. We also change the Gaussian distribution accordingly. That way, as illustrated in Figure 4.3, instead of having our next direction in the red range, we will have it in the green range. The mean direction deviation depends on the "incidence angle". If we are moving towards the right boundary with a direction (bearing) in the range [0°, 90°] then we need a negative mean for the next deviation calculation. Likewise, when the current direction is in the range [90°, 180°] then we need a positive mean for the next direction deviation. We chose the means to be -22.5° and 22.5° respectively in our case. We also adjusted the variance of the Gaussian noise to fall within the range [-2.5°, +2.5°] in order to target the ranges [-25°, -20°] and [20°, 25°] respectively for the calculated direction deviation. Again, one could have chosen different values. More aggressive direction deviations would lead to less coverage of the boundary region, less aggressive ones would result in nodes moving towards and potentially beyond the boundary often (if not for the alternative mechanisms presented below). We did experiment with a couple of values and the ones we use are the ones that gave the most satisfactory results in terms of region coverage, as judged from the traces of the trajectories of the MNs. A good coverage shows a node that pretty much covers all the sectors of the region and also has smooth turnings close to or at the boundaries.

The boundary-avoidance mechanism, as depicted in Figure 4.3, forces the direction once, but for the next step the node might end up targeting an out-of-region position again. In order to avoid this situation, we repeat the process three times. This ensures that the nodes smoothly turn back in the direction of the center of the region (Figure 4.4). After those three repetitions, we return to the original equations. We do not want to force the direction more than three times as we want to "release" the constraint on the movement as soon as we are (reasonably) sure that the node is no longer headed to a boundary. This approach (forcing the direction three times) is obviously not the only one possible. In fact, another approach could have been to not limit the number (three), but instead keep forcing the direction until the node is back into the "safe area" (inside the dashed lines in Figure 4.3), and this could have required forcing the direction four, five, six, or even only one or two times. We chose to proceed with the fix number approach

mainly because the direction at which we “release” the node with (after forcing the direction a number of times) does matter a lot. It is not just about entering the “safe area”. For example, let us imagine that *P2* (Figure 4.3) falls inside the “safe area” and we “release” the movement direction (after only one forcing). The node, given its new direction, is now very likely to be headed to the upper boundary; which is not something we specially want to allow. This is one limit of that other approach. At the end of the day, forcing the direction three times gives the impression of a progressive U-turn towards the center region (or “safe area”) and we can even afford to release the movement before entering the “safe area” with the certitude of heading there as we can see with *P5* in Figure 4.4.

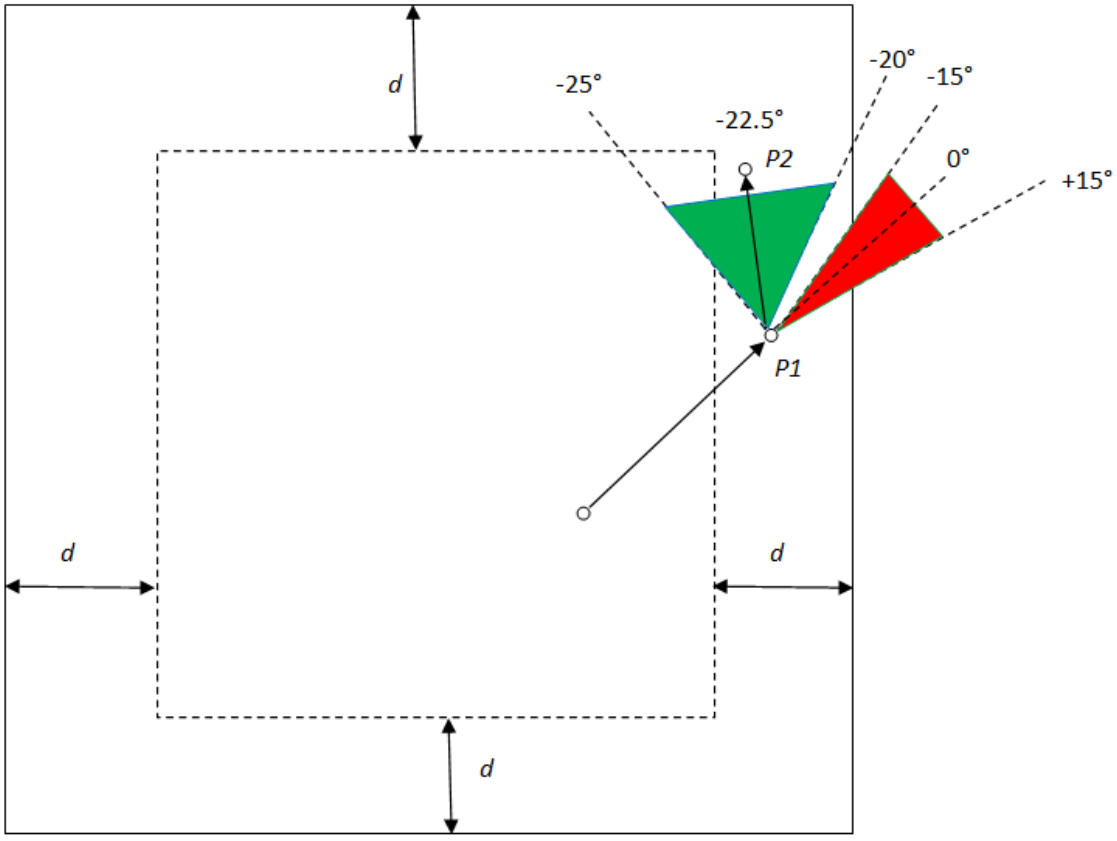

Figure 4.3: Direction distribution change

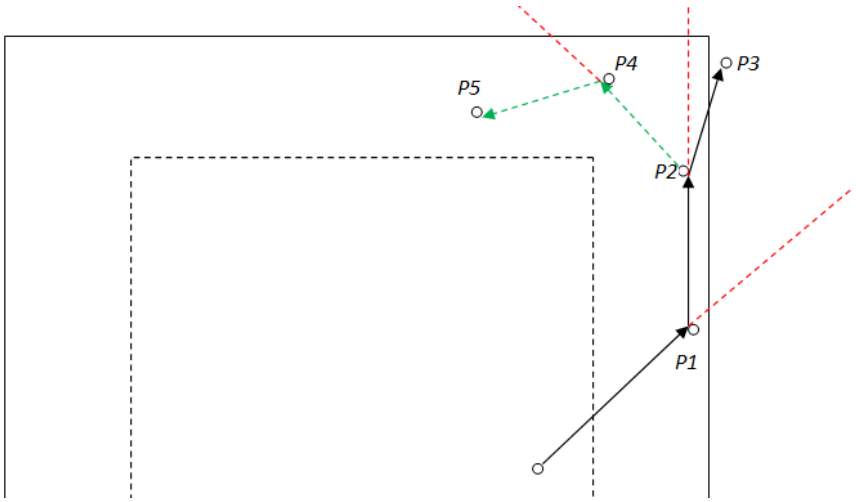

Figure 4.4: Boundary turning

While this process works well, in a few cases we still run the risk of generating trajectories that end up outside the area boundary even after having initiated the progressive turn. In these rare cases, we fall back on a more radical alternative to avoid leaving the simulation area. One of those cases is shown in Figure 4.5. In Figure 4.5, *P1* is our current position at the end of the previous time period. When the next position (*P2*) falls outside the region, we proceed as follows. We force the next position (*P3*) to be half way between the current position (*P1*) and the intersection (*I*) with the boundary. By doing so, while maintaining speed and direction as calculated, the time period is reduced just for that step. When we get to the end of the new time period (at position *P3*), we calculate the next speed and direction (with the original time period). Those will eventually lead us to position *P4*. Given that the general range of our direction deviation is [-15°, +15°], *P4* is likely to fall inside the region. When it does, bear in mind that the progressive U-turning described in Figure 4.4 will still be triggered if (which is very likely at this point) we are not in the "safe area". In the event that it falls outside the region again (*P4i*), we repeat the above process.

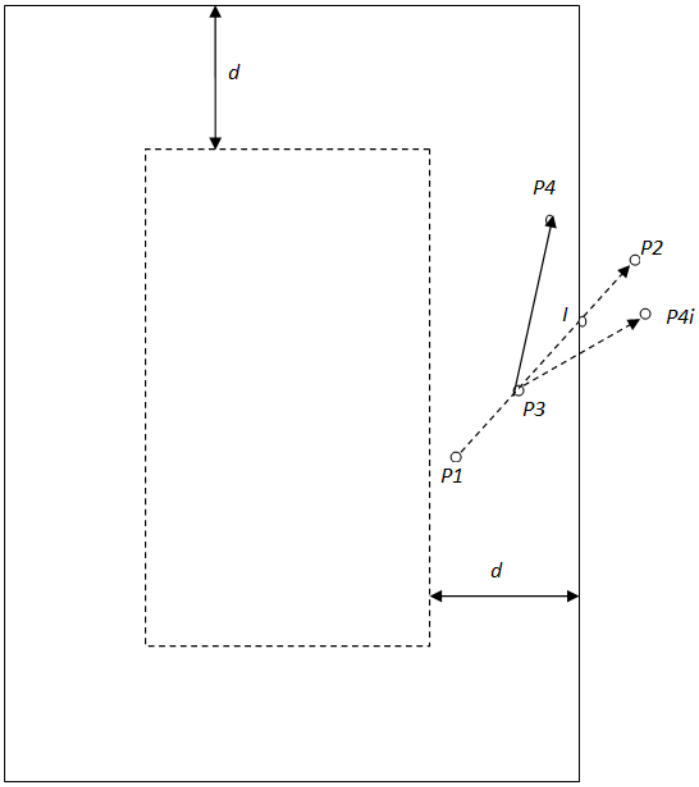

Figure 4.5: Out of region example

Now the problem is that as we keep taking that middle point, we get closer and closer to the boundary. The probability that the next position falls outside the boundary increases and we are likely to stay along the boundary for an extended period of time; which is not what we want. In order to avoid this situation where the node dwells along the boundary, we introduce a "position correction" scheme as follows (Figure 4.6). As soon as we realize that the current position is less than a distance *err* from the edge, if the next position (*P2*) falls outside the boundary, then instead of making *P3* the next position as before, we make *P4* (Figure 4.6) the next position. *P4* will be the point inside the region that has the same y-coordinate (or x-coordinate in the case of the top or bottom boundaries) as *P2* but is *corr_pos* away from the boundary. When doing this, the next direction is no longer the one calculated. We set *err=5m* and *corr_pos=7m*. By choosing those values, we ensure that the forced direction is close to parallel to the boundary, giving the impression that the node is just slightly nudged back into the region. Again, when the node is finally forced back into the region, bear in mind that the progressive U-turning described

in Figure 4.4 will still be triggered if (which is very likely at this point) we are not in the "safe area".

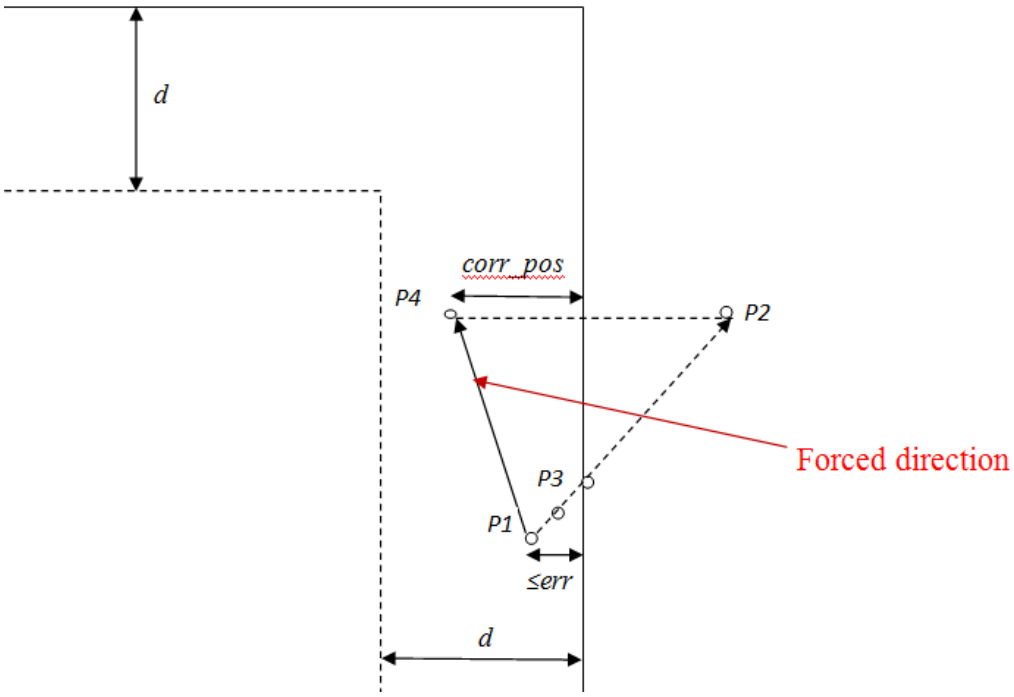

Figure 4.6: Out of region correction

Finally, Figure 4.7 shows another rare case where we fall back to a radical alternative in order to avoid leaving the simulation area. This is a special case of what was described with Figure 4.5. When the next position (*P2*) falls into the blue region, as we are working with lines' equations we want to know with which boundary the intersection occurs: the top boundary or the right one? As we know the equations of the lines that carry the boundaries, we can calculate the distances to those boundaries. In the case presented in Figure 4.7, the distance to the top boundary is denoted by *d1*, and the distance to the right boundary is denoted by *d2*. If $d1<d2$ we know that we intersected with the top boundary and that our computed next movement will look like the one depicted by the pair *P1b -> P2*. Otherwise, if $d1>d2$ then we are in a situation similar to *P1a -> P2*.

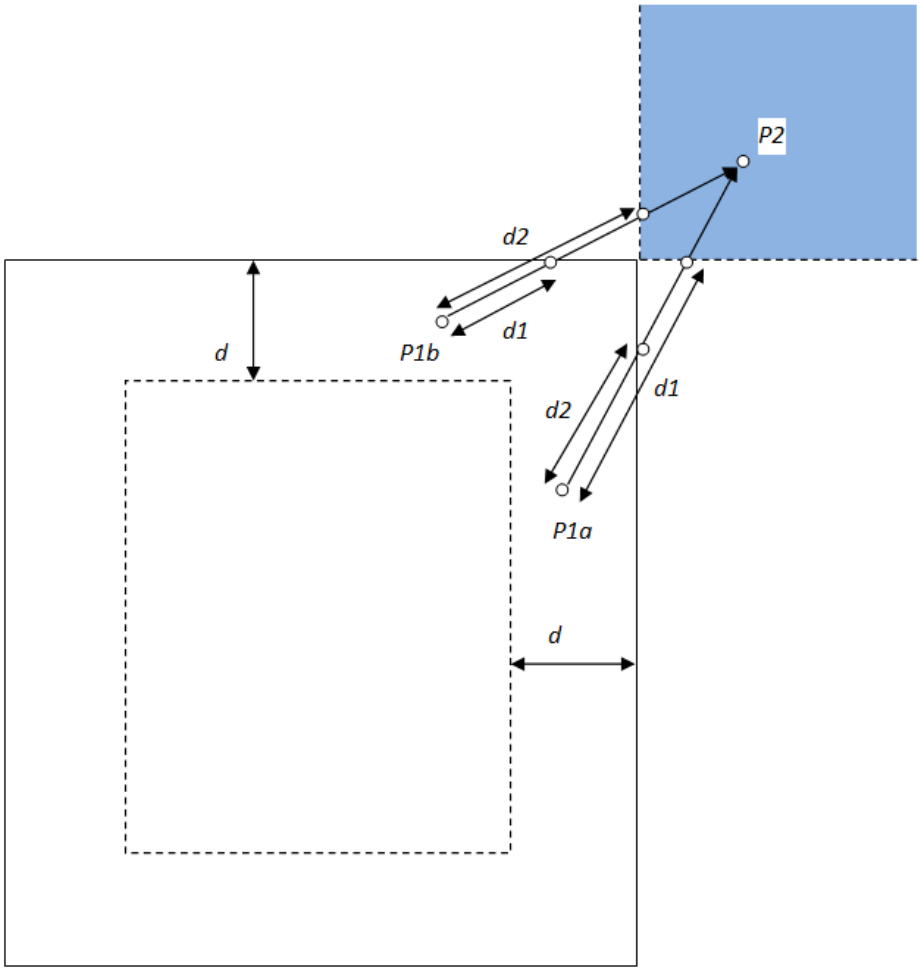

Figure 4.7: Corner

4.3 Summary

To summarize, the key mechanisms of the mobility model are as follows: when the node is far enough from the boundaries, the next speed and the next direction deviation are calculated using Equations (4.1) and (4.2). When the node gets close to a boundary (within a distance of 250 m from it) we initiate a turning toward the center region. This is in order to smoothly/gradually avoid hitting the boundary. If, despite our boundary-avoidance scheme, we still determine that the next position of the node falls outside the region, then we "bring it back in" a little more forcefully. However, the situations where we have to use more "force" are expected to be rare. Figure 4.8 summarizes these key mechanisms in a flow chart.

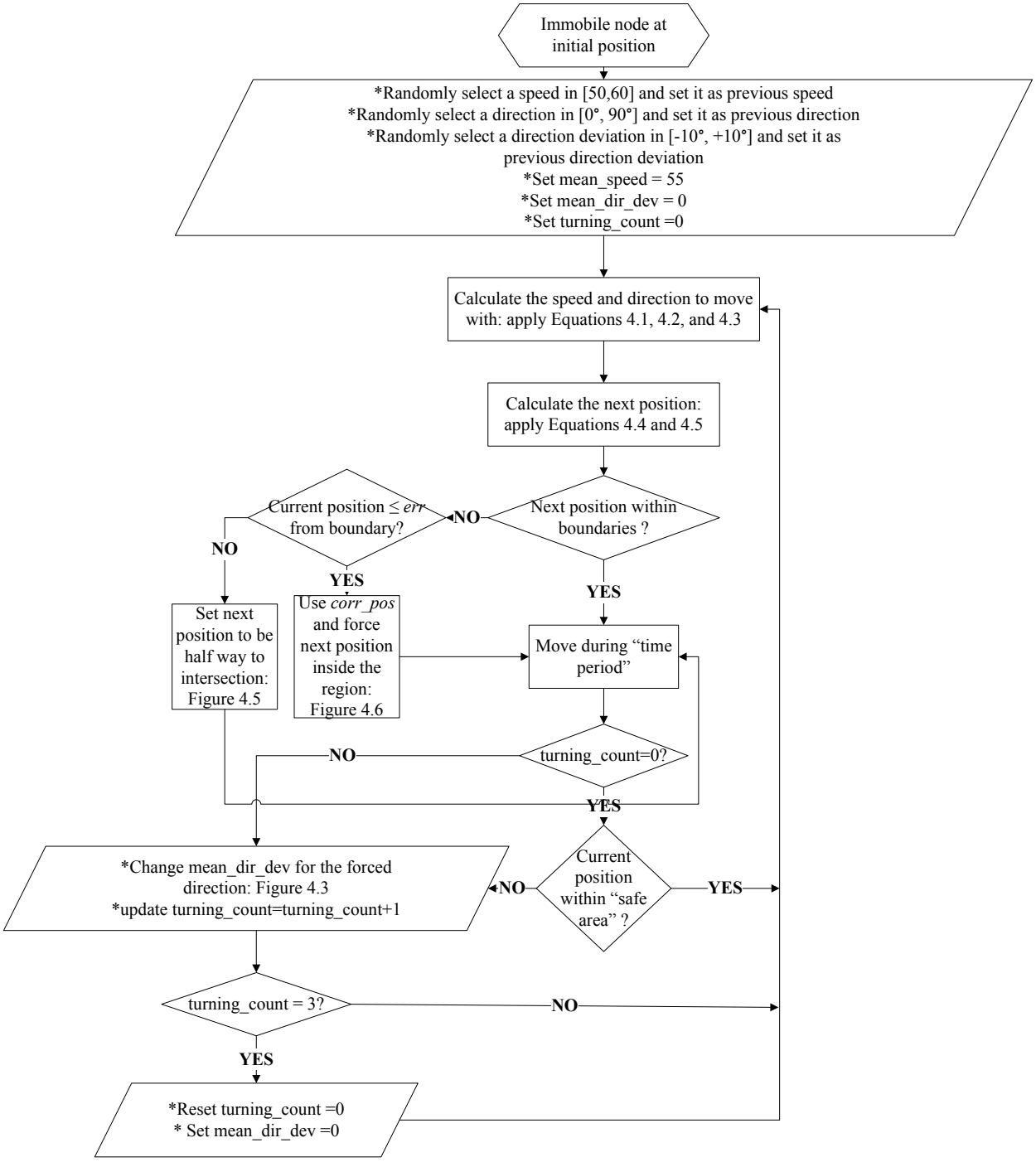

Figure 4.8: Key mechanisms of EGM

Finally, Figure 4.9 shows the trajectory of a UAV when EGM mobility is used in a simulation. As we can easily observe, the trajectory is noticeably smoother than the one

generated with RWP (Figure 2.3). There are virtually no sharp turns and the node does manage to avoid the boundaries in a very smooth and progressive way. This behavior that we observe at the boundaries is an advantage that EGM clearly has over the ST mobility model for example where we observe sharp turns at the boundaries (see Figure 2.6). The trajectory with GM (Figure 2.7) is not clear enough about what happens at the boundaries; or rather, it shows that the coverage in the vicinity of the boundaries (and in the entire region overall) is pretty poor compared to EGM. With EGM, as we can see (Figure 4.9), the node covers almost the entire region. Coverage here is only judged visually. This is an easy and not necessarily a reliable way of evaluation of the coverage. One better way of doing it would be to (mathematically) find out the steady state distribution of the nodes' locations. Having that information will give us a more informed and reliable idea of the actual coverage. We would then be able to compare the steady state node distribution of EGM versus that of RWP or any other mobility model.

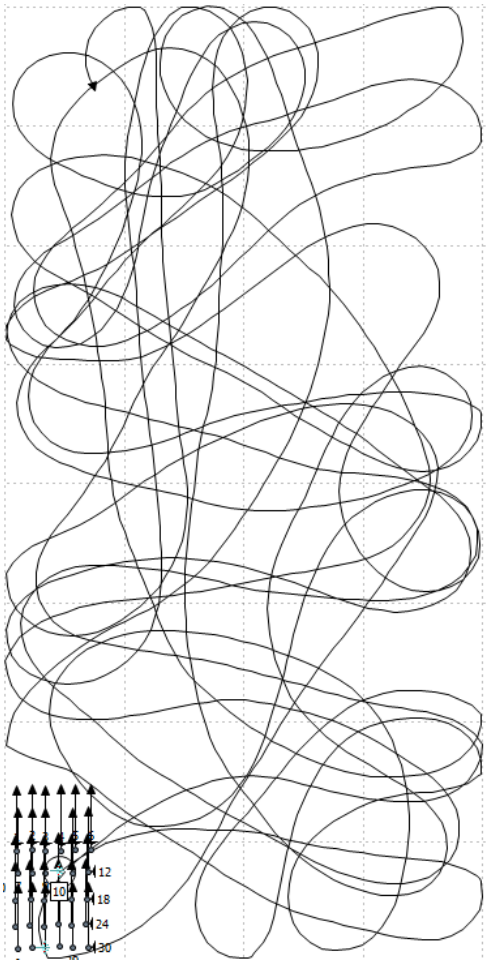

Figure 4.9: Trajectory of a UAV under EGM Mobility Model after 1800 s of Simulation

Only the trajectory of one mobile node is depicted (Figure 4.9). The trajectories of the 29 remaining nodes are hidden. Those hidden trajectories are not identical but present the same general characteristics such as smooth turns. Note that all 30 nodes are clustered at the bottom-left corner of the simulation region. That corner is considered as our UAV launch point.

In Chapter 5, we "quantitatively" compare EGM with two other mobility models, namely RWP and ST, by means of the performance of a routing protocol (namely the Optimized-RGR) and Flooding when evaluated using mobility scenarios based on these models. Note that we do not "quantitatively" compare EGM with GM because of the unavailability of the latter in OPNET. RWP was already available in OPNET, and we were able to easily implement ST in OPNET Modeler 16.0 [35] alongside EGM.

The EGM mobility model is more suited for searching missions where every node tries to explore the whole region independently from the other nodes' movement. At the same time, EGM is not too application-specific and therefore can be used for simulating a wide range of searching applications; which would not be the case had the mobility model been more deterministic/specific to a given application.

Chapter 5 : Simulation Results

5.1 Introduction

In Chapter 3, we presented two enhancements to the RGR protocol. First, the introduction of a stability criterion in the construction of the routes led to a new version of RGR referred to as Modified-RGR. Secondly, we introduced a recovery strategy for GGF failure. This strategy also applies to geographic routing in general. Since Modified-RGR (inherited from RGR) does feature geographic forwarding at some point, we integrated our strategy to Modified-RGR to obtain what we call Optimized-RGR. In this chapter, we present the results obtained when these two enhancements are added to the RGR protocol. More precisely, Section 5.2 introduces the simulation model, the parameters and also defines the performance metrics. Then, Section 5.3 presents the simulation results for Modified-RGR whereas Section 5.4 presents the results for Optimized-RGR.

In Chapter 4, we presented the EGM mobility model for simulation of UAANETs. In Section 5.5, we compare EGM with both the RWP and the ST mobility models. The comparison here is made by means of the performance of both Optimized-RGR and Flooding when each of these three mobility models is used for simulation.

5.2 Simulation Model, Parameters, and Performance Metrics

We used OPNET Modeler 16.0 to implement the modifications to RGR and simulate the end result. We set the channel capacity to be 11 Mbps for all mobile hosts. The rest of the simulation settings and parameters are summarized in the table below. The values of those parameters are set as presented before launching the simulation, and they do not change over the course of the simulation.

Table 5.1: Simulation Parameters

Parameter	**Value**
Number of simulated nodes	30
Area length	2000 m
Area width	4000 m
Wireless transmission range	1000 m
Packet size	1024 bits
Traffic rate	5 pkts/s
Speed	50 to 60 m/s
Pause time at simulation	0 s
Simulation time	1800 s

Note that, given the dimensions of our simulation region (2000 m x 4000 m), the 1000 m transmission range is chosen arbitrarily bearing in mind that we want to have multi-hop routes for packets in the network. For example, a 4000 m transmission range would have resulted in almost exclusively 1-hop routes with most nodes being able to directly communicate with any other node in the network. Also, a 100 m transmission range would have resulted in too sparse a network (given the area dimensions). Therefore, the choice of a 1000 m transmission range is a trade-off. The dimensions of the area are taken into account when making that choice. For the purpose of achieving a 1000 m transmission range, the packet reception power threshold of the nodes is set to -95 dBm.

By default, we have only one flow of information: one source node and one destination node for data packets. All 28 remaining nodes are just potential forwarding nodes. The nodes move according to the RWP mobility model by default. This is not the case in Section 5.5 where we

explore other mobility models. Occasionally, we make use of multiple data flows for analysis. We specify the details when that is the case for each set of results below.

The propagation model considered in our simulations is the same as the one considered in [1]. It is a free space path loss model that models the propagation as a disc around the transmitter. The effects of channel impairments are not addressed in this work as our focus is on routing protocol and mobility model design. We therefore assume a simple channel model with a predictable set of specifications.

For the MAC layer, we used the IEEE 802.11g standard where the Clear-to-send-to-self (CTS-to-self) protection mechanism is used in order to share the medium between the devices/nodes.

Note that we performed our simulations in 2D scenarios. This is only for simplicity sake. One can easily extend our simulations to 3D by simply adding a third coordinate to our nodes positions for instance. A consequent modification of the formulas we used would be necessary. However, it will require a specific (non-free) license in OPNET to visualize the network in 3D.

Performance is evaluated according to three metrics. First, we have the average PDR, which is the ratio between the successfully received packets and the total number of packets sent. The other two metrics are the control overhead and the average end-to-end delay. The control overhead is the number of control packets (per second) such as RREQs, RREPs, RERRs, and HELLO messages. Basically, it is the amount of extra traffic that is distributed in the network in order to provide the possibility of sending data packets. Finally, the average end-to-end delay is the averaged delay over all the data packets that make it from the source to the destination.

With our set of parameters presented earlier, we generate 10 independent scenarios using 10 different seeds of the pseudo-random number generator available in OPNET. By doing so, we have 10 sets of pseudo-independent results for every metric for every protocol. The 10 results are then averaged and the 95% confidence intervals determined for every metric. The confidence intervals, which are represented by small vertical segments along the graphs, help us establish the statistical significance of the differences or gaps between any two graphs. Note that we could have chosen to perform more runs than 10 in order to have smaller error margins. As we know, there is an inverse square root relationship between confidence intervals and sample sizes

(number of runs). For example, if we want to halve the margin of error that we have with 10 runs, we need to approximately quadruple our sample size, i.e. consider around 40 runs. Yet, more runs means more overall simulation time as each run take a considerable amount of time. By limiting ourselves to 10 runs we make the choice of less overall simulation time while, of course, taking the small risk of not establishing statistical significances. We primarily plotted our three metrics as a function of time in order to capture their behavior as the simulation runs. For analysis purposes, the metrics are further plotted as functions of network density, HELLO interval, etc. Note that at the very beginning of the simulation, all the nodes are bundled together at the bottom left-hand-side corner of the simulation region. That corner is referred to as the launching point. The nodes spread out as the simulation proceeds and eventually reach a steady state distribution. Therefore, when dealing with metrics as function of time, we are going to consider the values of our metrics toward the end of the simulation as they represent the values at steady state. The fact that the nodes are initially together typically provides the highest PDR and lowest delay. Also, note that all the metrics are averaged over time since the beginning of the simulation. All the metrics reach a plateau, typically after less than half the simulation time, with further variance well within the 95% confidence interval. We therefore conclude that these values represent the typical performance of the final steady-state UAV distribution. Note that steady state distribution here concerns only the nodes' location. In the case of RWP being used, the steady state distribution is a non-uniform spatial distribution [11] where the node density is maximum at the center of the region and almost zero around the boundaries. It takes some simulation time, depending on the minimum speed of the nodes, before that distribution is reached. In the case of another mobility model that steady state will not necessarily be that same non-uniform spatial distribution.

5.3 Simulation Results for Modified-RGR

As was seen in Section 3.2.1 of Chapter 3, the integration of a route stability criterion in the route construction process of RGR yielded a new version of the protocol that we refer to as Modified-RGR. We set the constant $w = 5$ for computing the reliable distances. We will see the impact of changing that value as well as the explanation of why we chose that value later on. As

we can see in Figure 5.1, we have a statistically significant increase in PDR of about 3.5% with Modified-RGR compared to the existing RGR protocol. This was expected as we now utilize more robust links.

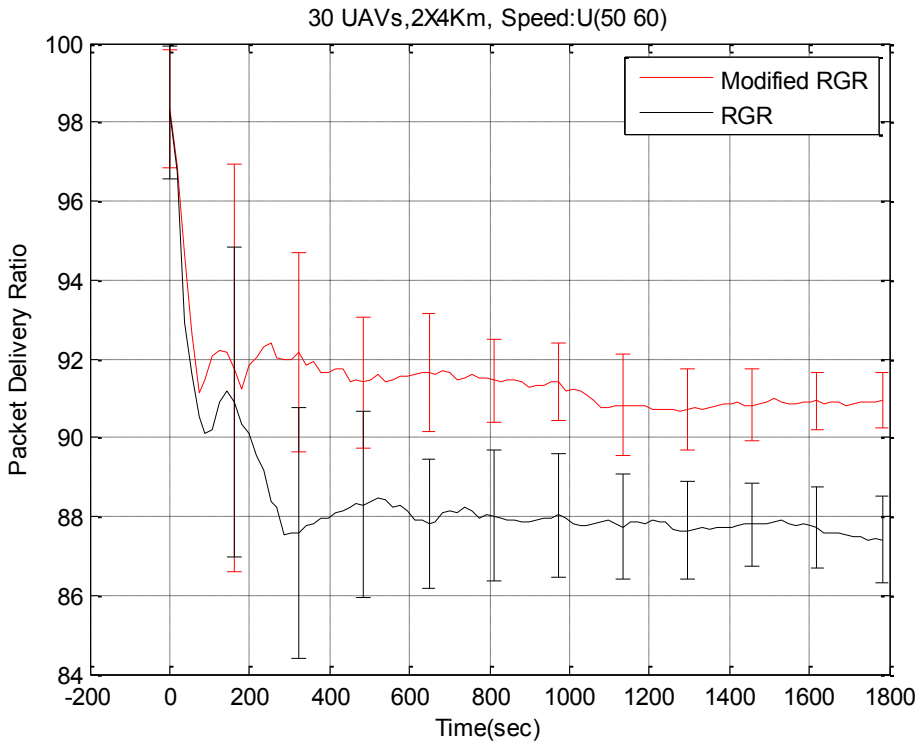

Figure 5.1: Packet Delivery Ratio

Figure 5.2 highlights a slight drop in control overhead. This results from the fact that with Modified-RGR, we have fewer route discoveries (starting with fewer RREQs issued) since the created routes are more stable. The stability also leads to fewer broken routes and therefore fewer RERR messages. Note that the single biggest contributors to protocol overhead are the periodic HELLO messages. Here we choose our HELLO Interval (HI) to be about 1 second. In fact, the HI is picked from a uniform distribution [1, 1.1]. Therefore, on average, the HI is about 1.05 second. Considering that all the 30 nodes generate a HELLO after every HI, the contribution of the HELLO messages in the control overhead is on average equal to 30/1.05 = 28.57. The actual HELLO overhead varies slightly from this number, as HELLO message generation is not unconditional. Two conditions must be satisfied for a node to generate a HELLO: i) the node has

not sent a HELLO or RREQ (initiated or rebroadcasted) within the last HELLO Interval and ii) the node is part of an active route. Note that, despite having only one flow of information, all nodes will almost always be part of an active route as any node that receives a broadcasted HELLO automatically creates an active route to the source of the HELLO. On the other hand, some HELLOs are cancelled because a RREQ has been sent. Therefore, the real contribution of HELLOs to the overhead is about 28.57 minus the number of RREQs (initiated and forwarded/rebroadcasted). In both protocols, HELLO messages contribute equally to the overhead initially. The difference in overhead therefore comes from other control packets (RREQs, RREPs, and RERRs). As shown in detail in Figure 5.3, Modified-RGR, by design, leads to fewer RREQs (thus RREPs) and RERRs, as the routes are more stable. Thus the slight drop (noticeable and statistically significant) observed in the average number of control packet.

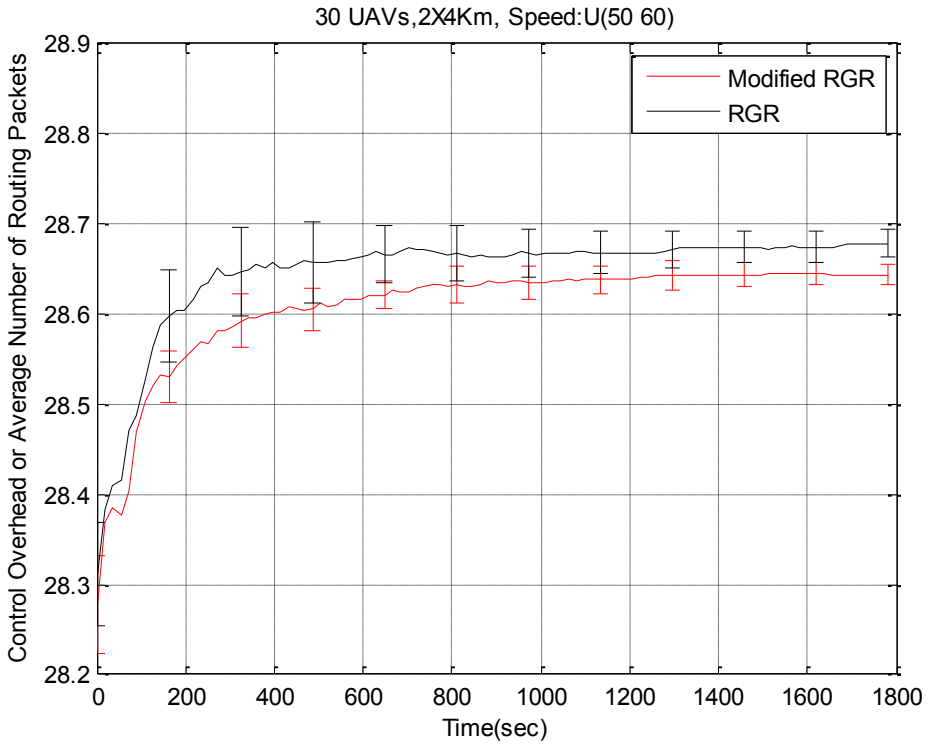

Figure 5.2: Average Routing Traffic

In terms of end-to-end delay, there is no statistically significant difference between Modified-RGR and the existing RGR. Figure 5.4 shows that via the overlapping confidence

intervals. The result here is expected since there is no modification in Modified-RGR, compared to original RGR, that induces additional delay. We did not really strive to reduce packet latency. Plus, it is apparent that the additional packets that are delivered do not experience any difference in end-to-end latency compared to the others (delivered by regular RGR).

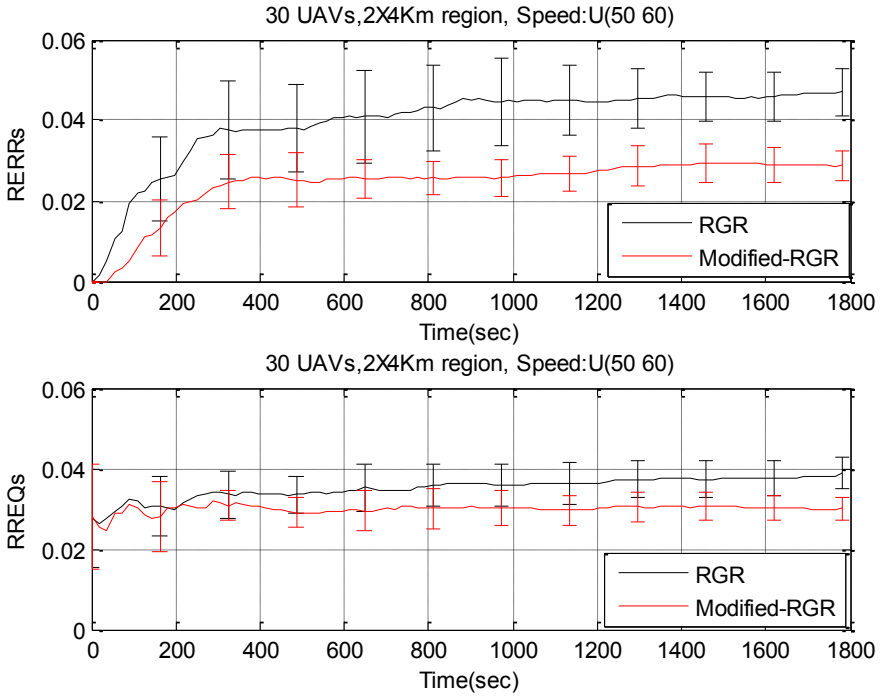

Figure 5.3: RREQs and RERRs

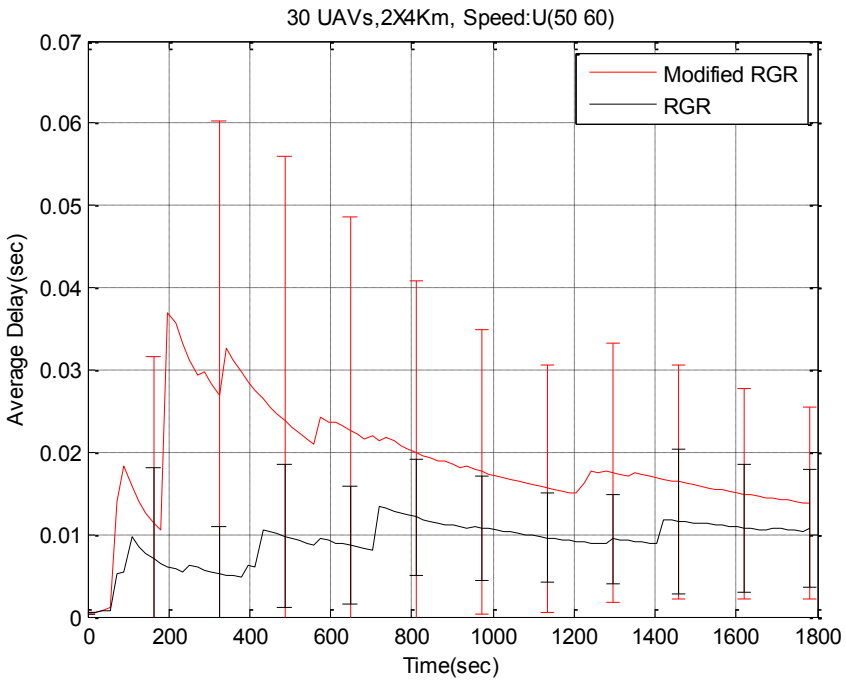

Figure 5.4: Average Packet Delay

As we saw previously, the HELLO messages constitute the bulk of our routing overhead. One option to reduce this overhead would be by increasing the HELLO Interval. The higher the HELLO Interval, the lower is the control overhead. For example, doubling the HI from 1 second to 2 seconds divides the number of HELLOs by 2; therefore it also divides the control overhead by 2 as we already know that HELLOs are the main component.

Figure 5.5 shows how changing the HELLO Interval affects the performance of the protocols. We focus on the PDR metric. We can see that the PDR decreases considerably as we increase the HELLO Interval. This decrease is the price we pay for reducing the control message overhead. While increasing the HI from 1 to 2 halves the control message overhead, PDR is decreased by about 6% for both protocols. We can also observe that changing the HI affects both protocols almost equally (comparable slopes in Figure 5.5). Moreover, Modified-RGR still constitutes a considerable enhancement over RGR independent from the value of the HELLO Interval. Note that a HI of 0.5 second gives the highest PDR, yet choosing HI=0.5 second over HI=1 second for our protocols will double the overhead.

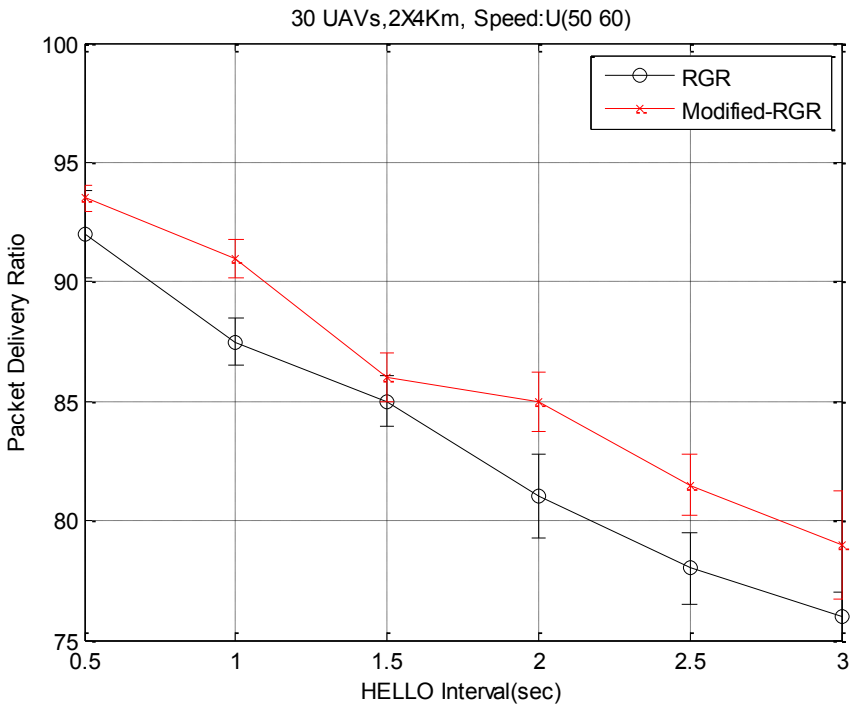

Figure 5.5: Impact of HELLO Interval

Figure 5.6 shows the effect of changing *w* on the PDR of Modified-RGR. The value of *w* is essential in the calculation of the reliable distance *r*. Note that our minimum value of *w* in the figure is 1, not 0. From Equation 3.3, it is understood that if *w*=0, *r* simply becomes the transmission range and the PDR for Modified-RGR is expected to be the same as the one for RGR as there is no reliability/stability added to the routes. On the other hand, a very high *w* will result in very small reliable distances and therefore a lot of possible routes will not be considered, resulting in either a long path with many short hops or, in the worst case, no detectable route at all. Both have a negative impact on the PDR: long paths (i.e., many packet retransmissions) increase the chance that something goes wrong as the packet is forwarded towards the destination, and the absence of a valid route leads to network partitions. These negative impacts are confirmed in Figure 5.6. For example, we know that our nodes move with a speed in the range [50-60] m/s. Let us assume a relative speed of 6 m/s, as was observed during simulation. In this case, a value of *w*=100 results in a reliable distance of about 400 m. Links that have a distance of 700 m will be deemed unreliable even though in reality they will not break for

another few seconds. The values w=5 and w=10 achieve the highest PDR. However, as we can see, for all values of w in [5, 100] we get a PDR in excess of 90%, which is better than original RGR (87.5%).

Figure 5.7 focuses on the effect of network density in terms of the number of nodes in a given region of 2 km x 4 km. As expected, the sparser the network, the poorer the PDR gets. Moreover, as the network gets sparser (number of nodes ≤10), the difference between our two protocols becomes less apparent as routes become more difficult to find to begin with. The same is expected to be true when the number of nodes is high. In this case, most routes will be pretty reliable/stable as the nodes constituting them will be closer to one another, and it does not make a difference to use RGR or Modified-RGR. In fact, RGR tries to find the shortest route (in terms of hops). Now, with more nodes, there is a higher probability that the hops on a route are all of about equal length and/or are short enough not to break before a good while. Therefore there is no big difference with Modified-RGR that explicitly enforces stability through the reliable distance. For example, if source and destination are 1500 m apart, we need a 2-hop route. Due to a high population of nodes maybe we have one where the first hop is 800 m and the second hop is 700 m. And maybe another one with 3 hops where the first hop is 600 m, the second hop is 500 m and the third one is 400 m. RGR would choose the 2-hop route. These hops of 800 m and 700 m form a pretty stable route that does not break for a while. Had we not had an important population of nodes to begin with, we probably would have had a 2-hop route where one of the hops is 980 m and breaking soon. And in this case, using Modified-RGR would have made a difference because that route with a 950 m hop would have been discarded in the first place due to non compliance with a reliable distance that most likely is considerably smaller than the 1000 m transmission range. At the end, Modified-RGR would have resulted in the use of a 3-hop route that is more stable, yielding a higher difference in PDR with RGR. We observe that the difference between the two protocols is highest with a 30 node network.

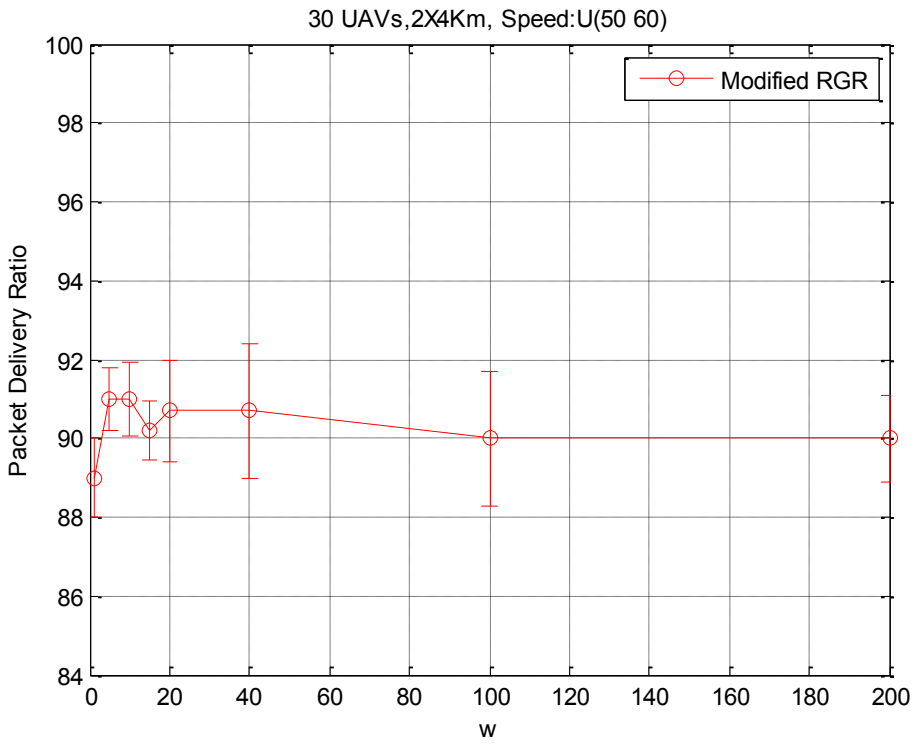

Figure 5.6: Impact of w varying in set [1, 5, 10, 15, 20, 40, 100, 200]

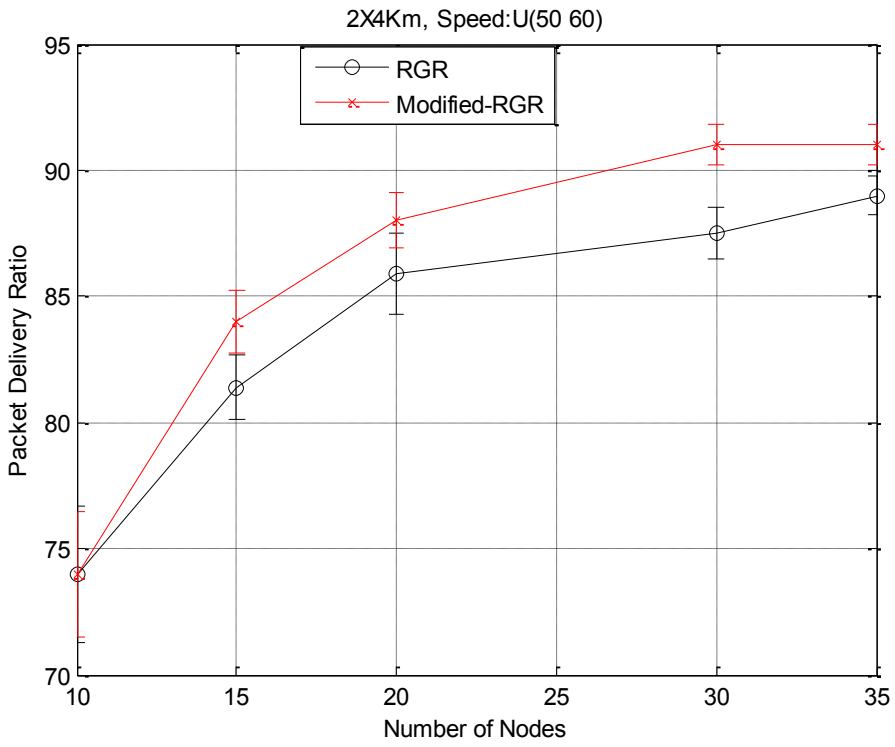

Figure 5.7: Impact of Network Density

Finally, Figure 5.8 focuses on the effect of the number of flows of information on the PDR of both protocols. Adding flows of information does not have a noticeable impact on the PDR of RGR, whereas the PDR of Modified-RGR slightly drops when we go from a single flow to multiple flows. The large channel capacity (11Mbps) prevents saturation, since our network load is only in the order of 100 Kbps. This is the reason why there is virtually no impact on RGR. The reason for the slight decrease observed for Modified-RGR is not clear at this point, and we can only conclude that introducing a reliability criterion in route construction works best when we deal with a single flow of information. Nevertheless, for all number of flows Modified-RGR outperforms RGR.

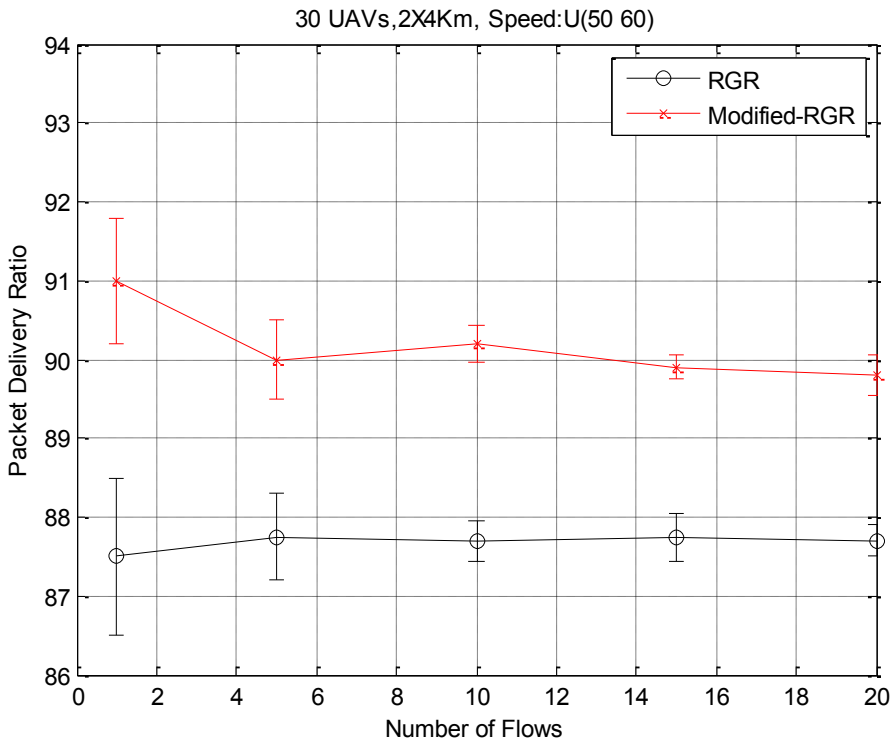

Figure 5.8: Impact of the Number of Flows varying in set [1, 5, 10, 15, 20]

5.4 Simulation Results for Optimized-RGR

As discussed in Section 3.2.2, each of our three strategies to recover from GGF failure is added to Modified-RGR to obtain a temporary version of Optimized-RGR. The temporary version obtained with Strategy 1 for instance will be referred to as Optimized-RGR-Strategy 1. All three versions of Optimized-RGR are compared to one another and are also compared to Modified-RGR. RGR is also presented as an indication of where Modified-RGR comes from.

Figure 5.9 shows the PDR. Optimized-RGR-Strategy 3 exhibits the highest PDR (about 92.8%) of all three strategies. This suggests that picking the Best-Moving Node in order to forward the packet provides a higher chance to reach the destination than picking the furthest neighbor (Strategy 2) or retrying GGF (Strategy 1). We achieve an increase of about 1.8% compared to Modified-RGR, and an increase of about 5.3% compared to RGR. Strategy 1,

retrying GGF once, does not help the PDR at all; suggesting that once GGF has failed, there is no point looking for a neighbor closer to the destination than the FN, even after a hold-on period.

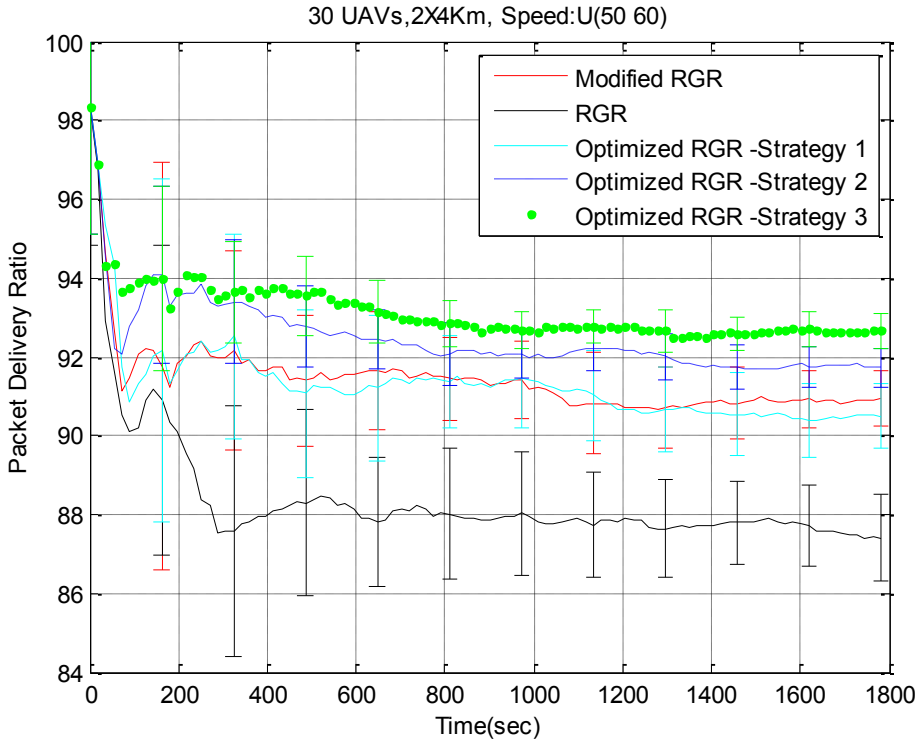

Figure 5.9: Packet Delivery Ratio

Figure 5.10 shows no difference in control overhead for all three strategies, nor does it show any difference at all compared to Modified-RGR. This was expected since, with Optimized-RGR (including all three strategies), we just take care of GGF failure with no additional route discovery whatsoever. Note that the notion of retrying to forward a packet after a hold-on period does not mean that the packet is actually forwarded unless there is a suitable neighbor (depending on strategy 1, 2, or 3) to forward it to. The term "retrying" only means that we explore the neighborhood again. At the end of the day, the packet is forwarded only once: when there is a suitable neighbor. Therefore, there is no additional overhead induced by data packet retransmission whatsoever.

In terms of end-to-end delay, all three strategies have a little higher delay than RGR and Modified-RGR, yet it is still of the same order and below 0.05 second (Figure 5.11). The increase in delay comes as no surprise since we introduce a hold-on period for the packets when GGF fails.

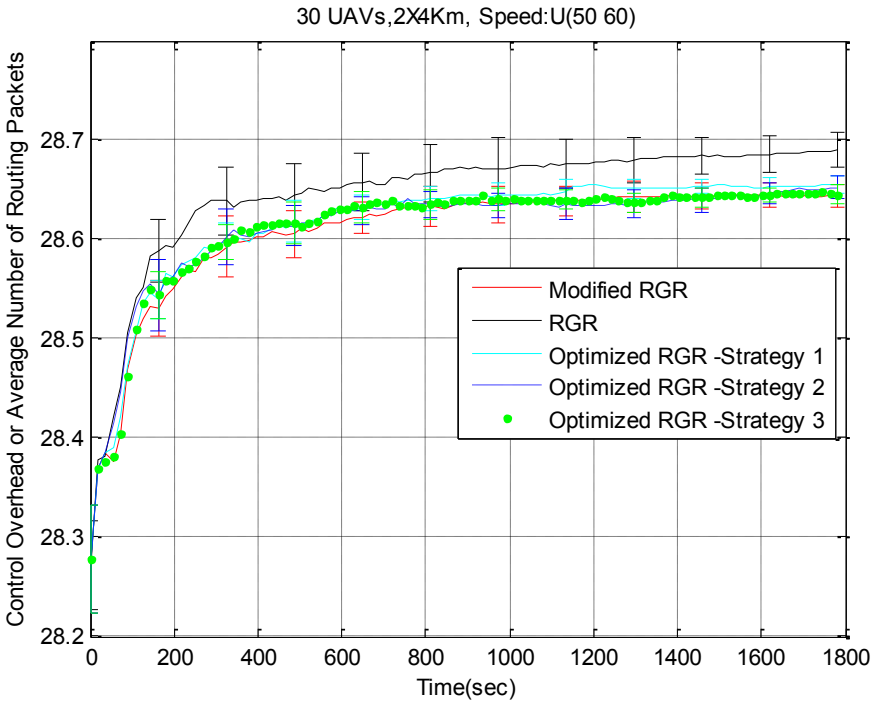

Figure 5.10: Average Routing Traffic

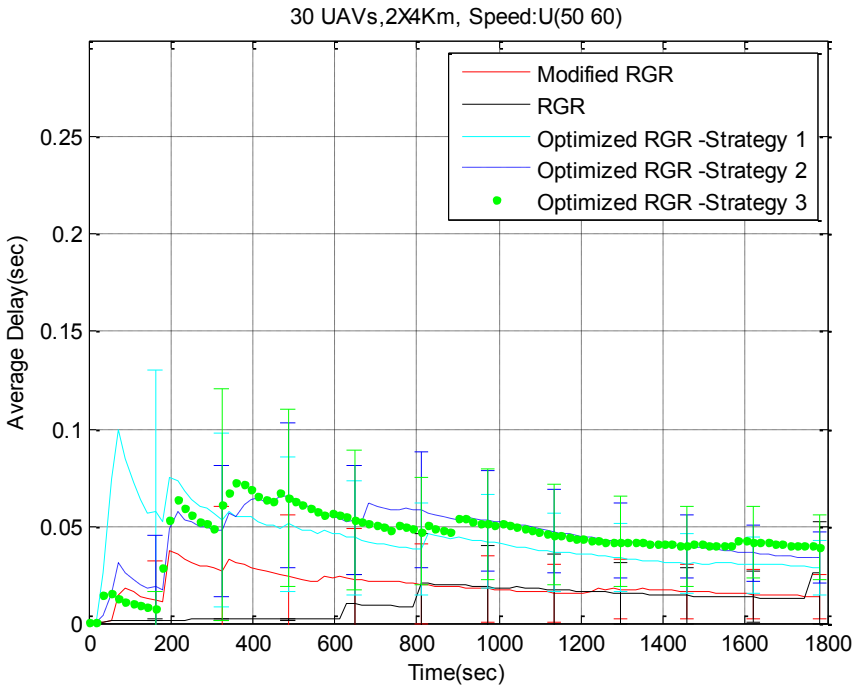

Figure 5.11: Average Packet Delay

From these results, it is clear that Optimized-RGR-Strategy 3 is the best choice to further improve on Modified-RGR. Therefore, from now on, when we refer to Optimized-RGR we mean Modified-RGR to which we append Strategy 3 i.e. forwarding the packet to the Best-Moving Node when GGF fails. Strategy 3 is our proposed recovery strategy for GGF failure.

As we claimed earlier, the recovery strategy is not specific to Modified-RGR but applies equally well to other geographic routing protocols. To provide a small demonstration of this, we added the same recovery strategy (determine and forward packet to the Best-Moving Node when GGF fails) to the RGR protocol. Figure 5.12 shows that, when utilizing the same GGF recovery strategy that was added to Modified-RGR (in order to obtain Optimized-RGR), RGR has an increase of about 3% in PDR (compare black line with blue); which is higher than the 1.8% increase observed with Optimized-RGR (compare red line with green). Yet, the resulting PDR is at about 90.5%; which is lower than the one in Optimized-RGR (92.8%). These two results are expected. First, it is normal to have a higher PDR with Optimized-RGR because the latter is based on an enhanced version of RGR (Modified-RGR with 91% PDR) to which the recovery strategy is added. Secondly, it is also normal to have a slightly lower increase in PDR (1.8%) from Modified-RGR to Optimized-RGR because we start from a higher PDR (Modified-RGR at 91%) than RGR (87.5%). It therefore becomes more difficult to deliver more packets.

In the next few paragraphs, we explore the impact of varying three parameters on the performance (PDR) of Optimized-RGR. The three parameters are: the HELLO Interval, the network density, and the number of flows. The impact on Modified-RGR and RGR is also presented at the same time for comparison purposes.

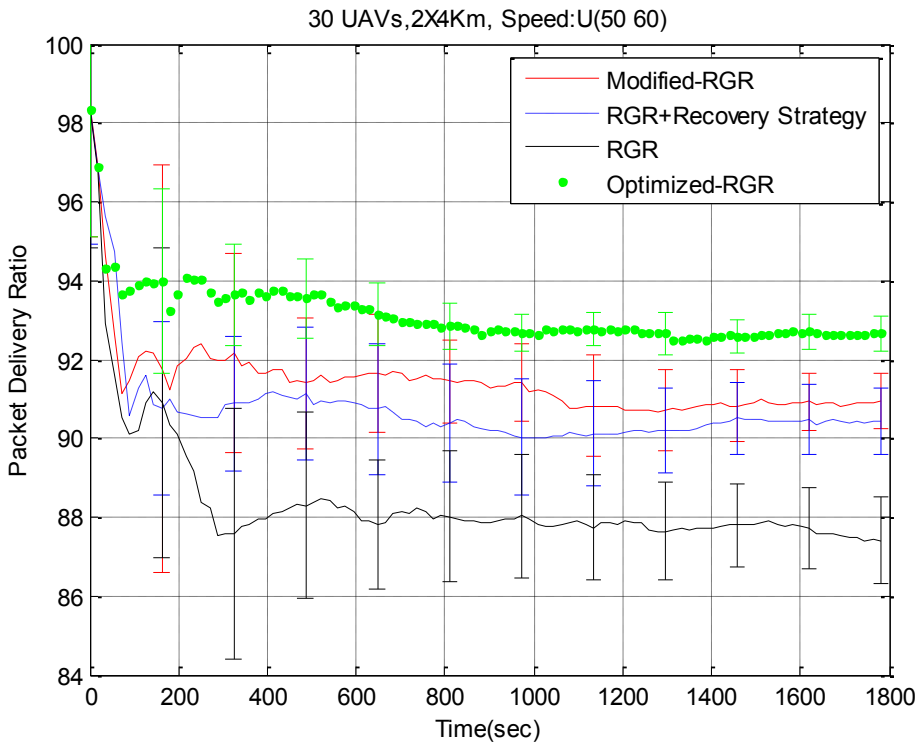

Figure 5.12: Optimized-RGR Vs RGR with Recovery Strategy

Figure 5.13 shows that the PDR in Optimized-RGR decreases virtually at the same pace as RGR and Modified-RGR when the HELLO Interval is increased. The GGF recovery strategy does improve the performance of Modified-RGR at any given HELLO Interval. Choosing a HELLO Interval comes down to how much control overhead we are willing to afford; as HELLO messages constitute the bulk of overhead in our protocols. The widely used value of 1 second (default value in OPNET) also shows the clearest difference between the protocols. The highest PDR (95%), using Optimized-RGR, is achieved when the HELLO Interval is 0.5 second, however, this doubles the control overhead.

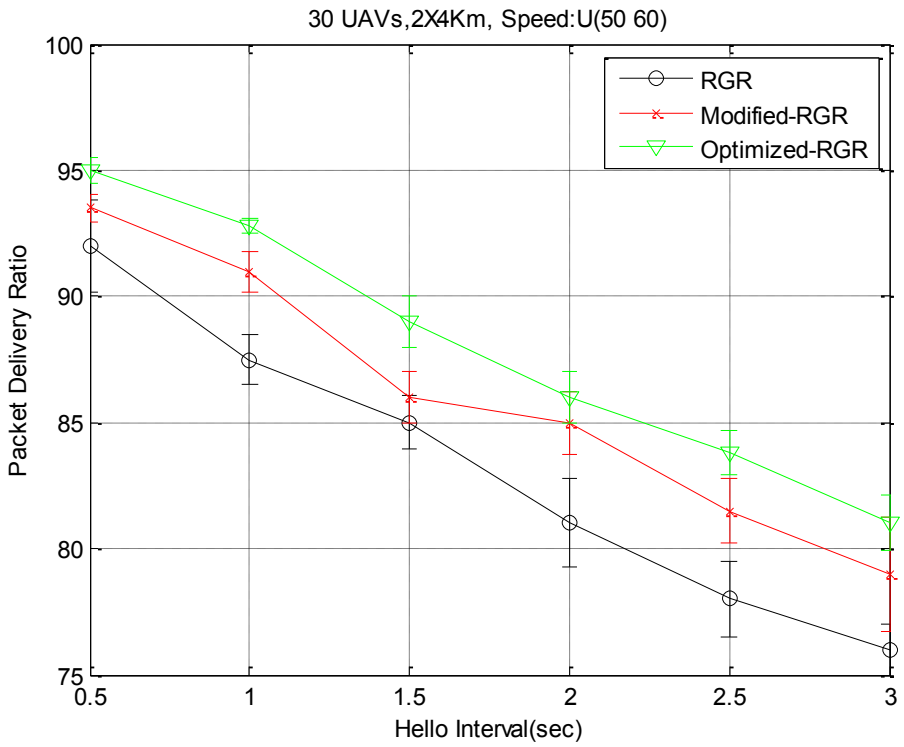

Figure 5.13: Impact of HELLO Interval

Figure 5.14 shows that Optimized-RGR clearly outperforms Modified-RGR when the number of nodes falls within certain boundaries. Too low a number of nodes, like 10, results in a barely noticeable difference between the protocols because the network is too sparse. Also, when the number of nodes is too high, it is expected (as it starts to show in the figure with 35 nodes) that all protocols will perform very well with almost no difference. GGF almost never fails (with an increased number of neighbors resulting in an increased probability to find a suitable one for GGF) and the presence or not of a recovery strategy therefore makes almost no difference. In the meantime, a population of 30 nodes presents the highest difference between the protocols, and Optimized-RGR achieves its highest PDR, and does not further improve for the 35 node case.

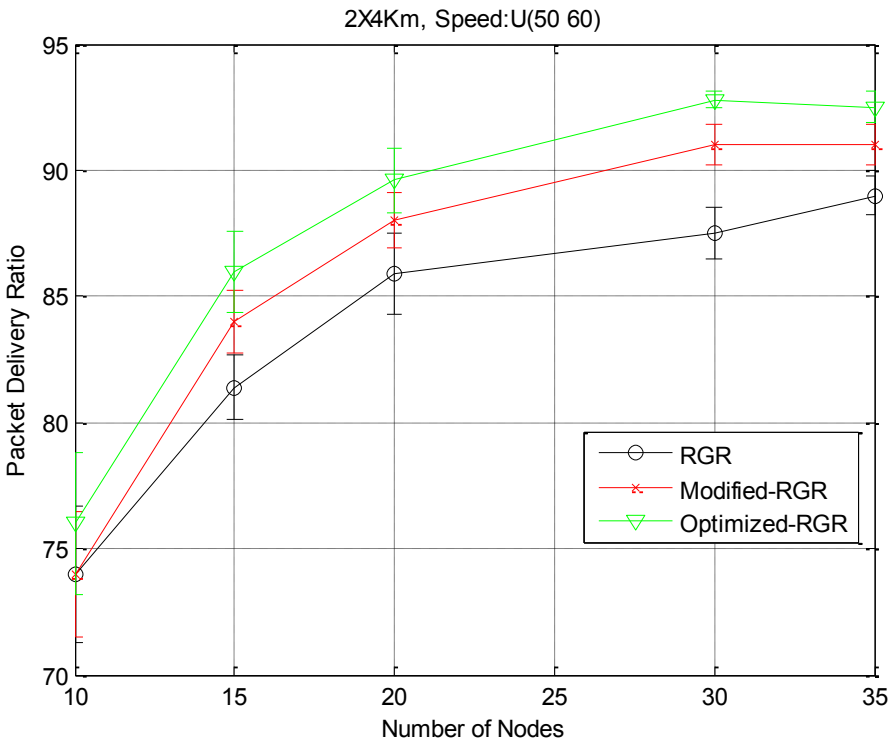

Figure 5.14: Impact of Network Density

Finally, Figure 5.15 shows that Optimized-RGR always performs better no matter the number of flows. There is a notable drop in performance when we go from 1 flow to 5 flows, as observed for Modified-RGR in Section 5.3 already. The performance is quite stable after that. The conclusion is that adding a GGF failure recovery strategy always improves the protocol performance, notwithstanding the number of flows.

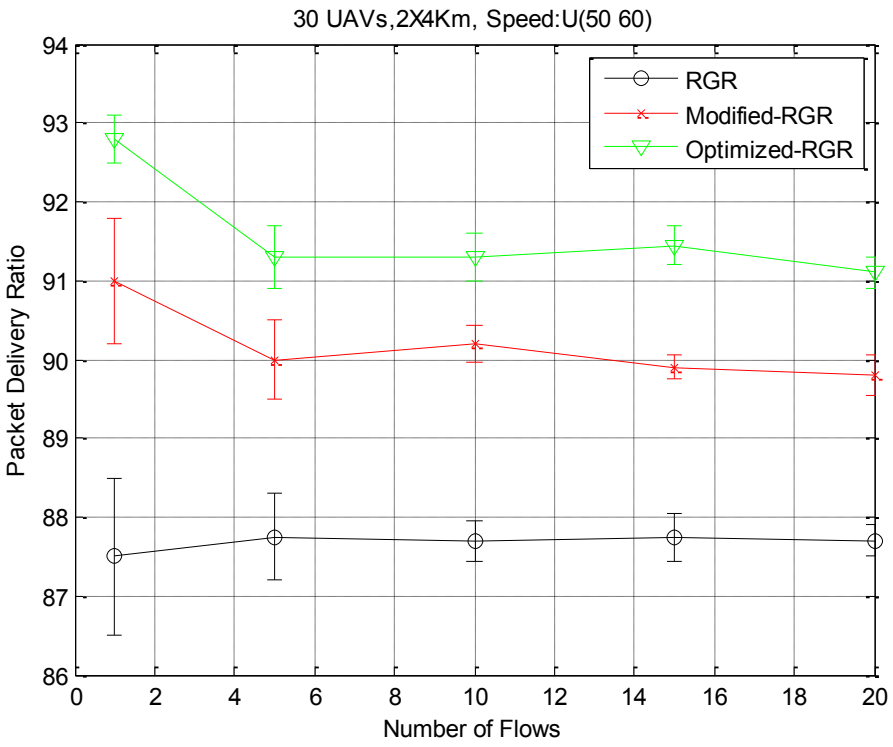

Figure 5.15: Impact of the Number of Flows varying in set [1, 5, 10, 15, 20]

5.5 Simulation Results for the Mobility Model

We presented the EGM mobility model in Chapter 4. In this section, we compare EGM with both the RWP and ST mobility models in terms of their effect on the routing protocol performance. For this purpose, we observe the behavior of Flooding and Optimized-RGR in terms of PDR under all three mobility models. More precisely, we want to see whether the protocol performance will differ when evaluated under different mobility models. And if there are differences, we want to know what it is about the models that causes them.

Note that, although EGM is based on GM, we do not consider GM for comparison because of two reasons. First, it is unavailable in OPNET to begin with. Secondly, as pointed out in Chapters 2 and 4, we have an issue with the very notion of "mean direction" used in GM. RWP was already available in OPNET, and we did implement ST in OPNET ourselves alongside EGM.

Table 5.2 outlines the simulation parameters and their values that we used for the EGM model. Table 5.3 does the same as far as the ST implementation is concerned.

Table 5.2: Parameter description for EGM implementation in OPNET

Simulation parameter	**Description**	**Value**
Bounds	2D boundary of the cruising area (meters)	X: [0, 2000] Y: [0, 4000]
Simulation time	Total duration of the simulation (seconds)	1800
T	Parameter recalculation interval (seconds)	1
alpha	Tunable parameter α	0.86
mean_speed	Average velocity of the nodes (m/s)	55
Speed range	Speed values allowed for nodes (m/s)	[50, 60]
mean_dir_dev	Average direction deviation of the node. It is relative to previous direction. (degrees)	0: when far enough from boundaries. 22.5 or -22.5: within distance *d* to boundary
speed_normal_dist	Gaussian noise distribution for the velocity calculation.	Normal distribution. Mean = 0 , variance = 1.55
dir_dev_normal_dist	Gaussian noise distribution for direction deviation calculation	Normal distribution. Mean = 0, variance = 6.21: when far enough from boundary. Mean = 0, variance = 0.388: within distance *d* to boundary.
Margin *d*	The distance to a boundary at which the node begins to turn progressively back to the center region (meters)	250

Table 5.3: Parameter description for ST implementation in OPNET

Simulation parameter	**Description**	**Value**
Bounds	2D boundary of the cruising area (meters)	X: [0, 2000] Y: [0, 4000]
Simulation time	Total duration of the simulation (seconds)	1800
Epoch/time step	Vehicle's Location recalculation interval (seconds)	1
ExponentialMean	Mean value of the waiting time, the time during which the airborne vehicle keeps the same turning center (second)	11
Varian	Variance of the normal distribution from which radius inverses are picked from	0.00000155
Speed range	Speed values allowed for the nodes (m/s)	[50, 60]
radius_inv_normal_dist	Gaussian distribution for the radius inverse.	Normal distribution. Mean = 0, variance = varian

Each of the next three figures shows the PDR of both Flooding and Optimized-RGR when evaluating the protocols using the three different mobility models.

Figure 5.16 shows the PDR under EGM. Compared to the results under RWP (Figure 5.17), there is a drop of about 2.5% in PDR performance when evaluating Optimized-RGR. This drop in performance is explained by the fact that, under EGM, the network is a lot more partitioned than under RWP. We distinguish two types of partitioning. On the one hand, there is partitioning where the source node and the destination node are in two different partitions. We are going to call this a Type-1 Partition. On the other hand, there is partitioning where the source node and the destination node are part of the same partition. This is a Type-2 Partition. A Type-1 Partition explains the drop in PDR for flooding, whereas Type-2 explains the drop for the routing protocol under certain conditions: First, the affected node(s) must have been part of an active route just before the partitioning occurred. Secondly, at least one of the switches to GGF (which is part of

the protocol) has to fail along the way, i.e., neither GGF nor the salvaging via determining the BMN will succeed in forwarding the data packet.

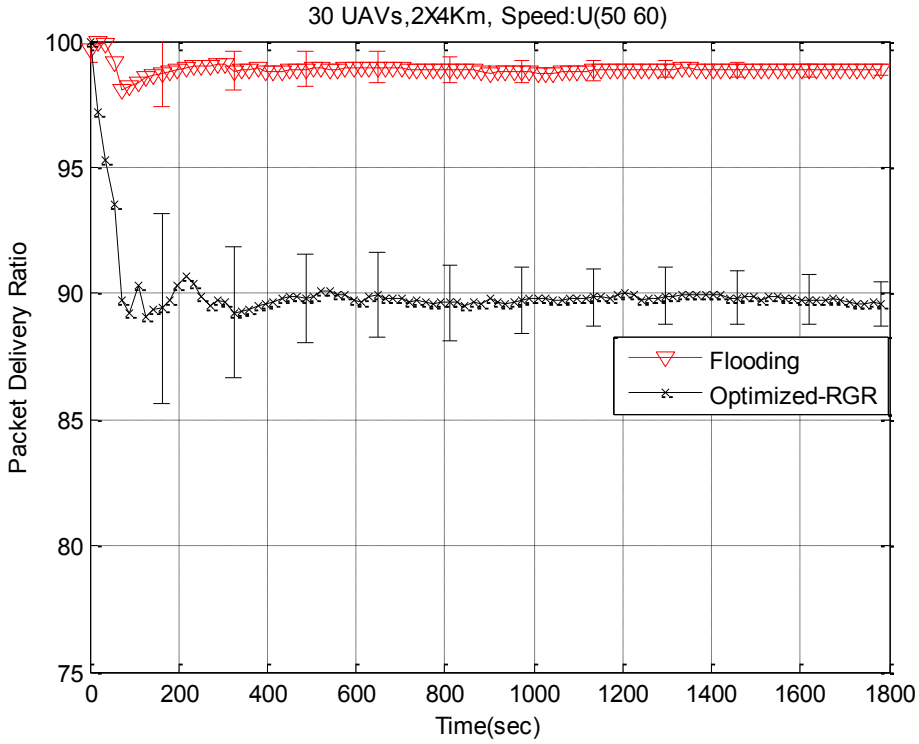

Figure 5.16: PDR under EGM

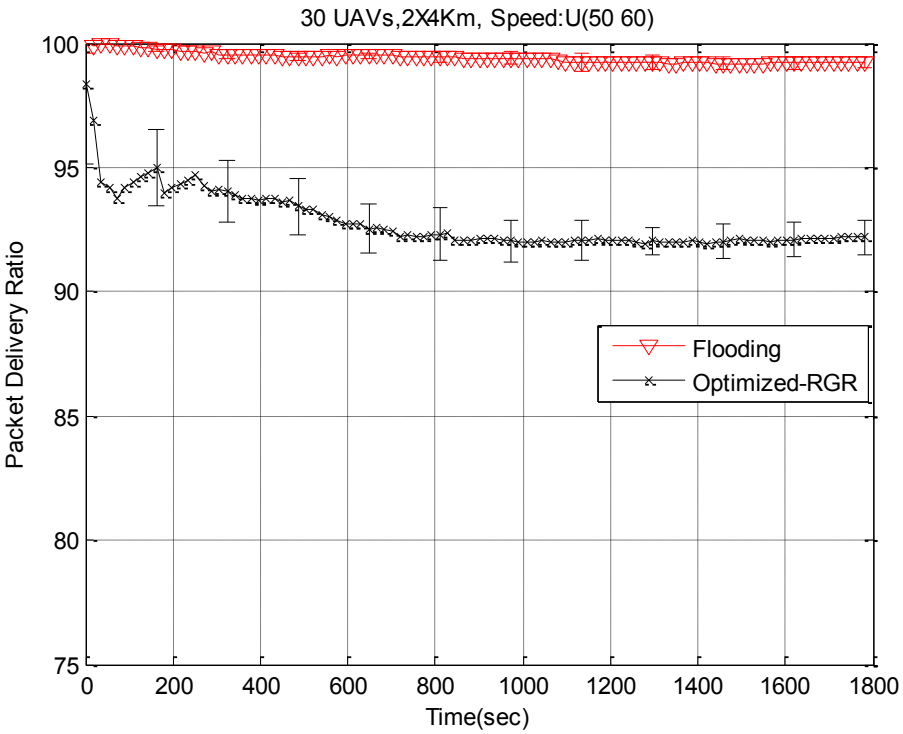

Figure 5.17: PDR under RWP

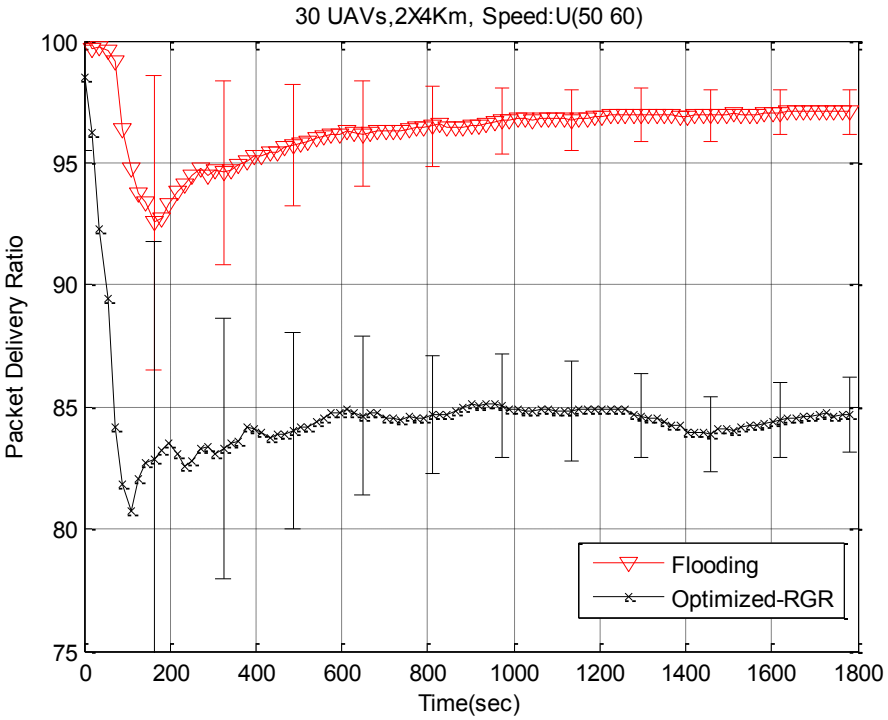

Figure 5.18: PDR under ST

Table 5.4 shows the number of partition occurrences for all mobility models. We determined this number by taking periodic snapshots of the network every second. It turns out that the number of times a partition is observed is also a good approximation of the relative frequency, as we varied the interval time and got very similar results when taking snapshots every 0.1 seconds. The numbers presented here are averages over 10 different runs. Under RWP, the network spends approximately 5 seconds (out of the 1800 seconds of simulation) being Type-1 partitioned, which is about 0.3% of the time. Under EGM, the network is Type-1 partitioned for 20 seconds, which is about 1% of the time. Type-1 partitions explain why the PDR is not 100% when packet flooding is used. For RWP, we expect a PDR of about 100%-0.3%=99.7%. For EGM, we expect a PDR of 100%-1%=99%. This corresponds well with the results seen in Figure 5.17 (RWP) and Figure 5.16 (EGM), which show that both mobility models achieve results of 99% and above.

Table 5.4: Number of partition occurrences

Mobility model	Type-1 Partitions	Type-2 Partitions
EGM	20	137
RWP	5	63
ST	41	179

Considering Type-2 partitions, under RWP the network spends 3.5% of the time (63 out of 1800 seconds) being Type-2 partitioned. Under EGM, the network spends about 7.6% of the time (137/1800) being Type-2 partitioned. Type-2 partitioning partially explains the PDR gap under both mobility models. The difference between the 2 mobility models in terms of time spent in Type-2 partitioning is about 7.6%-3.5% = 4.1%. The difference (statistically significant) in PDR for the routing protocol (in black color) that we see between Figure 5.17 (RWP) and Figure 5.16 (EGM) is about 2.5%. Not all Type-2 partitioning will result in failure to deliver a data packet. In fact, a Type-2 partition just tells us that one or multiple intermediate nodes are disconnected from the rest of the network. These intermediate nodes have to meet the two conditions mentioned earlier in order to affect the PDR. Therefore, it is to be expected that we observed a smaller gap (here only 2.5%) in PDR performance, rather than a gap of 4.1%. Finally,

note that, unlike Type-1 partitions with Flooding, Type-2 partitions do not explain completely why the routing protocol does not reach 100% PDR. As discussed, under EGM we spend 7.6% of time in Type-2 partitions. A fraction of that time directly affects the routes (and thus the PDR). Say that fraction is 2.5% / 4.1% = 0.6. Generalizing this, we therefore expect a loss of 7.6% * 0.6 = 4.56% in PDR stemming from Type-2 partitions. Had the routing protocol worked perfectly, we would have expected a PDR of about 99% - 4.56% = 94.44% (flooding achieves, due to Type-1 partitions, about 99% PDR only). Yet, instead of a 94.44% PDR we have a PDR of about 90%. The imperfect operation of the routing protocol itself causes an additional loss of about 4.5% of PDR.

Finally, the observations and explanations when comparing EGM to RWP also hold for the comparison between EGM and ST (Figure 5.18). Only the numbers (thus the gaps) change. Under ST, the network spends about 41/1800= 2.3% of the time in Type-1 partitions, explaining the observed gap to 100% PDR with Flooding. Using ST as mobility model, the network spends about 179/1800 = 9.9% of the time in Type-2 partitions. In brief, partitions (both Type-1 and Type-2) are worse under ST than under EGM.

5.6 Summary

In this chapter, we presented our simulation results. Those simulation results can be grouped in two sets. In the first set, we presented the results for the routing protocols: an existing one (RGR) and also our proposed enhancements or versions (Modified-RGR and Optimized-RGR). These results were obtained when using the RWP mobility model. In the second set, we presented some results in order to compare our proposed EGM mobility model with other models; namely the RWP and the ST.

The results show that our two enhancements to RGR brought about a considerable improvement on the protocol with a 5.3% increase in PDR at very low costs. The results also show that, in reality, UAANETs deal with a lot of network partitions. This was shown through simulation under realistic mobility models including our proposed EGM model and the ST

model. General conclusions to our work as well as a discussion on avenues for future work are presented in the next chapter.

Chapter 6 : Conclusions and Future Work

6.1 Conclusions

A packet delivery ratio of about 87.5% suggested that there existed a significant room for improvement in the existing RGR protocol. In that regard, RGR presents two major defaults. First, the construction of routes only considers topology information: freshness and route length in terms of hops. The selection of the routes does not take into account the fact that some links constituting them might be at the brink of breakage. As a second major default, the GGF mode in RGR does not have a strategy to salvage packets when GGF fails. In an effort to improve RGR by addressing these two defaults, we proposed a first enhancement consisting of introducing a stability criterion in the construction of routes during the route discovery in the Reactive mode of RGR. This is achieved by making use of the concept of *reliable distance*. The reliable distance between any two nodes involved in an RREQ transmission at any given time depends on the velocity and direction of those nodes. Based on that reliable distance, a decision is made on whether or not to drop the RREQ. As routes are only constructed over links that forwarded an RREQ, the discovered routes are more robust. This first enhancement results in an increase in PDR coupled with a slight drop in control overhead at no extra cost in terms of end-to-end delay.

As a second enhancement, with respect to GGF failure, we proposed a recovery strategy for Geographic routing in general that also applies to the GGF mode of RGR. The existing recovery strategies in the literature are complex, have high overhead, and are not very applicable to UAANETs. Our recovery strategy consists of forwarding the packet to the Best-Moving Node when GGF fails. The Best-Moving Node is the node, within the transmission range (the forwarding node included), that is deemed to be the best candidate node moving towards the destination. The selection in that regard is made considering the speed, the direction, the current location, and a predicted location of both the node (FN or a neighbor) and the destination node. This strategy has low complexity, low overhead, and is perfectly applicable in the context of UAANETs. Therefore, we integrated this strategy in the GGF mode of RGR. This integration

resulted in a further increase in PDR at the cost of slightly higher end-to-end delay. Yet, overall, the end-to-end delay is still relatively low. Also, as expected, the control overhead has not been affected by the introduction of the recovery strategy.

Overall, the two enhancements to RGR yielded an increase in PDR of about 5.3% (placing it at 92.8% now) and a slight drop in overhead with a very negligible cost in terms of increased end-to-end delay. The 5.3% increase in PDR is a non-trivial improvement given the fact that, in comparison, RGR (PDR of 87.5%) only yielded a 4.5% increase in PDR compared to AODV with local repair (PDR of 83%) in a similar scenario. Moreover, as we achieve higher and higher PDRs (in the 90% range), it logically becomes more difficult to deliver additional packets; therefore, an increase of 5.3% is non-negligible.

The widely used Random Waypoint mobility model is not a very realistic mobility model due to its inherent sharp turns and sudden stops that fail to capture the reality of UAVs' movement patterns. In this work, we proposed the EGM mobility model in 2D. EGM is a more realistic model based on the Gauss-Markov mobility model. The model ensures that there are neither sudden stops nor sharp turns within the simulation area; making it a more realistic mobility model. However, unlike other realistic mobility models such as ST and GM, EGM features a clear progressive boundary-avoidance mechanism that tends to eliminate sharp turns at the boundaries. In addition, EGM modifies the way direction is determined, avoiding the need to define a "mean direction" like in the GM model. Simulations showed that, compared to RWP, EGM causes a significant number of network partitions; and this has a negative impact on routing protocol performance. Another existing realistic mobility model, ST, also showed with acuity the problem of network partitioning. Therefore, we can conclude that, due to UAVs' actual constrained movement, real UAANETs deal with a non-negligible amount of network partitions, certainly compared to the use of the RWP mobility model in networks of similar densities; and this therefore needs to be taken into account when designing any routing protocol for such networks.

6.2 Future Work

Despite the fact that in our new version of RGR we have more reliable routes to begin with and we have a strategy to rescue data packets when a route breaks and GGF fails, the PDR is still significantly below the 99% achieved when Flooding is used. This tells us that there are still many packets that do not make it to the destination despite the recovery strategy. Either a refinement of the proposed strategy or a number of alternative strategies may be needed to further improve the protocol performance; which is a subject for future work.

Another item of future work is to use a limited form of flooding. For example, when GGF fails, instead of using any recovery strategy at all, we can send the packet to all (or a subset of) the neighbors up to a certain number of hops: 2, 3, etc. The number of hops can be decided after observing that on average the switches to GGF occur at a certain number of hops to the destination. We can then limit the flooding to that number of hops with a degree of certitude that they will reach the destination. Another idea would be to not switch to GGF at all and employ this limited flooding whenever the reactive route fails. However, these strategies would create additional overhead as now data packets are duplicated in the network, but the advantage is that we may get closer to the PDR performance of flooding.

We learnt from realistic mobility models that in reality, due to mechanical and aerodynamic constraints of the UAVs, UAANETs can be considerably partitioned. This suggests that holding on to the packet, as done in delay-tolerant networking schemes, for example, may help routing protocols cope with those temporary partitions. The hold-on time does not necessarily have to be constant like in our GGF recovery strategy. It might be better to adjust it so that it accounts for the actual movement of the nodes (direction and speed at a given instant). This constitutes another avenue for future work.

In the same vein of bettering routing protocols based on what we have learnt from mobility models, another future work would consist of developing mobility-model-aware or mobility-model-specific routing protocols. With those protocols, a node will be able to have a good approximate global view of the entire network. It will only require a node to know the initial location of all the other nodes as well as the movement pattern of the mobility model. It can therefore calculate a couple of approximate possible locations of any node at any given instant

during the mission/simulation. A fallback mechanism might be necessary for when a node goes down; a mechanism that will, for instance, get the information across that node is no longer present in the network at all and that it is no longer necessary to predict its possible locations.

Another direction for future research will be to extend our EGM mobility model to 3D. Fitting the parameters of the model using real flight data would also be interesting as it will further make our model closer to reality than ever. A collision avoidance mechanism like the one presented in [44] can also stand as an add-on.

We already know that the steady state distribution of nodes in the RWP model is a non-uniform spatial distribution where nodes tend to cluster around the center of the region. It would be interesting to investigate and mathematically determine the steady state distribution of nodes under the proposed EGM mobility model. That would be an additional evaluation criterion of our model. In fact, it will give us a clearer idea about the coverage achieved when the EGM model is used.

Another avenue for future work will be to evaluate the degree of randomness of EGM using an entropy rate-based measure introduced in [47]. This evaluation was conducted for some mobility models in [46]. Similar to another evaluation performed in [46], it would also be interesting to evaluate the Location Aware Routing for Opportunistic Delay-tolerant networks (LAROD) [48]-[50] geographic routing protocol with the EGM mobility model. LAROD, which is equipped with the Location Dissemination Service (LoDiS) [48]-[50], was specifically designed for Intermittently-Connected MANETs (IC-MANETs) [48], [51], and we already saw that EGM results in a lot of network disconnections.

Testing our Optimized-RGR in a real network and comparing the results with what we obtain when using the EGM mobility model in OPNET will also be a good direction to pursue.

Finally, throughout this work, we have considered the antennas of the UAVs to be isotropic. Now, one may wonder what happens when the antennas are directional for example, as increasingly non-isotropic antennas are used to reduce either the required energy to transmit data between nodes, to increase the achievable transmission range, and/or to increase the data rate. This will definitely affect our present routing protocols as, for example, not all current neighbors will be neighbors anymore due to the directionality of the transmission/reception. Therefore,

developing a version of Optimized-RGR or simply a different protocol that optimizes routing for UAANETs formed by UAVs with directional antennas is another interesting avenue for future work.

List of References

[1] R. Shirani, "Reactive-Greedy-Reactive in Unmanned Aeronautical Ad-hoc Networks: A Combinational Routing Mechanism," *Master's Thesis*, Carleton University, Canada, August 2011.

[2] "Unmanned Aerial Vehicles Classification," www.vectorsite.net/twdrn.html, [Accessed: September 2013].

[3] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, RFC2501," www.ietf.org/rfc/rfc2501.txt, [Accessed: June 2013].

[4] A. Maghsoudlou, M. St-Hilaire, and T. Kunz, "A Survey on Geographic Routing Protocols for Mobile Ad hoc Networks," *Technical Report SCE-11-03*, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, October 2011.

[5] C. Perkins and P. Bhagwat, "Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers," Comp. Commun. Rev., pp. 234–44, October 1994.

[6] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol, RFC3626," http://www.ietf.org/rfc/rfc3626.txt, [Accessed: June 2013].

[7] E. Perkins, C. Belding-Royer, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing, RFC3561," http://www.ietf.org/rfc/rfc3561.txt, [Accessed: May 2012].

[8] J. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4, RFC4728," http://www.ietf.org/rfc/rfc4728.txt, [Accessed: July 2012].

[9] Z. Haas and M. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," ACM/IEEE Trans. Net., vol. 9, no. 4, pp. 427–38, August 2001.

[10] Y. Wan, K. Namuduri, Y. Zhou, D. He, and S. Fu, "A Smooth-Turn Mobility Model for Airborne Networks," in *Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications*, ACM, pp.25-30, June 2012.

[11] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication and Mobile Computing*, vol. 2, no. 5, pp. 483-502, October 2002.

[12] Y. Wan, K. Namuduri, Y. Zhou, and S. Fu, "A Smooth-Turn Mobility Model for Airborne Networks," *IEEE Transactions on Vehicular Technology*, no. 99, 2013.

[13] J. Xie, Y. Wan, K. Namuduri, S. Fu, G. Peterson, and J. Raquet, "Estimation and Validation of the 3D Smooth-Turn Mobility Model for Airborne Networks," in *Proceedings of Military Communications Conference (MILCOM 2013)*, 2013.

[14] 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," in *Proceedings of 9th Annual Conference on Communication Networks and Services Research Conference (CNSR 2011)*, pp. 161-166, May 2011.

[15] Y. Li, R. Shirani, M. St-Hilaire, and T. Kunz, "Improving Routing in Networks of UAVs: Reactive-Greedy-Reactive," *Wireless Communications and Mobile Computing*, vol. 12, no. 18, pp. 1608-1619, December 2012. DOI: 10.1002/wcm.2333.

[16] Y. Ko and N.H. Vaidya, "Location-Aided Routing (LAR) Mobile Ad Hoc Networks," in *Proceedings of ACM/IEEE Mobicom*, pp. 307-321, October 1998.

[17] M. Aparna, M. Reza, P. Sahu, and S. Das, "An Efficient Approach Towards Robust Routing in MANET," in *Proceedings of International Conference on Communication Systems and Network Technologies*, 2012.

[18] R. Thakur, S. Sharma, and S. Sahu, "Accumulating Path Information in AODV for Ad-Hoc Network," in *Proceedings of International Conference on Computational Intelligence and Communication Systems*, 2011.

[19] Mou Zonghua and Meng Xiaojing, "A Modified AODV Routing Protocol based on Route Stability in MANET," in *Proceedings of 4th IET International Conference on Wireless, Mobile & Multimedia Networks (ICWMMN2011),* pp. 63-67, 2011.

[20] Zhao Qiang and Zhu Hongbo, "An Optimized AODV Protocol in Mobile Ad Hoc Network," in *Proceedings of 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, October 2008.

[21] E. Kranakis, H. Singh, and J. Urrutia, "Compass Routing on Geometric Networks," in *Proceedings of 11th Canadian Conf. Computational Geometry*, pp. 51–54, August 1999.

[22] M. Mauve, J. Widmer and H. Hartenstein, "A Survey on Position Based Routing in Mobile Ad-hoc Networks," *IEEE Network Magazine*, vol. 15, no. 6, pp. 30–39, November 2001.

[23] H. Frey and I. Stojmenovic, "On Delivery Guarantees of Face and Combined Greedy-Face Routing in Ad Hoc and Sensor Networks," in *Proceedings of ACM MobiCom*, September 2006.

[24] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," in *Proceedings of the 3rd ACM International Workshop on discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M 99)*, pp. 48–55, August 1999.

[25] E. Kranakis, H. Singh, and J. Urrutia, "Compass Routing on Geometric Networks," in *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99)*, pp. 51–54, August 1999.

[26] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proceedings of the 6th ACM/IEEE Annual International Conference on Mobile Computing and Networking (MOBICOM-00)*, 2000.

[27] 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 Computing and Networking (MobiHoc 2003)*, 2003.

[28] B. Leong, S. Mitra, and B. Liskov, "Path Vector Face Routing: Geographic Routing with Local Face Information," in *Proceedings of the 13th IEEE International Conference on Network Protocols (ICNP 2005)*, 2005.

[29] Q. Fang, J. Gao, and L. J. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," in *Proceedings of IEEE INFOCOM 2004*, March 2004.

[30] F. Bai and A. Helmy, "A Survey of Mobility Models in Wireless Ad-Hoc Networks," *Chapter 1 in Wireless Ad-Hoc Networks*, Kluwer Academic, 2006.

[31] M. Alenazi, C. Sahin, and J. Sterbenz, "Design Improvement and Implementation of 3D Gauss-Markov Mobility Model," in *Proceedings of the 48th International Telemetering Conference (ITC)*, October 2012.

[32] V. Tolety, "Load Reduction in Ad Hoc Networks Using Mobile Servers," *Master's thesis*, Colorado School of Mines, CO, USA, 1999.

[33] "ns-2," *the NS Manual,* http://www.isi.edu/nsnam/ns/doc, [Accessed: August 2011].

[34] "The ns-3 Network Simulator," http://www.nsnam.org, [Accessed: July 2009].

[35] "OPNET, Application and Network Performance," http://www.opnet.com, [Accessed: August 2011].

[36] J. Ariyakhajorn, P. Wannawilai, and C. Sathitwiriyawong, "A Comparative Study of Random Waypoint and Gauss-Markov Mobility Models in the Performance Evaluation of MANET," in *Proceedings of International Symposium on Communications and Information Technologies (ISCIT)*, October 2006.

[37] E. Hyytia, P. Lassila, and J. Virtamo, "Spatial Node Distribution of the Random Waypoint Mobility Model with Applications," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 680–694, June 2006.

[38] C. Bettstetter, G. Resta, and P. Santi, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 2, pp. 257 – 269, 2003.

[39] D. Broyles, A. Jabbar, and J. P. G. Sterbenz, "Design and Analysis of a 3-D Gauss-Markov Mobility Model for Highly Dynamic Airborne Networks," in *Proceedings of the International Telemetering Conference (ITC)*, October 2010.

[40] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser, " An Analysis of the Optimum Node Density for Ad hoc Mobile Networks," in *Proceedings of the IEEE International Conference on Communications(ICC)*, June 2001.

[41] B. Liang and Z. Haas, "Predictive Distance-based Mobility Management for PCS Networks," in *Proceedings of the IEEE INFOCOM 1999*, vol. 3, pp. 1377-1384, March 1999.

[42] S. Kaur, H. Singh, T. Singh, and Y. Singh, "Mobility Models In Adhoc Networks," *Journal of Global Research in Computer Science,* vol. 3, no. 11, November 2012.

[43] G. Amoussou, H. Mohanna, Z. Dziong, and M. Kadoch, "An Opnet Implementation of Gauss-Markov Mobility Model and Integration of Link Prediction Algorithm in DSR Protocol: A Cross-Layer Approach," in *Proceedings of OPNETWORK 2005*, August 2005.

[44] F. Borrelli, T. Keviczky, and G. J. Balas, "Collision-Free UAV Formation Flight using Decentralized Optimization and Invariant Sets," *in Proceedings of 43rd IEEE Conference on Decision and Control*, December 2004.

[45] A. Nayak and I. Stojmenovic, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. Wiley, 2010.

[46] J. Xie *et al.*, "Analysis of Mobility Models for Airborne Networks," *in Proceedings of the IEEE Military Communications Conference*, November 2013.

[47] Y. Wan, K. Namuduri, Y. Zhou, and S. Fu, "A Smooth-Turn Mobility Model for Airborne Networks," *IEEE Transactions on Vehicular Technology,* vol. 62, no. 6, 2013.

[48] E. Kuiper and S. Nadjim-Tehrani, "Geographical Routing with Location Service in Intermittently Connected MANETs," *IEEE Transactions on Vehicular Technology,* vol. 60, no.2, pp. 592-604, 2011.

[49] E. Kuiper and S. Nadjim-Tehrani, "Mobility Models for UAV Group Reconnaissance Applications," in *Proceedings of the International Conference on Wireless and Mobile Communications*, pp. 33-48, July 2006.

[50] E. Kuiper, "Mobility and Routing in a Delay-Tolerant Network of Unmanned Aerial Vehicles," *Master's Thesis*, Linköping, 2008.

[51] E. Kuiper, “Geographic Routing in Intermittently-Connected Mobile Ad Hoc Networks: Algorithms and Performance Models,” *Ph.D. Dissertation*, Linköping , 2012.