

A COGNITIVE PACKET PROTOCOL FOR OPPORTUNISTIC COMMUNICATIONS

Eirini V. Liotou

Supervisor: Prof. Erol Gelenbe



This report is submitted in partial fulfilment of the requirements for the degree
of
Master of Science (MSc) in Communications and Signal Processing
and the
Diploma of Imperial College London (DIC)

Department of Electrical and Electronic Engineering
Imperial College London

September 2012

Abstract

Opportunistic Networking is a very challenging and promising paradigm for mobile communications. In contrast to any other network type, opportunistic networks do not rely on the existence of an end-to-end path between the communicating entities. In addition to that, even if any paths exist, these may be highly unstable or break very quickly. Carrier nodes in this network are meeting sporadically and intermittently, but communication may still be achieved by adopting a "store-carry-forward" approach. This is only possible however at the price of additional delay.

Many routing protocols have been proposed in order to provide communication in such delay tolerant environments. Most protocols are based on the idea of building routes dynamically while packets are travelling towards their destination, without acquiring any prior knowledge concerning the network topology.

We introduce a novel protocol for opportunistic communications that differs conceptually from any other protocols introduced so far. The proposed protocol adopts some characteristics of Cognitive Packet Networks. Routing is based on the inherent intelligence of packets rather than on the nodes carrying them. Any knowledge is acquired based on packets' previous experiences, as those are reported back to a source node by acknowledgement messages. We implement this cognitive type routing protocol in an opportunistic network model where all routing decisions are based on the notion of direction, meaning the direction towards where the packet "wants" to travel.

Results obtained by this study show that not only such an approach is feasible for opportunistic communications, but also provides satisfactory network performance in terms of packet delay and delivery probability, subject to the environment under study. Consequently, this dissertation may constitute a starting point for future research in this specific field.

Acknowledgement

With this opportunity, I would like to express my deepest gratitude to my supervisor Professor Erol Gelenbe. His valuable guidance and impeccable ideas have helped me complete the -what I consider- most interesting and most thorough project of my technical career so far. Leading by example, Prof. Gelenbe has inspired and motivated me to surpass my potential and successfully deliver this very challenging and innovative work. I will always be indebted to him, for helping me turn his innovative ideas into reality.

From the bottom of my heart I would also like to thank my parents, my aunt Tata, my sister and Alexandros, who have supplied me with immeasurable courage and strength throughout this very tough academic year. Their belief in me was my source of inspiration and hope.

Finally, I need to thank God, for looking after me...

Eirini V. Liotou
September 2012

Contents

Abstract	ii
Acknowledgement	iii
Contents	iv
List of Figures	viii
List of Tables	x
Abbreviations	xi
Chapter 1. Introduction	1
Chapter 2. Literature Review	5
2.1 Cognitive Packet Networks	5
2.1.1 Introduction to basic CPN entities & operations	5
2.1.2 CPN in mobile environments and further research	7
2.2 Opportunistic Networks	8

2.2.1	The concept of Opportunistic Networks	8
2.2.2	More aspects of Opportunistic Networks	10
2.2.3	Routing protocols	11
2.3	The ONE Simulator for DTN Protocol Evaluation	12
Chapter 3. Model Description		14
3.1	Network environment	14
3.1.1	Time and Space	14
3.1.2	Movement modelling	15
3.2	Source-Destination communication	17
3.2.1	Source-Destination pairs	17
3.2.2	Types of packets	18
3.2.3	Explorers or agent packets	18
3.2.4	Payload packets	19
3.2.5	Acknowledgements	20
3.2.6	Time To Live (TTL)	20
3.3	Nodes' functionalities	21
3.3.1	Source node	21
3.3.2	Intermediate node	24
3.3.3	Destination node	25
3.4	Routing algorithm	25
3.5	More features	27

3.5.1	Multicast and Warm-up	27
3.5.2	Packet dropping policy	29
3.5.3	Energy consumption model	29
3.5.4	Packet chains	30
Chapter 4. Simulation Environment		32
4.1	Simulation environment	32
4.2	Input-Output	35
4.2.1	Input	35
4.2.2	Output	36
4.3	Basic Java classes	37
Chapter 5. Simulation results		43
5.1	Input values	43
5.2	Survey 1: Energy-related metrics	45
5.2.1	Effect of the number of users on the energy consumption	45
5.2.2	Effect of users' mobility on the energy consumption	46
5.3	Survey 2: End-to-end delay and TTL	47
5.4	Survey 3: Population density effect	49
5.5	Survey 4: Nodes' mobility effect	51
5.6	Survey 5: Predictability of the carriers' movement	52
5.7	Survey 6: Overhead in the network	54
5.8	Survey 7: Average number of hops	55

5.9	Survey 8: Effect of common routes and common sleep locations	56
5.10	Survey 9: Packet chains	57
5.11	Survey 10: Route history list	59
Chapter 6. Conclusions and Future Work		60
Bibliography		68
Appendix A.		I
A.1	JAVA classes	I
A.2	Scenario description of 5.11 (Route history list)	I

List of Figures

1.1	Store-Carry-Forward Approach [1]	1
3.1	Node A sends a packet to D via B and C [2]	25
3.2	Routing decision based on direction	27
4.1	Cognitive Opportunistic Network Simulator: nodes & packets	33
4.2	Cognitive Opportunistic Network Simulator: map pattern	34
4.3	UML class diagram	38
5.1	Energy efficiency vs. number of nodes	45
5.2	Energy consumption vs. nodes' mobility	47
5.3	Packets delivered with delay	47
5.4	Delay of packets vs. TTL	48
5.5	Average delay vs. population density	49
5.6	Delivery probability vs. population density	50
5.7	Average delay vs. area size	50
5.8	Delivery probability vs. area size	51

5.9	Average delay vs. nodes' mobility	52
5.10	Delivery probability vs. nodes' mobility	52
5.11	Average delay vs. distance-to-travel	53
5.12	Delivery probability vs. distance-to-travel	54
5.13	Network's overhead with time	54
5.14	Average number of hops with time	55
5.15	Delivery probability vs. grouped nodes	57

List of Tables

3.1	Route History list	22
5.1	General input parameters	44
5.2	Four different types of mobility	46

Abbreviations

ON:	Opportunistic Network
CPN:	Cognitive Packet Network
DTN:	Delay Tolerant Network
SAN:	Self Aware Network
MANET:	Mobile Ad-Hoc Network
AHCPN:	Ad-Hoc Cognitive Packet Network
PSN:	Pocket Switched Network
RNN/RL:	Random Neural Network/Reinforcement Learning
SP:	Smart Packet
CP:	Cognitive Packet
DP:	Dumb Packet
ACK:	Acknowledgement
CM:	Cognitive Map
ONE:	Opportunistic Networking Environment
RW:	Random Walk
RWP:	Random Waypoint
MBM:	Map-Based Mobility
SPMBM:	Shortest Path Map-Based Mobility
WDM:	Working Day Movement
QoS:	Quality of Service
GA:	Genetic Algorithm

Chapter 1

Introduction

OPPORTUNISTIC Networks (ON) or Delay/Disruption Tolerant Networks (DTN) are based on spontaneous interactions between mobile users carrying wireless devices. Such networks do not rely on end-to-end connectivity, as is the case for conventional networks, but they establish a "store-carry-forward" routing mechanism (Figure 1.1). In such types of networks, although absolutely no assumption is made on the existence of a complete path between a pair of nodes wishing to communicate, these may still be able to exchange messages, as long as proper opportunistic networking techniques are adopted.

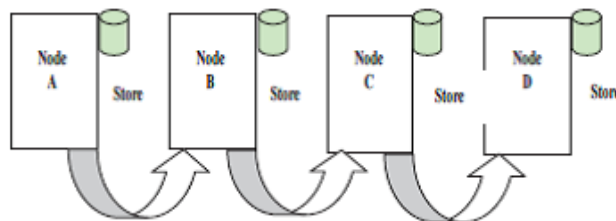


Figure 1.1: Store-Carry-Forward Approach [1]

Various challenges are raised due to both the inherent uncertainties of the connections and the impulsive nature of this kind of networks. Some of these challenges include data dissemination and routing decisions, security concerns, incentives and collaboration issues, as well as challenges concerning the cost in terms of energy, the delay, memory, network throughput, congestion, QoS and delivery probability. This research field of opportunistic communications is gaining increasing interest recently and many projects have been trying

to address these challenges, with perhaps the "Opportunistic Networking Environment (ONE)" simulator being the one most praiseworthy so far.

The purpose of this study is to design, implement, validate and evaluate a new, innovative routing algorithm based on the notion of travelling nodes' direction. To the best of our knowledge, such an approach has never been studied before. Furthermore, this proposed model for opportunistic communications attempts to exploit and properly adjust many of the Cognitive Packet Network's (CPN) features, creating in this way a new routing approach for delay-tolerant environments. This means that intelligence concerning routing decisions will be built into the packets, so that the opportunistic routing will rely on information stored inside the packets rather than being concentrated on the nodes carrying them.

Routing algorithms documented in literature so far have to do with different factors, which for instance might be related to any kind of network-context information, such as the daily mobility pattern of the users, the communities they belong to, the history of encounters between nodes, their social relationships or their social attraction, the periodicity of a specific user profile or even the amount of incentive/altruism of intermediate nodes in carrying packets for different source-destination pairs.

In this study, in order to analyse the proposed Opportunistic - CPN-enhanced network, a new stand-alone simulator has been developed from scratch. This simulator includes all the fundamental entities that comprise a network, such as nodes, packets and routes, and implements all the basic network operations. A node may be of type source, destination or intermediate (carrier) node, or it may even play all these roles at the same time, referring to different connections. As far as the basic network operations are concerned, these include but are not limited to: modelling the movement of nodes, generating payload packets aimed for a specific destination, identifying routing opportunities to other nodes and taking appropriate routing decisions, forwarding or multicasting packets according to the circumstances etc. The modelling of the nodes' movement is based on realistic every-day patterns of human mobility and contains "map" information in a way, which is

achieved by modelling the locations where users may go as well as the routes that users may follow.

Apart from these fundamental functions, the new model introduces numerous extra ones, such as: generating smart packets called agents in order to discover routes to the destination, producing acknowledgements upon the reception of a new packet so as to inform the packet generator of a successful route, keeping a record of the physical locations where a packet has been forwarded from one node to another, and taking advantage of previous packets' experiences in order to route new packets to the destination. Previous successful routes are kept in a list at the source node, they are updated whenever a new acknowledgement is received and they are scanned in order to select the best recommended path each time. This recommended path will be stored inside a newly created packet so that nodes carrying it will take advantageous routing decisions throughout the whole life of the packet. In this way, new messages will tend to be routed with significantly less uncertainty than before.

Furthermore, this model investigates the adoption of "packet chains", where packets heading towards the same destination are combined in order to update their stored recommended paths with the best available choice. This decision is based on some factors, such as the expected elapsed delay of a travelling packet and most importantly the inspection of whether a packet's recommended route is up to date or obsolete. Finally, the proposed model uses the travelling nodes' direction information in order to take routing decisions using some geometrical calculations, as it will be analysed in following sections.

As it will be demonstrated in later chapters, the performance of this network model in terms of delivery probability, delay etc. is promising, subject of course to the environment's characteristics, such as the population density in an area, the area pattern and the mobility of the nodes. Moreover, it is interesting how this model acts as a "route discovery" or "node tracking" protocol, as will be explained later on.

The simulation environment has been implemented using Java and it is capable of visualising node mobility, message generation and message forwarding in real-time using a

graphical interface. Moreover, it offers the ability to display the routes that users travel as well as their current or future physical locations. The GUI has been essential in order to validate the model in an intuitive way. The source code is properly structured and easily expandable, in case that any future enhancements are to be included.

This remainder of this document is organised as follows: Following this short introduction, a state of the art literature review is made and the well-known opportunistic routing protocols are categorised and described in Chapter 2. Moreover, the Cognitive Packet Network model is discussed. Following that, the original work of this study is divided into three chapters. In the 3rd chapter, the proposed opportunistic network model is analysed and documented in detail. The simulation environment and the algorithm used to implement this model are described in Chapter 4. Chapter 5 contains the results of several simulation scenarios, as an effort to evaluate the behaviour of this new model in terms of average end-to-end delay, delivery probability, energy consumed, network overhead etc. A simple characteristic scenario is described in the Appendix, in order to provide a better understanding of the simulation model. Finally, the last chapter contains the conclusions of this study together with some challenging ideas for future work.

This work aspires to make a genuine contribution to the field of opportunistic communications by combining cognitive and opportunistic network functionalities for the first time and by introducing an innovative direction-based routing algorithm. The model is based on assumptions and behaviours as realistic as possible, so that it may constitute a promising brand-new protocol for opportunistic communications to be used in academia or even in real-life.

Chapter 2

Literature Review

2.1 Cognitive Packet Networks

2.1.1 Introduction to basic CPN entities & operations

THE term *Cognitive Packet Network (CPN)* was coined in 1999 [3]. Cognitive Packet Networks are not conventional networks, in the sense that any intelligence is concentrated inside the packets themselves rather than on nodes carrying those packets or on high-layer protocols. *Cognitive packets (CPs)* rely minimally on routers in order to be forwarded; they use their own observations as well as the experiences of other packets travelling in the network in order to be in the position to make appropriate routing decisions. A packet's knowledge and perception of the network is stored inside its own *Cognitive Map (CM)*. Moreover, packets may be grouped together into *classes*, as long as they share similar Quality of Service (QoS) requirements, control rules etc.

In this kind of networks, nodes only serve as buffers, mailboxes and processors. Apart from the basic operations of receiving, storing and transmitting packets, nodes are also responsible for storing information of CPs into *Mailboxes* and for assisting packets to execute their code. More specifically, when cognitive packets visit a node, they read their own mailbox and may write information to other mailboxes too. This is the way in which

packets communicate with each other and improve each other's knowledge. Moreover, CPs execute the relevant code and update their Cognitive Map, which leads to an up-to-date routing decision. Through this process, packets constantly improve their perception concerning the network where they are travelling [4]. Cognitive packets may be divided in the following major categories:

- *Smart or Cognitive packets* (SPs or CPs), which embrace the functionalities of the packets previously discussed. They serve as "explorers" in this network model. Thus, their main goal is to discover the optimum route to a given destination or to seek new destinations, as well as to update existing information.
- *Acknowledgement packets* (APs or ACKs), which are generated once a packet reaches its destination. These packets travel back to the source following the reverse of the route recorded by smart packets, and their goal is to update the source with up-to-date valuable routing information. For that matter, ACKs also make use of mailboxes when they are "parked" on a CPN router.
- *Dumb packets* (DPs), which take advantage of knowledge previously acquired, in order to select the best possible routes for packets. A single decision will concern all dumb packets of the same QoS class. SPs and DPs actually transfer user-related data, while ACKs carry routing and delay information. DPs are the only payload carriers.

CPNs are very powerful, in that they are able to dynamically adjust to changing network conditions. For instance, experiments have shown that the CPN algorithm is capable to quickly discover new routes with less experienced delays, in case that e.g. the main traffic flow suddenly encounters significant delays and losses. Finally, it has been demonstrated, that the CPN is highly effective to adapt to different QoS requirements [5].

A Reinforcement Learning algorithm based on Random Neural Network (RRN/RL) is used for making routing decisions. Each node creates a specific RNN with as many neurons as outgoing links for each Source-Destination pair and for each QoS goal. The RNN weights associated to neurons are increased/decreased as a reward/punishment for the neuron's contribution to the fulfilment of a QoS goal.

CPN networks may be considered as one paradigm of the more generic *Self Aware Networks (SAN)* [6]. SANs differ drastically from conventional networks, in that routing information is not spread throughout the whole network so that this will be already available whenever a need occurs. On the contrary, any routing updates are initiated only by those nodes that need this information and only when they need it. Thus, all nodes act autonomously in discovering paths, tailored to their specific QoS criteria. In that sense, CPNs are self-aware networks, able to translate each node's objectives into real-time adaptive decisions. Another type of SANs is the "*Ant Colony*" paradigm, inspired by the use of pheromones by ants in order to mark their travelled paths and communicate with other colony members.

2.1.2 CPN in mobile environments and further research

In [7], a CPN-type extension to *Mobile Ad-Hoc Networks (MANET)* is discussed, as a solution to the challenging QoS issue in such networks of frequently changing connectivity conditions. This Ad-Hoc CPN (AHCPN) approach requires each node in the network to keep an up-to-date list of its neighbours, simply by listening to the radio channel. Moreover, each node is required to keep and dynamically update routing information concerning any communication pair that it is aware of. Route discovery is initiated on demand: when a node needs to discover a new route, it will either run the RNN/RL algorithm or initiate a broadcast, if sufficient information is not available.

[8,9] propose an innovative energy-efficient solution for AHCPNs, capable to handle the scarcity of resources in such mobile environments. The metric *path availability* has been introduced, as the probability to find nodes and links available for routing across a path. This probability is based on the remaining battery life of the intermediate nodes, causing nodes with longer remaining life to be preferred against others for packet forwarding. This metric may be combined with the need to reduce the end-to-end delay or even the mutual interference of adjacent communications, thus increasing the network throughput.

In [10] the application of the CPN algorithm in Voice over IP is examined. CPN needs to address the requirements of real-time voice applications regarding delay, jitter and

packet loss. Thus, a proper reward function needs to be defined, which will incorporate an averaged estimate of the forward delay from a node to the destination as well as the fluctuation in the delay, defined as "interarrival jitter".

The authors in [11] investigate the network performance when three different QoS goals are prescribed, i.e. when three different reward functions are defined. The Algorithm-D sets the delay as the QoS goal, the Algorithm-H uses the hop count and finally the Algorithm-HD a combination of both. Experiments conducted in an autonomous CPN testbed reveal that for highly congested networks, algorithms insisting on selecting the shortest path are not recommended in terms of delay, whereas for lightly loaded networks the least number of hops is the best choice.

A very extensive comprehensive survey of the CPN's variations, applications and experimental performance evaluations may be found in [12].

2.2 Opportunistic Networks

2.2.1 The concept of Opportunistic Networks

The terms *Opportunistic Networks* and *Delay Tolerant Networks* are often used interchangeably. In this type of networks, no assumption can be made concerning the availability of an end-to-end path between a source-destination communicating pair [13]. In fact, disconnections are rather the norm than the exception. New routing techniques and system architectures need to be deployed for this network model, so as to deal with the inevitable disruptions in communications, which cannot be handled by traditional protocols. For that matter, a *store-carry-forward* approach is used.

Opportunistic Networks were introduced for the first time in [14], as an innovative direction towards pervasive computing. A new network model was proposed, created by devices that overcome their "linguistic barriers" so that they become able to communicate with each other and self-organise in order to realize a common goal. The ON grows from its *seed*, and gradually forms an expanded network by inviting other *helpers* to participate in it. The

authors envisage that such a type of network may be used in order to handle bottlenecks or gaps in communications, as for instance to provide relief in emergency situations. As it is stressed though in this study, "*Pervasive computing has pervasive problems, not the least of which are interoperability, security and privacy*", implying that these issues need to be seriously addressed before Opportunistic Networks become reality. Finally, the challenge of providing proper incentives or reinforcements to nodes invited to participate in the network is emphasized. In this study though, no reference is yet made concerning the disruption or delay tolerance of this new type of networks.

In Delay Tolerant Networks, the energy consumption is a very sensitive issue. In human-to-human networks like *Pocket Switched Networks (PSN)* or even in networks employed for environmental and wildlife monitoring, the energy is constrained and thus should be only carefully wasted. Nevertheless, due to the inherent instabilities and intermittent connections characterizing the DTNs, the more energy is consumed for packet forwarding the larger are the probabilities for successful packet delivery. The authors in [15] try to address this issue by formulating an optimization problem where the message delivery ratio is to be maximised and the energy consumed is to be constrained. For that matter, they introduce a continuous-time Markov framework to model the message dissemination in DTN, where nodes are able to continuously exchange messages (not only at discrete timeslots). Both a *two-hop forwarding* and a *probabilistic epidemic forwarding* algorithm are implemented and different kinds of forwarding policies are investigated. Results show that by following a *threshold dynamic policy*, where the transmission probability is 1 before a time threshold and 0 afterwards, the best network performance is achieved.

An interesting application of ONs is wildlife monitoring. The "ZebraNet" collects information concerning the behaviour and interactions of wild species. The animals under study are carrying special devices with sensing capabilities, which are able to collect data concerning other animals encountered and transmit this information together with their own to specific fixed or mobile base stations. Another application of ONs is to provide Internet accessibility to developing countries or nomadic populations, thus helping bridge any communication gap [16].

2.2.2 More aspects of Opportunistic Networks

Human-to-human networks like PSNs have a social nature, which may be taken into consideration when making routing decisions. This perspective has been introduced in [17], where real human mobility traces have captured human social characteristics that include:

- The existence of *communities* or *clusters*, determined by the familiarity and regularity of contacts among users.
- The *centrality* or *betweenness* of a node, based on its location across paths between communicating entities.
- The *popularity* of a node, implying that popular hubs should be preferred over unpopular ones.

Using the finding of *heterogeneity* among nodes and based on the argument that "*social relationships vary more slowly than the topology*", the authors propose new routing protocols such as BUBBLE, where users are not statistically equivalent in terms of delivering messages to a destination. Another social characteristic, studied in [18] is that of *altruism* in opportunistic communications. The limited resources of hand-held devices on energy and memory, as well as the potential security risk on carrying messages from other nodes make altruism an important factor in the performance and robustness of the network. Various altruism models have been studied: for instance, the "Community-biased" model assumes that people belonging in the same community have higher incentive to help each other, while the "Degree-biased" model relates the centrality and popularity of a node with its altruistic behaviour to help others. According to this survey, when the altruism of participating nodes is taken into consideration when designing the network, this will be more efficient.

Recent studies based on empirical data concerning nodes' contacts [19, 20] provide a new direction towards the design of mobility models and routing protocols as well as on performance analysis. These studies define two new metrics: the *inter-meeting* or *inter-contact time* of two nodes, as the interval during which two nodes are not in contact but 'see' each other at the end of this interval and the *contact time*, as the period of existent com-

munication between them. These two metrics are very crucial concerning the experienced end-to-end delivery delay in the network or even the feasibility of such a network. More frequent contacts imply more often encounters among nodes, and thus more data-relay opportunities. Similarly, larger contact durations imply the forwarding of more packets and thus influence the capacity of the network.

According to these studies, it may be both empirically and theoretically proven that the inter-contact time of two independent mobile devices in a domain with finite boundaries always yields an exponential distribution. This is the consequence of nodes "bouncing back" to a bounded area once they hit the border, thus shortening the interval between contacts. On the contrary, mobility in an unbounded, open-space domain produces power-law type distribution for the inter-meeting times, since nodes are high likely to draw away from each other. Furthermore, it is found that the existing dichotomy in the distribution of inter-contact time takes place at a characteristic time in the order of half a day, which could be attributed to the daily periodicity of human behaviour [20].

2.2.3 Routing protocols

Routing protocols in Opportunistic Networks may be classified in various ways. First of all, depending on whether they exploit some form of infrastructure to opportunistically forward messages or not they may be categorised to networks *with* or *without infrastructure* respectively [16]. Furthermore, they may be classified into context-oblivious, mobility-based and social context routing protocols:

- *Context-oblivious* routing protocols are 'naive' in the sense that they do not take into account any user-related information (communities, mobility patterns etc.) when making routing decisions. *Flooding* (blindly forwarding all packets to all neighbours) or *epidemic routing* (forwarding only non-common packets) belong to this category. So does the *Spray and Wait* protocol, which performs a controlled replication of packets (spraying) into the network, creating thus sufficient diversity to explore the network efficiently.
- Some well-known *mobility-based* routing protocols are the: *PRoPHET*, *MV* (Meetings

& Visits) and *MaxProp*. According to these routing mechanisms, packets are forwarded to nodes with higher "delivery predictability" to the destination, as this is determined by the frequency in which a node meets another as well as the paths visited by those nodes. This information is available using past observations and nodes' experiences. The *MOBYSPACE* protocol only forwards packets to nodes with mobility patterns similar to that of the destination, whereas *solar-hub* protocols exploit information concerning the destination's frequently visited places.

- *Social-context-based* protocols not only take into account the mobility pattern of the users, but also the social relationships among them, node-specific profiles and the history of their past encounters. Nodes are able to exchange data during their contacts, so as to become aware of the context attributes of their potential message destinations. Routing is then based on selecting nodes that show high similarity to this context. Furthermore, "spatio-temporal" context information may be used, namely the periodicity in the activities of carriers in terms of location and time of day. The *HiBOp*, *Propicman* and *SpatioTempo* routing protocols belong to this category.

2.3 The ONE Simulator for DTN Protocol Evaluation

The *Opportunistic Networking Environment Simulator (ONE)* has been designed with the objective to evaluate DTN routing and application protocols. As such, it is modelling: node movement, contacts among nodes, routing and message handling [21].

We may compare the ONE simulator with the simulator created for the purposes of this study. Both support a mobility pattern of nodes, message generation, packet exchange, a routing protocol, a basic notion of energy consumption, visualisation and statistical output, identification of a node's neighbours etc. Moreover, in both simulators performance metrics include delivery probability and latency. Nevertheless, the ONE simulator, being the product of many years of research, includes many more features, such as numerous synthetic mobility models, mobility models based on real-world traces, as well as numerous DTN routing schemes. ONE is open source and thus is often enhanced by researchers or

other users worldwide.

The most well-known mobility models, which are implemented in this simulator, are:

- The popular *Random Walk* (RW) and *Random Waypoint* models (RWP) [22].
- *Map-Based Mobility* (MBM) models, where nodes are constrained to move inside predefined routes of a map, but their travel-to destinations are selected randomly. The *Shortest Path Map-Based Mobility* (SPMBM) model instructs users to select paths with the least distance to travel.
- The *Working Day Movement* Model (WDM) is a human-behaviour based model, and hence implements the three major activities of users inside a realistic opportunistic network, i.e. their 1) sleeping at home, 2) working in the office and 3) going out during the evening [23].

At this point, we should mention that the movement modelling implemented in the current thesis is a combination of the MBP and WDM, in that nodes do follow a working-day model so that their travel-to destinations are not randomly placed in an area. At the same time, nodes are only allowed to move inside predefined routes, as in the map-based mobility model.

The routing protocols included at the moment in ONE are the following: Direct Delivery, First Contact, Spray-and-Wait, PRoPHET, Max-Prop and Epidemic, and the possibility to add new routing protocols is present.

Surveys performed using the ONE simulator have confirmed the intuitive conclusion, that the performance of an opportunistic network highly depends on the routing protocols considered and on key factors such as the population density, the distance between two communicating entities and the mobility of the nodes. Moreover, it is of extreme importance whether and how well any design assumptions match the reality of the current environment under study. Similar conclusions have been drawn from the current study, as it will be thoroughly analysed in following sections.

Chapter 3

Model Description

IN a previous section, the proposed model for Opportunistic Networks' communications has been introduced and the basic framework has been described. The next step is to provide detailed specifications of this model together with its components and their inherent features, as well as to thoroughly analyse all algorithms involved. The model has been designed with the objective to be as realistic as possible while keeping its complexity within a reasonable level. The described model has been implemented exactly as documented below and more clarifications on this simulation will be discussed in the next chapter.

3.1 Network environment

3.1.1 Time and Space

In order to study and evaluate the proposed CPN-type routing algorithm, an Opportunistic Network Simulator has been developed from scratch. This simulator models a geographically restricted area of configurable size, inside which mobile nodes are constrained to move and interact with each other. The area is considered empty-space and open-air. The amount of nodes is adjustable and their initial positions are selected randomly, so that they tend to be almost uniformly distributed inside the whole area. The network can be

observed for any predefined amount of time, input by the user before the simulation starts.

3.1.2 Movement modelling

The nodes (also referred to as users, devices or mobiles) follow a specific movement pattern. We can distinguish three different types of nodes depending on their movement pattern. Specifically, nodes may move following a working-day movement model, a "random" movement model or may be moving in some central, busy routes.

- **Working-day movement pattern:**

These nodes represent users following a realistic, predetermined movement pattern, defined by three different physical locations: a user's house, its office and a place of interest such as a super-market, bar, shopping-mall etc. During a day, users start moving from their houses, then go to their offices and finally visit a place of interest before returning back to their houses. When they visit their next location, they stay there for a period of time, called *sleep time*. This is a rationally selected value drawn from a normal distribution, in the sense that users tend to have similar (but not identical) working-day movement patterns. This pattern is repeated by the nodes periodically throughout the whole simulation. Users travel on their unique "user-specific" paths in order to go from one place to another. This pattern is a simplification of the model described in [23].

- **Central routes' movement pattern:**

This pattern tries to capture the characteristics of the traffic in central highways and junctions, which serve to connect various parts of a large area, as in real-life. These central avenues are always occupied by a significant amount of users. The speed of the nodes travelling in these routes is associated to the routes themselves. This means that users travelling on the same route will have some average speed value, but with some deviation.

- **Random movement pattern:**

This pattern is followed by a limited number of users (around 5%) and in a real world may represent the movement of taxis, delivery of goods, tourists etc. Nodes moving in

that way may sometimes help discover isolated parts of the modelled area. Each node's movement can be totally defined by four parameters: a. direction, b. speed, c. distance to travel and d. sleep time (or wake up time). More specifically, every single node randomly selects the direction in which it will be travelling (in reference to a hypothetical x axis), the distance it is willing to cover and finally its speed.

The position of a node is totally defined by a pair of (x, y) coordinates. These positions are re-calculated per second, which is the smallest unit of time in the simulation. In real life, these positions could be acquired approximately, using for instance a built-in GPS or they could be estimated in reference to a Base Station or a Wireless Access Point. Given that a node starts moving from a well-known initial location, its exact position at any point in time can be directly estimated using its direction and speed.

The speed of a node may range from a minimum to a maximum value, modelling in this way the existence of pedestrians, various types of vehicles, or conditions constraining a vehicle's speed. As soon as a node has reached its travel-to destination, it pauses for a rationally selected period of time. The node remains still for as long as this "sleep time" lasts, until woken up to travel again according to re-determined movement-specific parameters. Nodes moving on central highways or moving randomly draw their sleep time values from a uniform distribution, whereas nodes following the working-day model select this value from a normal distribution as already mentioned before.

The existence of static physical locations such as houses, offices and places for activities, as well as the existence of predefined routes (central highways or user-specific routes) and central junctions create the notion of a map. For instance, the concentration of many routes, junctions or many buildings in an area indicates a central location, whereas the lack of them indicates an isolated area. (We would expect that the latter areas achieve less packet delivery ratio). The number of houses etc. is dynamic, and depends on the number of users. More than one user are grouped together in houses, offices and places of activities, creating in that way also the notion of communities and social relationships. In that sense, people in the same house are family or neighbours, people in the same office

are colleagues, and people in the same activity may be friends. Finally, as a simplification, we do not model the interior of these buildings and we consider them "dimensionless".

We assume that no mobile devices are switched on or off, so that the population remains fixed throughout the whole time the network is observed. Moreover, we assume that no node ever leaves the predefined geographical area, by "bouncing back" any user that has reached the border at any time. This could be alternatively considered as nodes leaving the area and simultaneously new nodes entering it.

3.2 Source-Destination communication

3.2.1 Source-Destination pairs

Throughout the whole simulation, every mobile node is equally likely to have new packets ready for transmission. The frequency in which nodes produce new packets is limited by a predefined message generation probability and each node takes a new decision every second about whether to create a new packet or not. We concentrate our study on source-destination pairs that follow the working-day model.

As it also happens in more realistic network environments, a user actually transmits a series of packets comprising a *message*. To be more specific, a node decides whether it has a new message ready to be sent, and if this is the case, a burst of packets will be sequentially transmitted. Messages may continue to be created until the device's queue buffer is full. Then the node has to pause for a while and wait for opportunities to transmit the existing queued packets before any new ones may be generated.

Furthermore, in order to keep the model realistic enough, as well as for reasons that will be revealed later (and have to do with buffer sizes), each node has a predefined limited *Contact List*. This list of contacts contains all potential destinations, to which the source node is willing to transmit packets. We restrict the users from sending messages to people in the same building, as these would be immediately received by them. Our objective is

to avoid such direct packet exchanges, so as to evaluate the model correctly.

Each node in the network has a specific *ID number*, which is serving as an IP address. The source and the destination ID of a new packet are stored inside its header, so that all intermediate nodes will keep relaying a packet that does not belong to them and so that the destination will recognise that a packet was destined for it.

3.2.2 Types of packets

The proposed Opportunistic Network (ON) model incorporates features similar to the ones met in the Cognitive Packet Network (CPN). For instance, the proposed ON model makes use of the three major types of packets found in CPN, but with adjusted characteristics and behaviour so as to address the special needs of the challenging, constantly-changing environment of opportunistic networks. The types of packets met in this enhanced ON model are the following:

- *Explorers or agents*, used for route discovery (similar to CPN's smart packets).
- Source routed *payload packets* (similar to CPN's dumb packets).
- *Acknowledgements* (ACKs).

3.2.3 Explorers or agent packets

As in CPN, these kinds of packets act as explorers for specific source-destination pairs. A new agent is created every time that a source node wants to send a packet to a destination without having any prior reliable information about how to route the packet towards it. These explorers are actually not inherently intelligent, since they are simply being multi-cast from one node to others every time such opportunities occur. So, unlike CPN's smart packets, ON agents are not routed based on experiences of previous packets. Opportunistic environments are highly spontaneous and uncertain, so ON agents are just charged with the task of discovering the most current physical location of the destination. This discovery is accomplished in a purely random manner. As it will be described later in more

detail, the sources also send such packets periodically to the network, destined for their contact list's entries (i.e. common packet targets), as a way to be proactively informed about their up-to date locations.

Agents collect precious information while travelling through the network. In particular, they save and carry a path of all the physical locations where their forwarding from one node to another took place. The first entry of this path is always the physical location of the source node. Then, every time that a packet is routed to a new node, this node is entitled to append the current path with the node's current approximate position. In the fortunate case that such an agent packet reaches its final destination, the destination will be familiarized with the physical route that this packet has travelled and will also append its position at the end of this path. This path will thus look like this: $\{(x_s, y_s), (x_A, y_A), (x_B, y_B), \dots, (x_D, y_D)\}$, where "S" stands for *Source*, "D" for *Destination* and any other letter for *Intermediate Node*. We are going to examine how this information is taken advantage of when we describe the functionalities of each node type.

Apart from this physical path, each packet is similarly updated with the timestamp when an exchange occurred. The source will first enter the time when the packet was created (t_S) and the destination will be the last to enter the time this packet arrived (t_D). Consequently, this path will be of the form: $\{t_S, t_A, t_B, \dots, t_D\}$. Finally, a counter of the number of hops that this packet experienced is kept.

3.2.4 Payload packets

Payload packets embrace all the aforementioned functionalities of the agents, in the sense that they also record the physical path travelled, together with the relevant timestamps and hop count. However, these packets are neither routed randomly nor multicast. On the contrary, they make use of the knowledge previously acquired by successful explorers (i.e. explorers that reached their destination as long as an acknowledgement successfully informed the source about this event). Thus, payload packets attempt to follow a predefined path, or better, some predefined intermediate hops on their way to the destination.

Details on how the information about previous successful paths is organised at the source node and how this information is used when making routing decisions will be described in a following section.

Payload packets are the only packets that carry useful information that the source actually intends to send. In terms of size, these will be the largest ones compared to the agents and the acknowledgements. Agents and ACKs constitute inevitable overhead in the network, and as such, their size needs to remain as small as possible. This is the reason payload packets are never randomly multicast in this network model, and explorers are always used instead.

3.2.5 Acknowledgements

Acknowledgements are generated by the destination each time an agent or a payload packet arrives, and they head back to the source attempting to follow the reverse of the route recorded inside the arriving packet.

The role of these packets is twofold. First of all, they inform the source that a packet has been successfully received by the destination. Similarly, the lack of an ACK within a certain amount of time will imply that a packet never reached its target. The source might decide to retransmit a packet using this information (not implemented at the moment). The second role of an acknowledgement is to notify the source about a successful, up-to-date physical path of a payload packet or explorer towards the destination, together with the end-to-end delivery time, called *delay*. For this reason, ACKs store and carry on their way back to the source the originally recorded physical route and delay involved. In this way, payload packets will be able to learn a successful route to the destination together with the expected delay.

3.2.6 Time To Live (TTL)

All three types of packets have a predefined TTL value, which is currently implemented as a timer in seconds. Another approach may have been to use the maximum allowed

hop count as the TTL value. This value should be relatively large for payload packets and acknowledgements and it should have a reasonable value depending on the end-to-end packet delay or equivalently to the network's diameter.

In the current model, agents are actually not forwarded but replicated, increasing in that way the possibilities of encountering the destination. Taking that into consideration, the TTL of agents should have a relatively small value. When a new replica of such a packet is created, the TTL could either be reset to its initial value or just inherited. In the first case, there will be smart agents alive in the network for a much longer time, updating in that way the source constantly about an updated position of the destination, if this is encountered. However, the burden in the network would be much higher in that case than if the current TTL value was simply copied from one node to another (inherited) instead of being renewed.

3.3 Nodes' functionalities

First of all, any node may act as a source, destination or intermediate node at the same time, referring to at least three different source-destination connections. Source nodes are capable of generating payload/agent messages aimed for a specific destination and the destination is able to identify them. Destination nodes produce ACKs to travel back to the source that is also able to identify these types of messages. All intermediate nodes are able to exchange packets when encountering other nodes while moving inside the modelled area. Apart from these quite obvious functionalities, each node type has much more advanced capabilities, as described next.

3.3.1 Source node

Every potential source node keeps a *Route History List* or *Route Repository* in its memory, which contains a predefined number of the most recent successful routes per destination. Since a node may only send packets to its Contact List entries, it allocates the necessary

memory space for each of them. Assuming that the allowed route entries per destination are e.g. 10 and that there are "n" possible destinations/contacts, this route history list will look like this:

Table 3.1: Route History list

Dest.	Timestamp of most recent ACK	Path	Delay
1	t_1	Route1: $\{(x_s, y_s), (x_{A1}, y_{A1}), (x_{B1}, y_{B1}), \dots, (x_{D1}, y_{D1})\}$	t_{d1}
		Route2: $\{(x_s, y_s), (x_{A2}, y_{A2}), (x_{B2}, y_{B2}), \dots, (x_{D1}, y_{D1})\}$	t_{d2}
	
		Route10: $\{(x_s, y_s), (x_{A10}, y_{A10}), (x_{B10}, y_{B20}), \dots, (x_{D1}, y_{D1})\}$	t_{d10}
2	t_2	Route1: $\{(x_s, y_s), (x_{A1}, y_{A1}), (x_{B1}, y_{B1}), \dots, (x_{D2}, y_{D2})\}$	t_{d1}
		Route2: $\{(x_s, y_s), (x_{A2}, y_{A2}), (x_{B2}, y_{B2}), \dots, (x_{D2}, y_{D2})\}$	t_{d2}
	
		Route10: $\{(x_s, y_s), (x_{A10}, y_{A10}), (x_{B10}, y_{B20}), \dots, (x_{D2}, y_{D2})\}$	t_{d10}
...
n	t_n	Route1: $\{(x_s, y_s), (x_{A1}, y_{A1}), (x_{B1}, y_{B1}), \dots, (x_{Dn}, y_{Dn})\}$	t_{d1}
		Route2: $\{(x_s, y_s), (x_{A2}, y_{A2}), (x_{B2}, y_{B2}), \dots, (x_{Dn}, y_{Dn})\}$	t_{d2}
	
		Route10: $\{(x_s, y_s), (x_{A10}, y_{A10}), (x_{B10}, y_{B20}), \dots, (x_{Dn}, y_{Dn})\}$	t_{d10}

Upon reception of an acknowledgement, the source node reads its content in order to isolate the originator of this message (column: "destination"), the recorded path of coordinates that was progressively created while the corresponding payload/agent packet was travelling to its target (column: "path"), as well as the delay experienced by this packet (column: "delay"). This new route might then be inserted in the 1st position as "route 1", the most recent entry. The rest of the entries will be shifted down by one, causing the oldest and thus most obsolete entry to be lost.

A route reported by an ACK will replace an already stored route only if the timestamp that this new ACK was created at the destination's side is more recent than the one stored in the column "timestamp of most recent ACK", which always corresponds to the most recent route 1. The ACK will then be deleted, as no longer useful. However, there is a possibility that the new ACK reports that the destination is still in the same location reported by a previous ACK. In that case, the source is going to examine whether this new ACK has recorded a fastest route, i.e. a route towards the destination with less

experienced delay. If this is the case, this new route will be the 1st entry of the Route List, no matter if the corresponding ACK was not the one most recently created by the destination. For instance, if ACK1 and ACK2 were created at clock time 100sec and 90sec respectively and it took them 40sec and 30sec to reach the source respectively, then at time 140sec the most recent ACK1 is received. However, this should not replace the entry already created at time 120sec by the fastest ACK2. Finally, if a route is identical to one already stored, it is ignored.

A path will be stored inside the Route History List after undergoing some processing. In particular, any loops of physical locations will be removed, meaning that if a packet was located in the same neighbourhood more than once, any in between physical location entries will be removed. As an example, if the recorded path looks like: {A B C **D** E F G **D** H I}, where each letter corresponds to a different neighbourhood, then this path will be first processed and then stored as: {A B C D H I}. The "same neighbourhood" is identified as a set of coordinates that could be clustered together, because the distance between them is less than the communication range.

The Route Repository is going to be accessed by the source node in two different circumstances. First of all, this list will be scanned each time that the source is creating a new packet, in order to search whether a relevant, reliable route to this destination is already available. If the search has a positive outcome, the recommended route will be stored within the new packet, with the hope that if followed, there are high possibilities that the packet will manage to reach its destination. In this way, uncertainty is significantly reduced. If this search does not reveal an available path to follow, an agent will be multicast, as already explained in previous sections and so the payload packet will go to *pending state*.

The second case that this list is accessed is every time that a new entry is created due to an ACK arriving, and all packets in pending state are scanned for possible matches. If a recommended path is found, the packets will be moved to the queue buffer, waiting to be transmitted with the first opportunity. All payload packets are also provided with

information concerning the expected delay of this route (used in packet chains as will be discussed later).

The best recommended route will be selected out of all the routes available (here 10). The criterion is a combination of how recently the route was stored together with its corresponding recorded delay, providing some QoS in this way. To be more precise, all routes will be compared to the most recent one as far as the recorded coordinates of the destination are concerned, and in case that two or more paths would lead the packet to the same approximate target location, the one that needed the least time to get there will be preferred. In real life, we could think of this case as the existence of two or more different routes to arrive at the same location; the fastest of them should be recommended.

If the source node is moving, it makes sure that all the recommended routes are updated with its latest position, in real-time. We are going to visualise this using GUI screenshots in a following section.

3.3.2 Intermediate node

Any intermediate node carrying a packet related to a particular source-destination connection will be called a *carrier* or *holder* of the packet. A carrier is able to identify the existence of other nodes inside its communication range, simply by listening to the radio channel; these nodes will be called *neighbours*. Encounters with neighbours represent routing opportunities and it is the carrier's responsibility to decide whether it is most advantageous to forward a packet or not, based on a well-defined routing algorithm. A very simple representation of the role of intermediate nodes is shown in Figure 3.1.

In general, the proposed routing algorithm works as follows: Each packet carrier identifies its neighbours at every time instant while travelling through the network and it isolates the "best candidate" for handing over the packet. Alternatively, the carrier may conclude that the most beneficial decision is not to forward the packet at all, and as a result it keeps carrying it itself. This decision is based on a *notion of direction* as will be explained thoroughly in the following section.

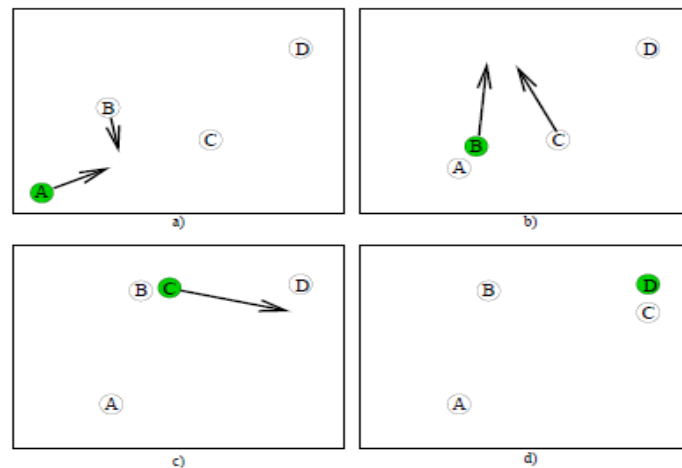


Figure 3.1: Node A sends a packet to D via B and C [2]

3.3.3 Destination node

The destination node is responsible for creating acknowledgement packets upon reception of a payload or agent packet. For that matter, it extracts any necessary information and uses it in order to create an ACK back to the source. Apart from the payload, useful information includes: the recorded physical path, the source-to-destination delay and the source node's ID. The ACK message will be prescribed to use the reverse of the original path recorded on its way to the source, after undergoing a similar to the source's loop-removal processing. Finally, the received packet will be deleted.

3.4 Routing algorithm

We are now going to describe how routing decisions are made each time that a packet carrier encounters new neighbours. At this point, we assume that every node is familiar with each of its neighbour's moving directions. This could be possible for instance by combining the "Direction of Arrival" with the power of the received signal from the neighbour at two successive time instances, as long as a device was equipped with the required hardware. Another way to achieve that would be the existence of a *negotiation protocol* between the two nodes, which could be used so as to exchange information relative to their positions or direction of movement, or even in order to let two neighbours agree whether they are

willing to co-operate in the first place (concept of altruism). None of these two options is currently implemented in the model; this direction-related information is considered already known.

Let's assume that the recommended path of a packet is $\{A B C D E\}$, where A is the current position of the source and E the current position of the destination. Each letter represents a pair of coordinates together with the surrounding area of a radius equal to the communication range of a mobile device. While the packet carrier is moving, it constantly scans the surrounding area for new encounters. The node then examines this subset of nodes one by one by checking the following:

- Among the subset of encountered neighbours, which one (including itself) is currently travelling on a direction higher likely to lead the packet to its next target hop (i.e. to position B if the packet's previous location was A, to C if its previous location was B and so on)? Then the node forwards the packet to the best candidate. Note: If a neighbour is not moving at the moment, it is not a candidate for getting the packet.
- Is the best candidate moving inside any target area B or C or D? If yes, update the recommended path to: $\{C D E\}$ or $\{D E\}$ or just $\{E\}$ respectively.

In the next figure we depict exactly how routing decisions are made. Let's assume that node N1 carries a packet for the "Dest.". While N1 is travelling, it encounters N2; the question is whether it should forward the packet to N2 or not. N1 is able to estimate 3 different angles related to its movement:

1. The angle relative to the x axis, which corresponds to its travelling direction, ω_1 .
2. The angle relative to the recommended intermediate next hop (here Hop1), θ_1 .
3. The difference between these two angles, ϕ_1 .

Moreover assuming that N1 knows N2's position and travelling direction, it may also estimate ω_2 , θ_2 and ϕ_2 . The packet will be forwarded if $\phi_2 < \phi_1$ (which is the case here), meaning that N2 is moving in a direction higher likely to reach Hop1. The " ϕ " angles

may be considered as the deviation between the node's actual direction and its direction if it was travelling towards the next hop.

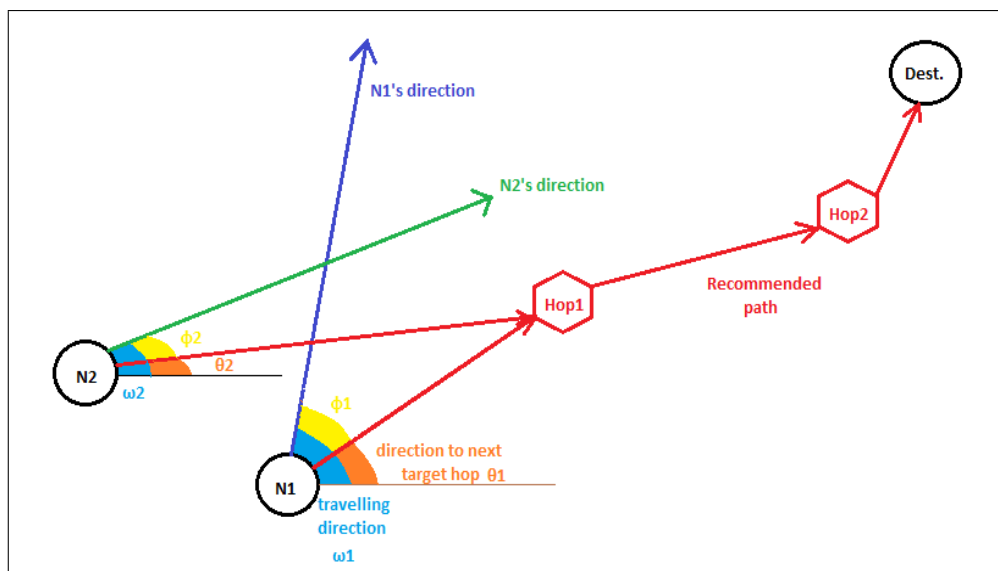


Figure 3.2: Routing decision based on direction

Similar analysis is possible if N1 encounters more than one nodes at the same time. Then, N1 will calculate all involved ϕ angles and the packet will be forwarded to the node with the smaller angle ϕ .

We have made one more assumption at this point. If the destination of the packet is a carrier's neighbour, then the packet will be delivered to it directly. This could be achieved for instance by broadcasting the packet to all its neighbours once the final target area $\{E\}$ is reached. If that wasn't the case, other neighbours could get the packet and randomly move it back and forth with limited probabilities for the packet to actually ever reach its destination (as validated using the simulation's graphical output).

3.5 More features

3.5.1 Multicast and Warm-up

Each node in the network is a potential packet originator, and as such, it needs to discover routes to all of its potential destinations. In order to make sure that a node has up-to-date

information about these routes, it initiates periodic multicasts to all of its contacts. Each node keeps a separate *multicast timer*, so that the network is not suddenly overwhelmed by all nodes' agent packets simultaneously. The contribution of such multicasts will be significant in the case that a node hasn't sent a packet to a target node for enough time for that node to change its location radically. However, if that target is not moving or not moving so fast or if the originator sends payload packets regularly enough, initiating a multicast will prove to be redundant.

A multicast may be initiated in two more cases. First of all, every time that a node following the working-day movement pattern reaches its next location (house, office or activity location) it forces a multicast to all its potential packet destinations. In this way, existing routes will be updated and future packets will be routed based on more reliable information.

The second case that a multicast is initiated is when the source node has not received ACKs for a significant amount of time. More specifically, a user keeps track of the time that the last packet was generated and of the time that the last ACK was received. If the interval between these two values is considered significant (e.g. relevant to twice a packet's TTL), the source probably has obsolete information relative to the destination, which may have moved to a new location, yet unexplored by the source node. A multicast will thus help overcome such issues.

As already mentioned before, multicast is the process of replicating a packet to a limited number of neighbours each time, with the constraint that neither the originator nor a node that already has a replica of this same packet is going to accept it again. More specifically, a node is not going to keep two agent packets of the same source-destination pair, since this would be unnecessary and redundant.

When the simulation starts, i.e. when the network has just started to operate, we allow a *warm-up* period, which is nothing more than an initial multicast from all nodes to all possible destinations. Nodes do not produce payload packets during that period, thus allowing routes to be discovered beforehand with some probability.

3.5.2 Packet dropping policy

Each node in the network has a limited buffer for storing all types of packets. Consequently, some packets will need to be dropped as soon as this buffer is full. Payload and ACK packets have a priority over agent packets, and most recent agent packets also have a certain priority over older ones. So, each time that a new packet is forwarded to a node of which the buffer is almost full, it will decide to drop the oldest agent packet it carries. In reality, since payload and agent packets have different sizes, a node should drop as many agents as necessary in order to accommodate a new payload packet.

It should also be mentioned at this point, that there is a certain threshold of the percentage of payload packets in "pending state" that a node may create. We limit this to an 80% of the total buffer size, so that this node will still be able to accept ACKs referring to its connections. Otherwise, if the buffer of a source node was 100% full with payload packets (which are never dropped - they may only expire), the ACK messages would never reach this node, causing the packets in pending state to never find a route and never be sent.

3.5.3 Energy consumption model

In order to be able to draw some conclusions concerning the energy efficiency of this Opportunistic Network model, a very simplified, yet useful, energy consumption model has been introduced. The energy cost is measured in dummy units, using the following guidelines:

- Each encounter between two neighbours, actually comprising a routing opportunity, costs 1 energy unit. This cost is attributed to the process of discovering the neighbours and to any communication between them either causing packet forwarding or not.
- Additionally to identifying a routing opportunity, a transmission-reception of a packet costs 2 extra energy units (1 for transmission and 1 for reception).
- Finally, the processing of a packet by the destination or of an ACK by the source

costs 0.5 energy units. The same energy amount is spent while creating "packet chains" (defined next), if any.

3.5.4 Packet chains

Payload and ACK packets travelling to the same destination may be *chained* together as long as they are located on the same carrier at some point in time. Chaining implies that all those packets will be afterwards treated as one when routing decisions take place, so that they will keep travelling inseparably.

The process of chaining is performed by replacing each node's originally recommended path to follow with a better one. For instance, if there are 3 packets heading to the same destination and their recommended paths are P1, P2 and P3 respectively, then if the best path recognised is e.g. P2, all packets are going to use P2 to reach the destination. As a result, all routing decisions will be taken collectively for these packets from that point on. If these 3 packets then find themselves to co-exist with a 4th packet following an even better route, say P4, their *paths to follow* will again be updated.

Selecting the best path among many depends on the nature of the network environment as well as on the goals imposed by the users. For instance, if the criterion is high delivery probability, the best decision would be to select that path that was most recently created (i.e. more reliable). If the criterion is to minimise the total number of nodes needed to carry the packet, the best decision would be to choose the path with the least number of hops required. Finally, if the criterion is to minimise the distance expected to be travelled, the decision would be made based on that. Of course, all these decisions are only based on "hope", and not on a "promise" for successful delivery.

The current model combines all these capabilities. So, chaining may be based on the shortest expected distance to reach the destination. An alternative more sophisticated approach has also been implemented: When some packets are going to be chained together, it is checked whether they all report that the destination is at the same final location. If not, the packet that carries the most recent, i.e. most reliable information will be used

to replace all other packets' recommended routes. If they all report that the destination is at the same location, then the best path will be found as a combination of the number of required intermediate hops to reach the destination and the expected elapsed delay in order to get there. The elapsed delay is calculated as:

$$\textit{expected elapsed delay} = \textit{total expected delay} - (\textit{clock} - \textit{time packet created})$$

where *clock* is the current time and the *total expected delay* was stored into the packet when that was created. This elapsed delay is important when for instance a packet knows how to get to the destination via a route of higher speeds.

Agents are never chained together, neither with payload packets nor among them, so that they keep moving randomly in order to explore the whole area independently.

In the next chapter we are going to present the simulation environment that implements the proposed model exactly as documented here. Following that, we are going to run multiple simulation scenarios in order to study the behaviour of this model.

Chapter 4

Simulation Environment

IN the previous chapter we have described the proposed system model in detail. In this chapter we are going to present the simulation environment that implements this model exactly as documented in Chapter 3 and which can be used to provide us with valuable statistical information. Following that, several simulation scenarios are going to be performed using this environment and all the results will be thoroughly presented and analysed.

4.1 Simulation environment

The Opportunistic Network Simulator has been implemented using Java version 7 (update 5). The simulator's output is depicted below at an arbitrary point in time. In order to implement this simulator some basic principles of *Java Game Programming* have been used [24,25].

As depicted in Figure 4.1, the basic entities that comprise the output of this simulator are:

- A restricted geographical area represented by the Java board (frame) of specific dimensions, as shown in this figure.

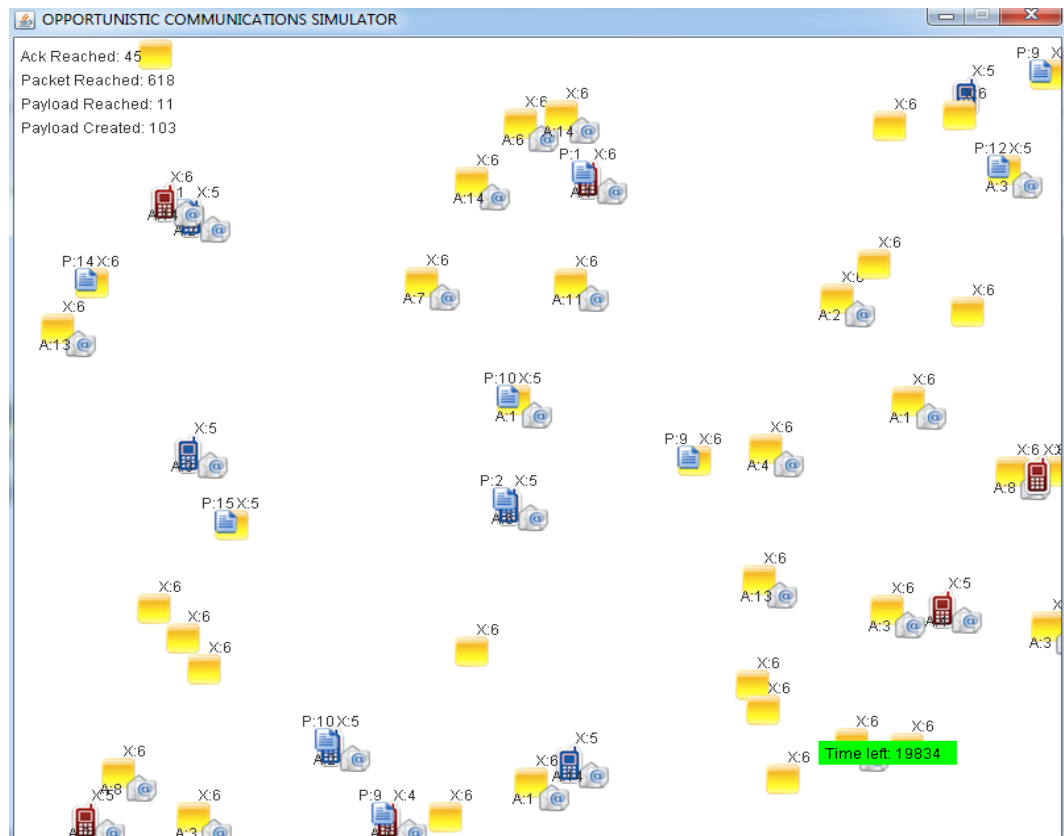




Figure 4.1: Cognitive Opportunistic Network Simulator: nodes & packets

- Packet generators and packet destinations represented using blue and red "mobile" icons respectively.
- Intermediate nodes, represented using yellow square icons. All three types of nodes are moving inside the board, and this movement can be actually visualised throughout the whole simulation time. Note: packet generators and destinations also serve as intermediate nodes, as shown in Figure 4.1.
- Packets (P), acknowledgements (A) and explorers (X). A node may be carrying any or all of these different types of packets, and their amount is also displayed in the basic frame. Packets are represented by the "message text" icon  and acknowledgements by the "envelope" icon . For agent packets no icon is used. Packets may be transferred from one node to another and this transition can be also visualised in real-time.
- On the top left corner of this Java frame, some output related to the arrival of packets

is constantly updated, and finally on the bottom right corner the time remaining for this simulation to end is presented (in seconds).

If now we suppress the graphical output concerning packets, ACKs and agents and enable the map-related output like routes and buildings, we may get the following (for a few initialised nodes):

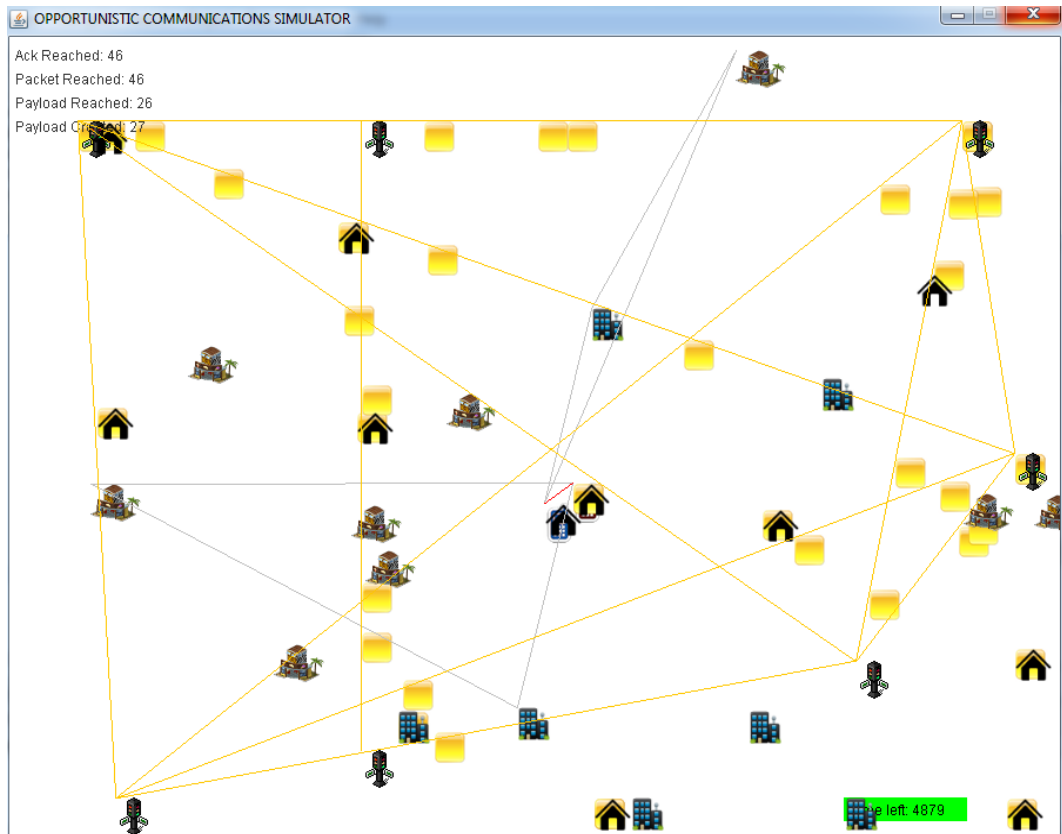






Figure 4.2: Cognitive Opportunistic Network Simulator: map pattern

This time we may observe the following entities:

-  represents a house, where many users may live.
-  represents an office, where many users work together.
-  represents a place of any activity (bar, mall etc.).
-  represents junction interconnecting routes.
- The orange lines in Figure 4.2 represent the central routes where nodes may move.

- The grey lines represent the working-day movement pattern of nodes.
- The red lines represent the up-to-date recommended path (explained in Chapter 5).

This graphical output is drawn using the painting tools provided by the Java 2D API, a set of classes for advanced 2D graphics and imaging. The real area size (in meters) is scaled to fit this window of 900x700 pixels. This output is extremely useful both for visualisation of the network environment in real-time as well as for testing purposes.

4.2 Input-Output

4.2.1 Input

The user may change the values of any of the following parameters, so as to simulate and test the network environment under different conditions. These input parameters include:

1. *simTime*: The simulation time, or in other words the period of observation of the network's behaviour.
2. *numUsers*: The total number of user nodes existing in the network.
3. *edge*: The edge of the geographical area under study (square area).
4. *range*: The communication range of each mobile device.
5. *minSpeed* - *maxSpeed*: The speed of the nodes may range between a minimum and a maximum value (as long as the node is not in sleeping state).
6. *minRouteSpeed* - *maxRouteSpeed* - *stdRouteSpeed*: Refers to the speed of nodes moving in central routes. The values are drawn from a normal distribution defined by these three parameters.
7. *minDistance* - *maxDistance*: A user moving randomly is allowed to travel any distance ranging between these two rational values.

8. *minWakeUpTime* - *maxWakeUpTime*: These parameters correspond to the minimum and maximum time that a user is allowed to stay in sleeping state. It refers to users moving on central routes or moving randomly.
9. *meanWakeUpTime* - *stdWakeUpTime*: The wake up time of nodes following the working-day pattern are drawn from a normal distribution with this mean and standard deviation.
10. *minMulticastPeriod* - *maxMulticastPeriod*: Each node performs periodic multicast every time that the relevant timer expires.

Finally, anyone running the program may wish to alter the following: a) the node's memory size buffer, b) the number of contacts in its Contact List, c) the packets' TTL values, d) the maximum allowed number of multicasts per user per second, e) the message length f) the message generation probability per user per second and g) the percentage of users following the working-day model or moving in central routes. Moreover, the user may change the number of users grouped per house/office/activity. Also, he/she may activate or deactivate the periodic multicast, the creation of packet chains and the graphical output (for testing purposes). Finally, the user may restrict the total allowed number of sources and destinations to a predefined value.

4.2.2 Output

Apart from the graphical output, console output is also used for gathering performance metrics' values. These estimated metrics are:

1. Total number of packets sent and total number of packets that reached their destination. The corresponding delivery ratio is also output.
2. Total number of payload packets sent and successfully received, as well as the average delivery probability.
3. The number of packets dropped due to buffer size limitations.
4. The amount of packets that expired due to TTL.

5. How many packet chains have been created during the whole simulation time, as long as chaining is active.
6. The average end-to-end delay, which refers strictly to payload packets.
7. The total energy consumed, in "energy units", according to the simplified energy consumption model described in the previous section.
8. The overhead of the network.
9. The average total number of hops a packet has passed through.
10. The percentage of payload packets that reached their destination while that was moving.
11. The percentage of payload packets that were exchanged directly in the case that the source and destination accidentally met (should be kept small).

4.3 Basic Java classes

Some major Java classes have been created so as to simulate the proposed Opportunistic Network model with cognitive capabilities and they are depicted in the UML class diagram of Figure 4.3. One may observe the associations between all those classes (represented by the arrows) but the classes are displayed as "black boxes". Nevertheless, in Appendix A one may find all the implemented classes, each one presented with all its attributes and operations. These 9 classes together with some of their most characteristic methods are summarised below:

- **Skeleton:**

This is the entry point of the simulation. It configures and draws the basic board that represents the network environment and it contains the `main()` method. It is also responsible for terminating the application.

- **SIM:**

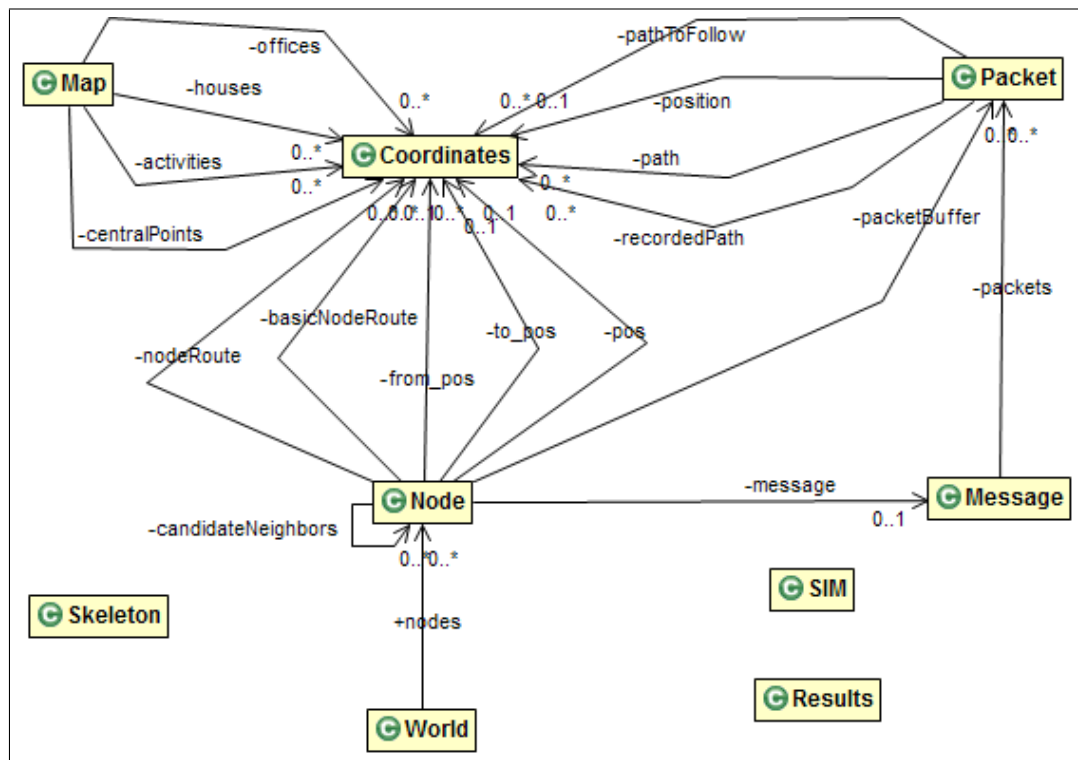


Figure 4.3: UML class diagram

This class is related to the simulation's input parameters. The user is requested to enter values to some input parameters before the simulation may start.

- **Results:**

This class is responsible for saving, processing and printing in the console output any metrics relative to the network's performance.

- **World:**

Inside this class all nodes are created, initialised with their original positions and given a unique ID number (via the `initNodes()` method). Moreover, new source-destination pairs are created and new messages are generated. The "World" is responsible for progressing the time of the simulation and for instructing the nodes to move inside the network area. Moreover, it loops through all nodes and their packets, and guides intermediate nodes to keep transferring any packets they are carrying, source nodes to process any acknowledgements they received and destination nodes to process any payload or agent packets. Additionally, this class manages all periodic multicasts and orders the update of each packet's TTL. Finally, the "World" is responsible for drawing the graphical output of this application, using Java's `paint()` method.

- **Map:**

This class keeps all the information concerning the location of houses, offices and activities as well as all the information related to the central routes (junctions connecting routes and mean speed values per route).

- **Message:**

A message consists of a dynamic number of packets, created by the source and ready to be transmitted once contact opportunities occur. Once a packet is sent, this is removed from the message's list; once all packets are transmitted, a source node may generate a new message with some probability.

- **Packet:**

A packet's characteristics are: its Source's ID, its Destination's ID and its type (ACK, agent or payload). A packet also carries the following information: a) the physical path of coordinates it has followed so far that is progressively updated every time it is forwarded to a new node, b) the corresponding timestamps, c) the recommended path to follow, if any and d) the expected delay. If the packet is of type "ACK", it also contains e) a recorded path to be carried back to the source node and f) the corresponding recorded delay. A packet is always travelling within a node, so its position is automatically updated with its carrier's coordinates via the `updatePosition()` method. Finally, every time that a packet has managed to reach an intermediate target hop, the `updatePathToFollow()` method is appropriately updating the recommended path.

- **Node:**

This is the most crucial class of this application, as it implements the majority of the algorithm's logic. A node's most important attributes are the following:

- A unique ID number,
- Movement-related parameters (speed, direction, sleep time),
- Position related parameters (initial, current and travel-to position),
- A node's house → office → activity route, if the working-day model is used,

- A node's starting position, if this is moving in central routes,
- A timer related to periodic multicasts,
- A packet buffer of limited size,
- A route history matrix (since any node may serve as a packet generator),
- A contact list,
- The last time instant that a node created a packet and received an ACK,
- An icon representing whether a node is of type source, destination or packet carrier (or all 3).

The most characteristic methods of a node, which have been already thoroughly described in Chapter 3, are:

- Movement related methods:
 - `createNodeRoute()`: Sets a node's working-day movement pattern.
 - `createCentralNodeRoute()`: Sets a node's path in a central route.
 - `createPath()`: Sets a node's next travel-to destination (predetermined).
 - `move()`: Moves a node to its new current position (updated per second).
 - `setSpeed()`: Sets a node's speed (from uniform or normal distribution).
 - `setWakeUpTime()`: Sets a node's "sleep" (or wake up) time.
- Methods related to nodes acting as packet generators (sources):
 - `createMessage()`: Creates a message consisting of multiple packets.
 - `createPacket()`: Prepares each one of the message's packets for transmission.
 - `processACK()`: Processes a newly arrived acknowledgement.
 - `processRoute()`: Removes the loops from a recorded path of coordinates.
 - `saveRoute()`: Saves a processed route into the Route History List.
 - `selectRoute()`: Supplies a new packet with a recommended route to follow.

- `setExpectedDelay()`: Provides a new packet with the information of expected delay for this recommended route.
- Methods related to nodes acting as packet recipients (destinations):
 - `processPacket()`: Processes an agent or payload packet that has just arrived.
 - `createACK()`: Prepares an acknowledgement to travel back to the source.
- Methods related to packet forwarding:
 - `readTarget()`: A carrier reads the target coordinates that the packet should reach next, on its way to the final destination.
 - `transferPacket()`: Implements the whole packet forwarding logic.
 - `multicastPacket()`: Refers to agent packets, that are replicated to a limited number of neighbour nodes whenever there is a new opportunity.
 - `setPacket()`: Performs the actual packet forwarding, i.e. moves the packet to its new carrier.
 - `delPacket()`: Deletes a just forwarded packet from its previous carrier.
- Methods related to routing decisions:
 - `isNeighbor()`: Identifies the current neighbours of a travelling node.
 - `findCandidate()`: Finds the best candidate to forward the packet to.
 - `isTarget()`: Checks whether this candidate was a target hop so as to update the recommended path, if necessary.
 - `estimateDirection()`: This function calculates the direction of a moving node relative to the target destination (using geometrical operations).
- More functionalities:
 - `startPeriodicMulticast()`: Instructs a node to start multicast of agents in order to explore new routes.
 - `checkIfIdle()`: Checks if a node has not received an ACK for a previously sent packet for a long time.

- `createContactList()`: Each potential packet generator creates its own contact list of possible destinations.
- `dropSmartPackets()`: Implements the dropping packet policy.
- Packet chains:
 - `createPacketChain()`: Groups packets travelling to the same destination so as to search if packet chains can be created.
 - `createPathChainShortest()`: Creates packet chains by updating all recommended paths with the one using the shortest route in terms of distance.
 - `createPathChainRecent()`: Creates packet chains using a combination of expected elapsed delay, route length and most recent information.

- **Coordinates:**

This class is simply used for grouping x and y coordinates' values together into pairs, referring either to nodes' or packets' coordinates.

Chapter 5

Simulation results

FOLLOWING the design and implementation of the proposed Opportunistic Network model that was analysed in the previous chapters, we are now going to test its behaviour by running multiple test scenarios and by recording performance related metrics. In this way not only can we validate the rationality of this model but we can also evaluate its performance. General conclusions on the usefulness, contribution and enhancements of this model will be discussed in the final chapter of this document.

5.1 Input values

Several surveys have been performed and will be described next, each one focusing on unique system's characteristics. In order to achieve that, we are going to keep under our control as many input parameters as possible, so that any output will be a function of the parameter under study each time. In this way, conclusions on dependencies between input parameters and performance metrics will be more reliable. Each survey contains several runs so that the recorded results are representative average values. However, each value acquired from one replication is already "averaged" in a way, since the network model is allowed to run for sufficient time and since all the nodes behave independently; thus each replication already provides characteristic output values. In the next table all input

parameters' values (as discussed in Chapter 4) are presented.

Table 5.1: General input parameters

Input parameter	Numerical value
Message generation probability	0.1
Max number of entries in a node's buffer (memory size)	20-60
Max number of pending packets at source	16
Max number of multicast copies per user per sec	2-5
Number of Route History List entries	20
Number of Contact List entries	1-5
Payload/ACK packet TTL (in sec)	200-2000
Agent TTL (in sec)	inherits value from payload/ACK
Message Length	1 to 5 packets
Simulation time (in hours)	2-6
Total number of user nodes	30-700
Square area edge (in meters)	500-1000
Communication range (in meters)	40-60
Min-Max speed for working-day movement model (in m/sec)	1 to 4
Speed for central routes	4 to 7 with std. deviation of 1
Min-Max distance travelled (in meters)	edge/4 to edge/2
Sleep time for working-day model (in sec)	60-1500
Min-Max sleep time at junctions (in sec)	20-60
Percentage of users following the working-day model	30-40%
Percentage of user traffic in central routes	55-65%
Percentage of users following random pattern	5%
Users grouped per house	5
Users grouped per office	10
Users grouped per activity	10
Number of central junctions	7
Warm up period	TTL value
Mean multicast period	relevant to source nodes' sleep time

The input parameters range between the above values and are properly adjusted per survey, so that specific characteristics of the model are highlighted. All the values have been carefully selected in order to serve realistic cases but at the same time trying to keep the simulation time of a single test into reasonable levels.

5.2 Survey 1: Energy-related metrics

5.2.1 Effect of the number of users on the energy consumption

In this survey we examine the impact of the number of nodes on the total energy consumed inside the network for as long as this is observed. Instead of presenting the total energy consumed in dummy "energy units", we will estimate the fraction of the energy used for transmission/reception and for processing operations over the total energy consumed. This information will give us an insight on how energy efficient the network is as a function of the users' number.

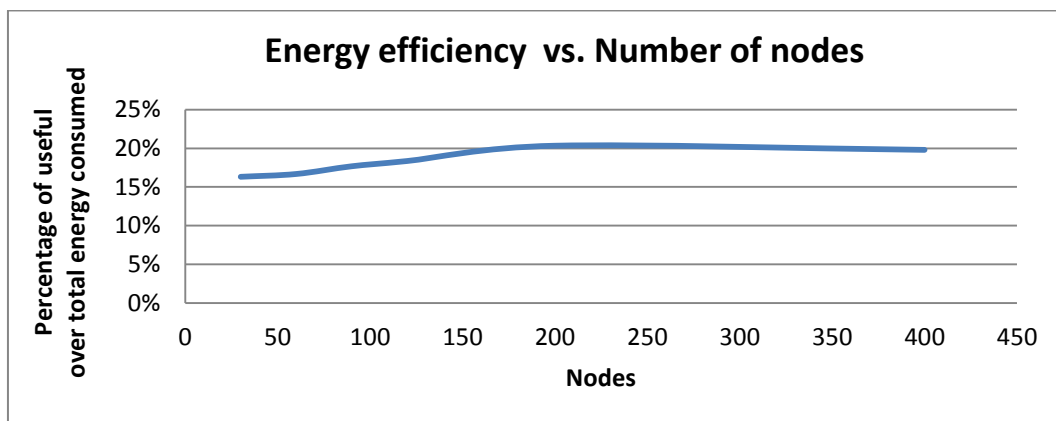


Figure 5.1: Energy efficiency vs. number of nodes

As it can be observed from this figure, as the number of nodes increases, the fraction of energy consumed on "useful operations" (transmission, reception and processing) over the total energy required for the network to operate slightly increases. This behaviour can be interpreted as follows: On the one hand, an increase in the number of nodes also increases the number of encounters among them and consequently increases the energy wasted on communications that may not lead to exploited routing opportunities. On the other hand, more encounters will lead to more exploited routing opportunities among nodes as well (i.e. more transmissions/receptions) and to more processing operations. The observed behaviour reveals an interesting result. If we consider this measured fraction of:

$$\text{energy fraction} = \frac{E_{\text{transmission,reception}} + E_{\text{processing}}}{E_{\text{transmission,reception}} + E_{\text{processing}} + E_{\text{encounters}}}$$

we realise that when the number of users increases, the number of useful operations (transmission reception, processing) increases more significantly than the number of non-useful ones (unexploited encounters), thus causing this described effect. This means that the network adapts to the increasing number of users in a very efficient way. However, we also observe a certain threshold of around 20%, meaning that this energy fraction is not going to increase forever.

5.2.2 Effect of users' mobility on the energy consumption

The users' mobility may be influenced by two parameters: 1) their speed (in m/sec) and 2) their sleep time duration (in sec) as defined in Chapter 4 for all types of nodes. We are going to study four different types of mobility, ranging from very slow to very high mobility, as shown in the next table:

Table 5.2: Four different types of mobility

Parameters Mobility	Central routes		Working-day		Central routes		Working-day	
	Min sleep	Max sleep	Mean sleep	Std. dev. sleep	Min speed	Max speed	Min speed	Max speed
lower	100	1500	2000	20	1	2	0	1
low	100	1000	1000	20	1	4	1	3
high	50	500	600	20	2	4	2	4
higher	10	100	400	20	3	6	2	5

By running the simulation using each one of these 4 mobility patterns, we get Figure 5.2.

All the values have been normalised over the highest energy value referring to the "higher mobility" case. In this way, it is clearly displayed how much more energy is consumed when users keep moving faster and faster. Higher mobility causes more encounters, i.e. more contacts, more opportunities for transmission/reception and consequently more frequent packet exchanges and packet processing. We are going to study how mobility affects other performance metrics such as end-to-end delay and delivery probability in the following sections.

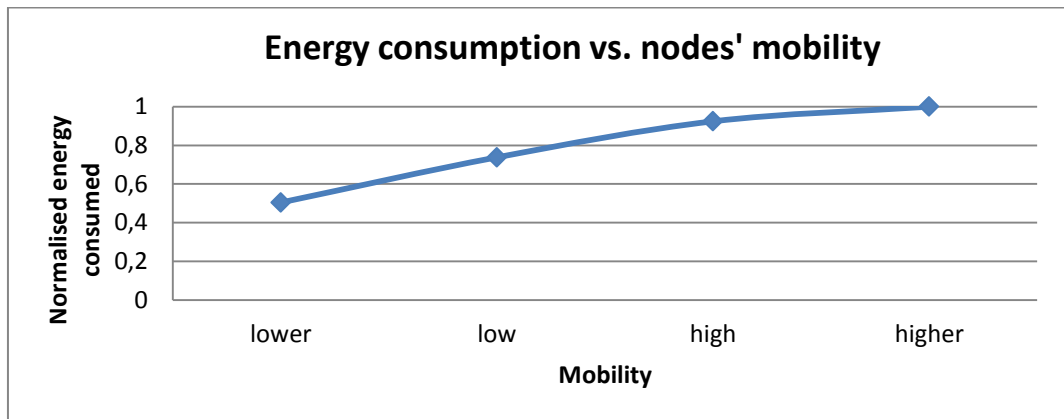


Figure 5.2: Energy consumption vs. nodes' mobility

5.3 Survey 2: End-to-end delay and TTL

For the purposes of this survey we are recording the delay with which all packets are arriving to their destination. The results are presented in the next figure:

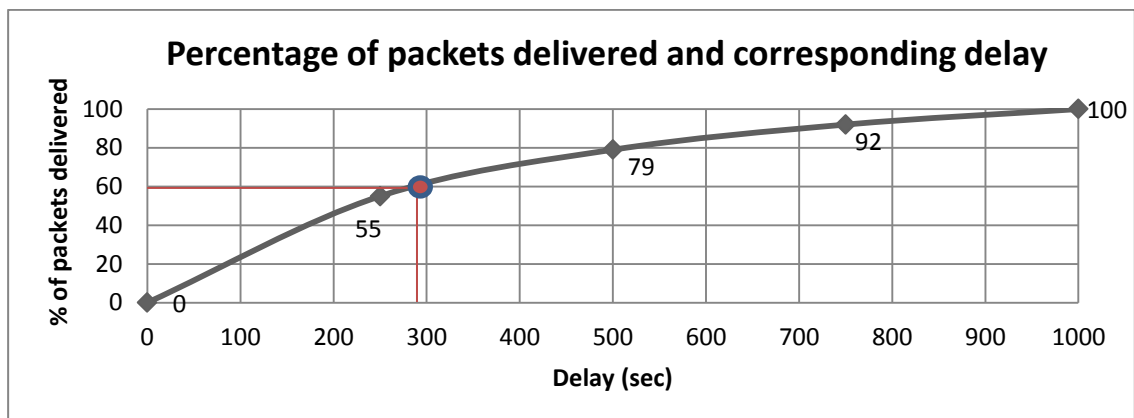


Figure 5.3: Packets delivered with delay

The y axis ranges from 0%-100%, but it should not be confused with the overall delivery probability metric. This axis actually represents the packets indeed delivered to their destination relevant to their delay to get there. The delay is measured from the moment a packet is generated (even if this is buffered at the source waiting to be transferred) up to the moment it reaches its destination.

The average delay here is around 290sec (corresponding to the 60% y axis value). Note: it does not correspond to the 50% value, as in this case the percentage of packets delivered with delay less than 290sec is 60%. This graph provides some information concerning

the deviation from the average delay, i.e. that: 55% of the packets are delivered to their destination within 250sec, 24% (55%-79%) are delivered with a delay between 250 and 500sec, 13% (79%-92%) with delay 500-750 and finally only 8% (92%-100%) with a delay of 750-1000sec. The TTL value in this case was 1000sec which explains this upper limit on the delay of the delivered packets. A delay equal to zero corresponds to packets exchanged when source and destination were instantly within communication range while moving, which occurred at around 0-3% in this experiment.

This figure also demonstrates how the TTL value affects the maximum delay allowed. By performing some surveys on how the selection of the TTL value affects the delay and the delivery probability we get Figure 5.4.

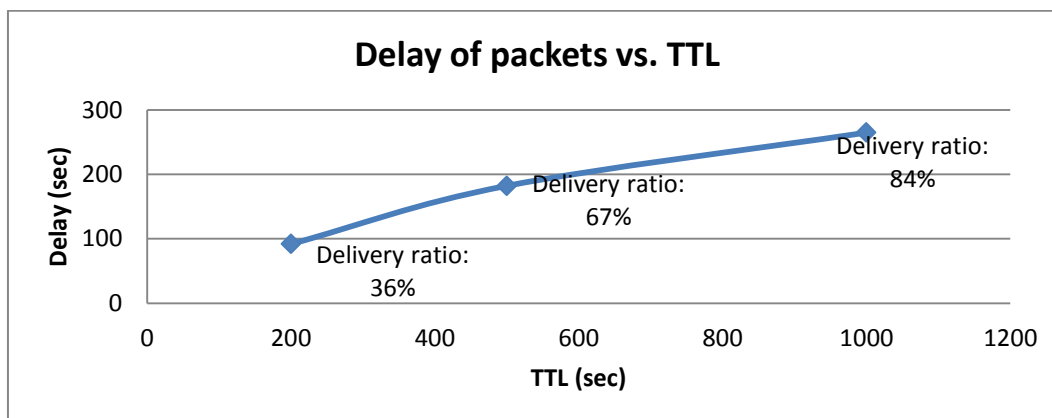


Figure 5.4: Delay of packets vs. TTL

As it might have been expected, by allowing the packets to live longer (larger TTL), then these are more likely to finally reach their destination. However, the delay will be much higher and in the same time the total buffer occupancy will be larger (and potentially packets will be dropped or not carried).

The purpose of this survey is to show the significance of the TTL value, which should be carefully selected based on the user's goals and the network's specific characteristics. For instance, in networks with high congestion the TTL should be kept to relatively low values. Besides, messages delivered after some specific time may be considered obsolete by their sender. On the contrary, in networks where the delivery ratio is very important at any cost, the TTL value should be large or even unlimited.

5.4 Survey 3: Population density effect

The purpose of this survey is to study the impact of the intermediate nodes' population density on the delay experienced by the packets as well as on their delivery probability. In order to purely and objectively study the effect of changing the number of intermediate nodes on the network performance, we are restricting the number of source-destination pairs to a predefined value. Under these conditions, we get the following results:

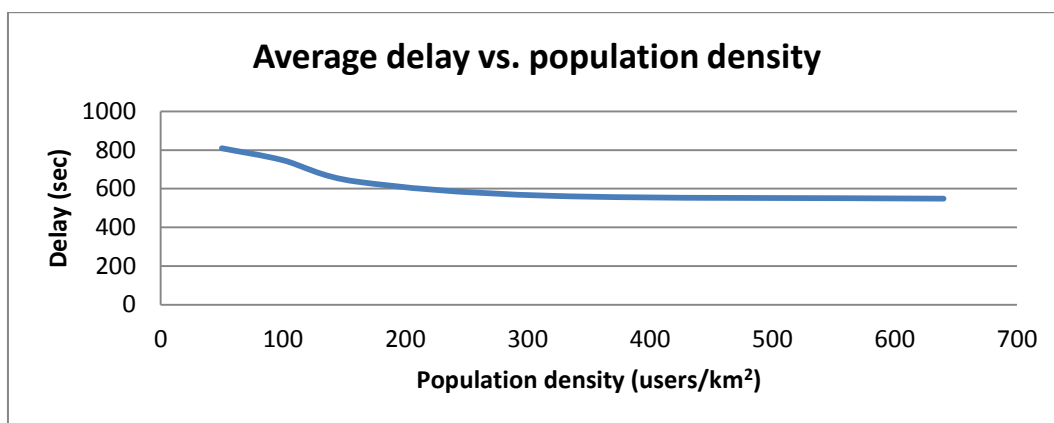


Figure 5.5: Average delay vs. population density

The population density is defined as the total number of users over the whole geographical area (in km^2). So, for instance a density of $320\text{users}/km^2$ may correspond to a population of 80 users inside a $500m \times 500m$ square area. As an indication, UK's population density is around $240\text{users}/km^2$ and a normal value for a metropolitan area would be $800\text{users}/km^2$.

In Figure 5.5 we observe an expected behaviour, namely as the number of intermediate nodes increases, the average delay decreases. This happens due to the fact that there will be more potential carriers inside the network and thus there will be much higher probability to encounter a node travelling on a more suitable direction towards the target destination. However, the decrease in the delay value is not too significant, since as long as packets are indeed delivered to their destination the total travelled distance by this packet as well as the speed of the carriers will not differ significantly. Finally, as already discussed in Survey 2, the value of the delay is closely connected to the TTL value, i.e. the maximum delay allowed by the system for communications (2000sec in this scenario).

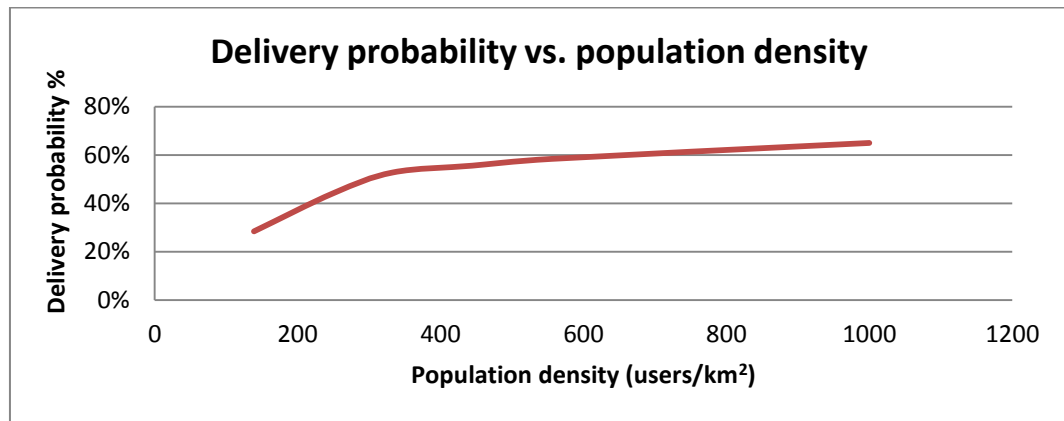


Figure 5.6: Delivery probability vs. population density

Furthermore, we would expect the delivery ratio of the packets to be larger as the population density increases, which indeed is the case, as shown in Figure 5.6. Packets are less likely to expire or to be "lost" as a result of being transferred by a carrier node to a non-desirable location inside the area, since as the population density increases there will be more chances to hand over the packets to more suitable carriers. Moreover, if more carriers are available, the total network capacity is larger, so that there are fewer chances for packets to be dropped.

By using the population density metric, we are trying to show that the delivery probability is not strictly dependent on the number of users or the area size separately, but on their combination. Nevertheless, the delay will significantly vary with the area size, since packets may have to travel much longer distances in order to reach their destination.

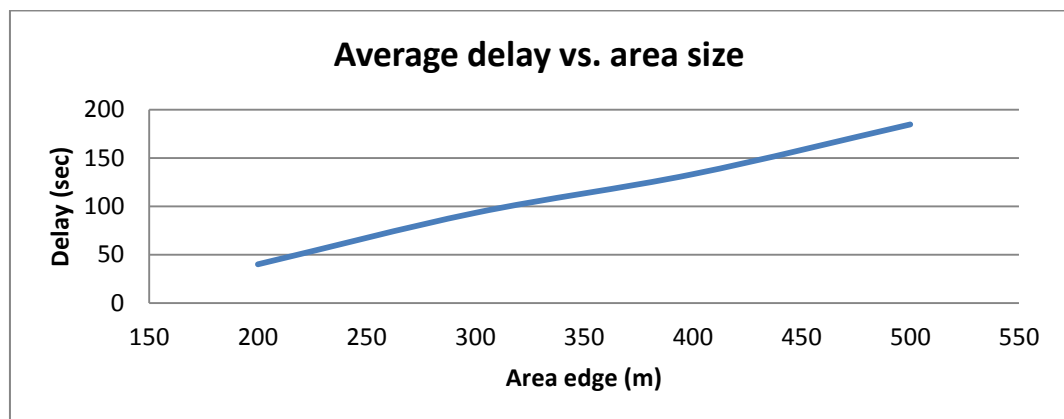


Figure 5.7: Average delay vs. area size

In order to obtain Figure 5.7, we have been increasing both the area edge and the number

of users at the same time, so that the population density remained exactly the same (if the area edge is doubled the number of users increases by a factor of 4). We may observe how the delay increases with the area size. We have also estimated the delivery probability in this case (Figure 5.8).

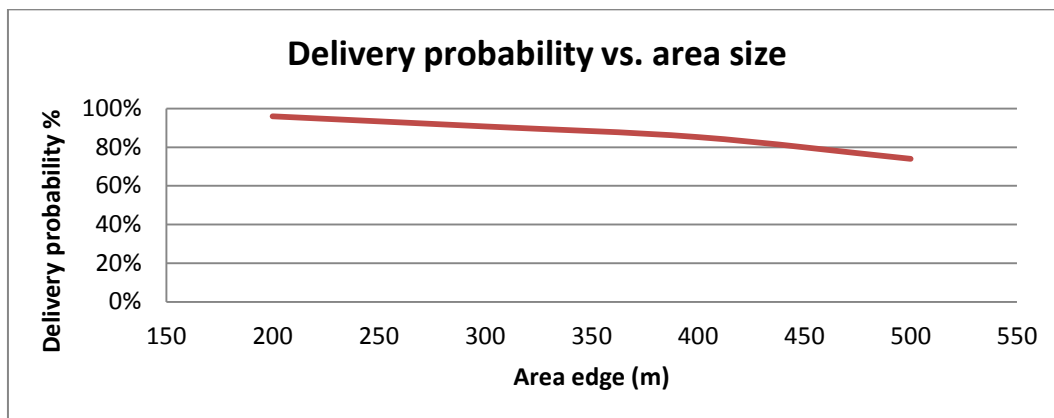


Figure 5.8: Delivery probability vs. area size

What is interesting from this figure is that for larger areas the delivery probability starts decreasing, even though the population density was kept at a fixed value. This can be explained if one considers the map pattern of the area. When a geographical area gets larger it is higher likely for parts of it to become isolated from central routes, and consequently more disconnected from the rest of the network. This would not be the case if there was no map pattern in the area and thus users were almost uniformly distributed inside it. Then we would expect exactly the same delivery probability for any area sizes (this conclusion has been validated by testing).

5.5 Survey 4: Nodes' mobility effect

We have already discussed how the mobility of the users affects the total energy consumed. We might also expect that it has an impact on the average end-to-end delay of the packets as well as on the delivery ratio. We have conducted new experiments in order to investigate these dependencies, where we are changing the mobility related parameters of intermediate nodes from "lower" to "maximum" mobility (as in Table 5.2). "Maximum" mobility refers to almost non-sleeping intermediate carriers, as a way to find the threshold on delay and

delivery probability in this scenario.

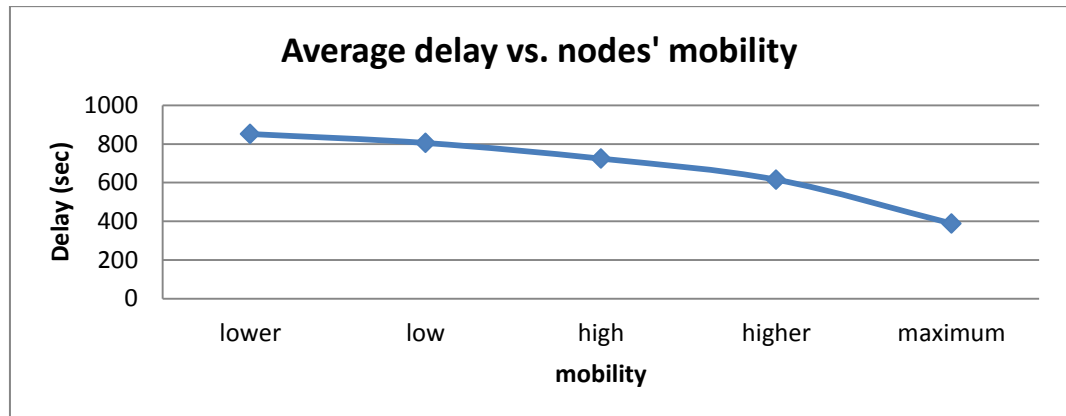


Figure 5.9: Average delay vs. nodes' mobility

These results show that larger mobility, interpreted as higher speed and/or smaller sleep time, leads to smaller packet delays and higher delivery ratios, since the encounters among nodes will be more frequent. One may also notice the "inversely proportional" relationship between the packet delay and delivery ratio, meaning that when the delay increases the delivery ratio is going to decrease and vice versa. This relationship may actually be observed in all conducted experiments.

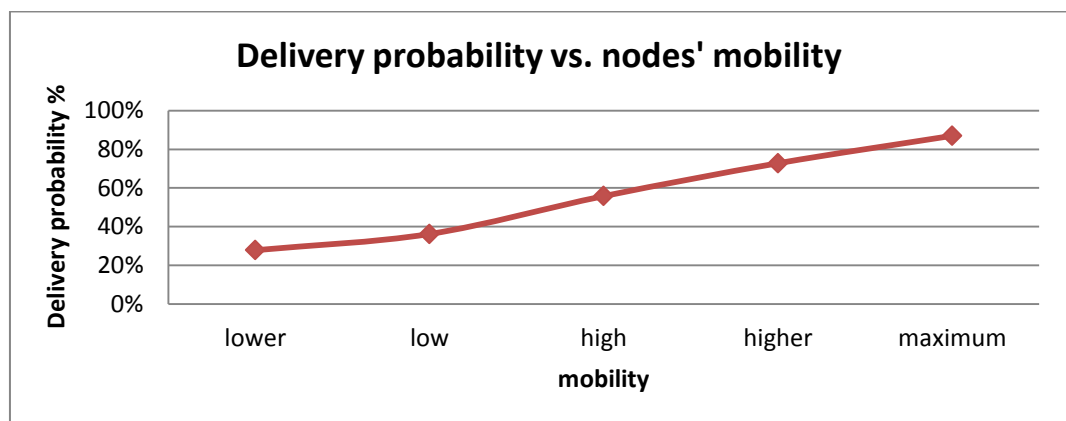


Figure 5.10: Delivery probability vs. nodes' mobility

5.6 Survey 5: Predictability of the carriers' movement

In order to conduct this survey we let all the nodes follow the random movement pattern as analysed in Chapter 3. In the proposed model, a packet is forwarded to another node if this is travelling in a direction higher likely to get the packet to its destination compared

to the current packet holder. Nevertheless, the new node is following a random movement pattern, meaning that any time it might change its direction and start heading towards a location different from the one that was used when the routing decision was made. Thus, it is made clear, that the predictability of a node's movement in the future is important for the performance of the network.

We expect that the more predictable the movement of a node, the more powerful the routing decision is, and thus the higher the probability that the carried packet will manage to reach its destination.

The predictability of the nodes' movement can be configured through the minimum and maximum "distance to travel" values. A node cannot change its direction until this distance has been covered, so by increasing these values we make its movement more predictable. Running the required tests, we have managed to confirm the aforementioned expected conclusions, as shown in the next two figures (the symbol " d " in the x axis symbolises the edge size).

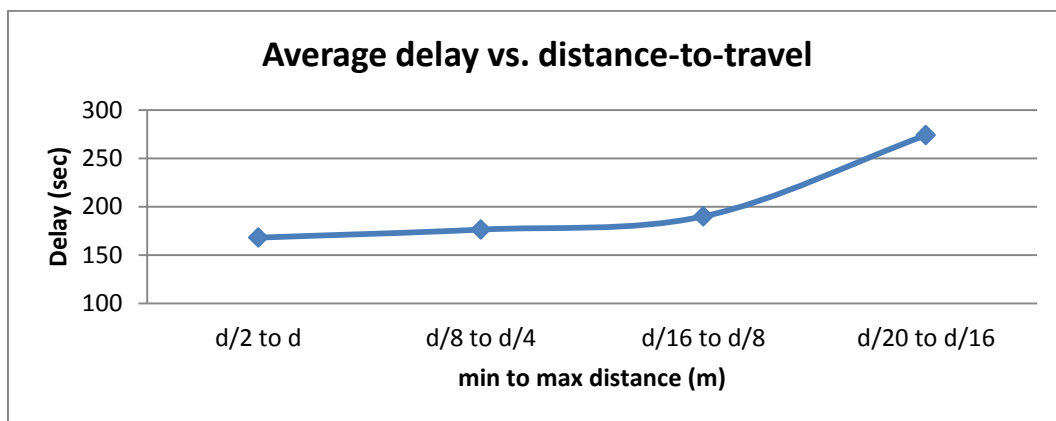


Figure 5.11: Average delay vs. distance-to-travel

We confirm that if the distance-to-travel is larger (e.g. $d/2$ to d), meaning that the node's movement is more predictable, the delay is less and the delivery probability is larger. Even though this survey has been performed using a random movement pattern, its conclusions may be generalised to realistic environments too.

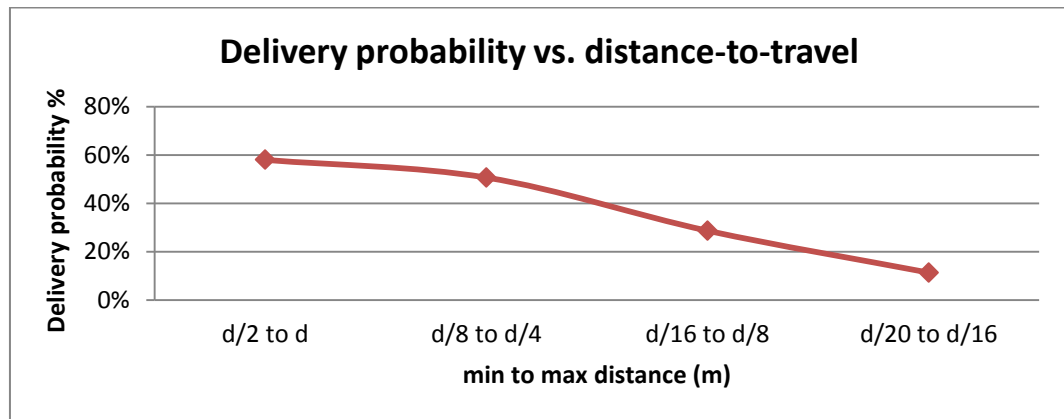


Figure 5.12: Delivery probability vs. distance-to-travel

5.7 Survey 6: Overhead in the network

We are going to study the incurred overhead in the proposed network model, defined as the unavoidable extra burden that the network has to carry and which does not lead to direct payload delivery. These inevitable extra packets refer to acknowledgements and agents, which do not carry information that the source actually intended to send. Moreover, they refer to payload packets that never reached their destination. Thus, the overhead in this model can be defined as:

$$\text{overhead} = \frac{\text{nr. of packets sent} + \text{nr. of ACKs sent} - \text{nr. of payload delivered}}{\text{nr. of payload delivered}}$$

Then, we get the next figure:

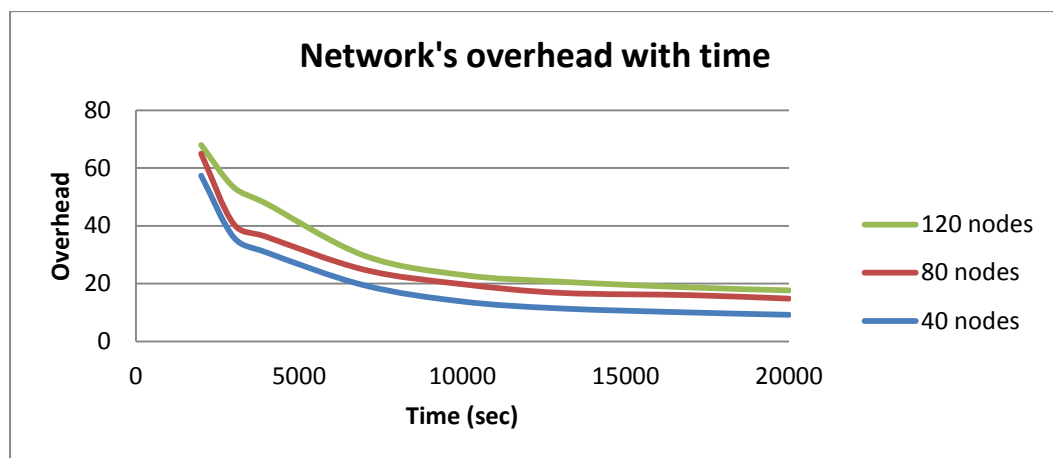


Figure 5.13: Network's overhead with time

This figure shows how the overhead in the network changes over time. As time progresses the overhead is significantly reduced because new routes are discovered and thus fewer explorers are sent to the network. Nevertheless, the existence of periodic multicasts and acknowledgements will never let the overhead fall below a certain value.

In the same figure we have displayed how overhead is affected by the number of nodes. It is a rational conclusion that an increase in the number of users in the network will slightly increase the overhead, not dramatically though, since there will be more payload packets delivered to their destinations.

5.8 Survey 7: Average number of hops

For the purposes of this survey we are going to measure the average number of hop counts experienced by payload packets travelling between the source and the destination node as a function of the simulation time.

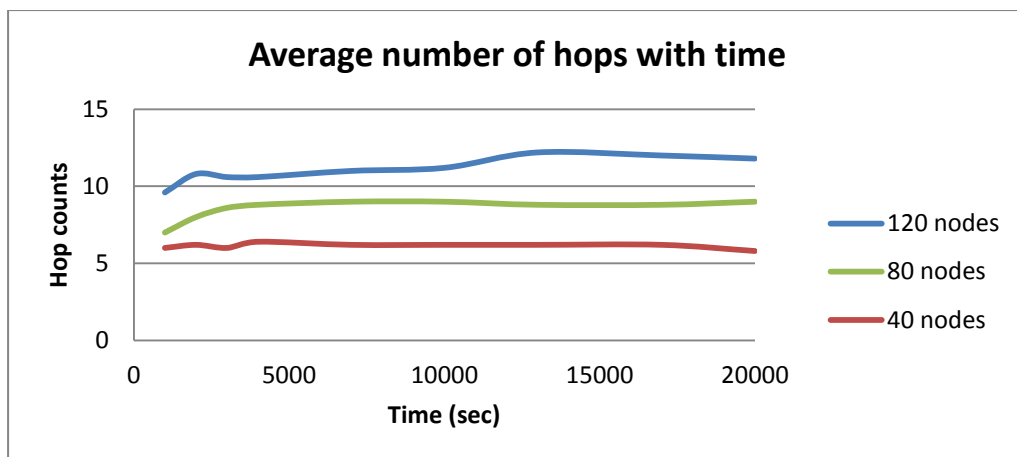


Figure 5.14: Average number of hops with time

We may draw two conclusions from this figure. First of all, the number of hops has an almost stable value throughout the whole time with the only exception being the start of the simulation, when the nodes are learning routes for the very first time. These almost stable hop count values can be explained by the fact that all new routes discovered are more or less similar, so that packets' routing behaviour does not vary significantly with time. Second, we observe that as the number of nodes increases, a packet will be exchanged more

often due to the existence of more advantageous routing opportunities. Consequently, the packet will probably experience more hops from one carrier node to another.

5.9 Survey 8: Effect of common routes and common sleep locations

In the current simulation we have a predefined number of central routes and a dynamic number of non-central routes, determined by the number of houses, offices and places of activities. The objective of this study is to examine how the available number of common routes and visited places affects the performance of the network.

In order to increase the number of common routes while keeping the number of users static (since we do not want the population density to affect the results), we could group more people together in houses, offices etc. In this way, fewer buildings will be created in the map in order to accommodate all users, and consequently fewer routes will be created that users can follow. Moreover, we may increase or decrease the amount of central junctions. Tests show that by decreasing the number of junctions (i.e. of central routes) the delivery probability is significantly reduced, which is a logical conclusion, since central routes are crucial for connecting parts of the map.

However, next we are going to focus only on how the number of common routes and common physical locations ("sleep locations") affect the network performance. The question raised is whether it is more efficient for each user to be assigned a unique route or sleep location or whether it is better when more users travel on common routes or share sleep locations.

The results show that the latter is preferable at least up to a point, and this can be interpreted as follows: First of all, users sharing sleep locations have more chances to quickly find someone that starts travelling on the desired direction, whereas if users are alone in their sleep location, they have to wait for a suitable passenger to hand over a packet. This result validates in a way the intuitive conclusion that the existence of social

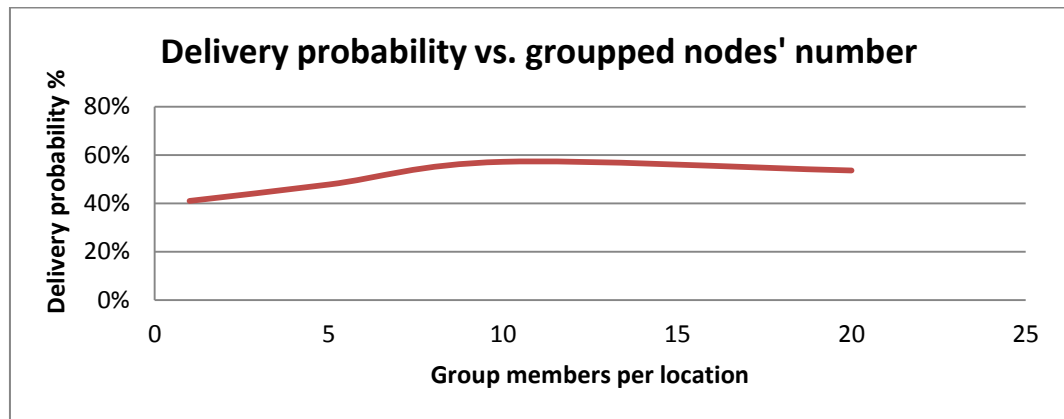


Figure 5.15: Delivery probability vs. grouped nodes

communities increases the chances of successful packet delivery.

Nevertheless, if there are fewer routes available, there will be less possible encounters while nodes are moving. As a result when more and more nodes are grouped together the delivery ratio does not keep constantly increasing; on the contrary it starts decreasing. We may draw the conclusion that based on the specific area under study, there will be an ideal value for the number of nodes grouped together that will lead to better network performance.

5.10 Survey 9: Packet chains

The concept of packet chains has been introduced in a previous chapter. We have performed several simulation tests in order to investigate the contribution of creating chains in the current network model and to gain some insight on how these chains would affect the network efficiency if used in slightly different models.

Surprisingly enough, simulation showed that chaining packets together using the algorithm described in section 3.5.4 did not make a significant contribution as far as delivery ratio and delay are concerned. Although an explanation for this behaviour is not so obvious, this may be interpreted if one takes the following facts into consideration:

- Packets are only chained together if they are travelling towards the same destination.

Thus, the amount of those packets accidentally carried by the same node at the same point in time together with the size of the carrier node's buffer highly determine whether packet chains will be created.

- Once a source has discovered a good path, it is going to recommend it to all new packets. In that sense, packets will be guided towards that path in the future, even if it is not the ideal one. Since all new payload packets are going to be guided in this way, the possibility that new routes will be discovered through payload packets is limited. (Agents produced by periodic multicasts overcome this issue). This means that packets deriving from the same source within some time period will rarely carry totally different recommended paths, since they will all represent the source's current perception of the network.
- We could generalise this by considering that, at a specific time instant, the routes carried by packets in the network are not so irrelevant with each other. Besides, the fact that two packets heading to the same destination were at the same place at the same time is not so coincidental.
- In general, it may be a good idea in this model to maintain some diversity among the routes instead of chaining them together, keeping of course in parallel the notion of direction. In this way, better routes may be discovered.
- When updating a packet's path-to-follow with a better one via chaining, this does not mean that the packet would not have been delivered to its destination after all. So we do not expect the delivery probability to be seriously affected.

This current behaviour by no means implies that the concept of packet chains may not contribute to network efficiency in different environments. On the contrary, one may imagine a scenario, where e.g. packet A contains the most up-to-date route to the destination, packet B contains the fastest route, packet C contains a longer route and packet D does not contain any route at all. In this case, it is quite obvious that updating the paths with packet A's or packet B's path (or a combination) will yield much better results. Unfortunately, at the moment, the current network model is not suitable for obtaining this kind of

results, since such opportunities are existent but not sufficient for improving the network's performance.

Packet chains in the current network model may significantly contribute though in order to decrease the processing time and corresponding energy consumption by nodes carrying those chained packets. More specifically, if a group of packets is tagged as 'chained', decisions could be taken collectively and not for each packet separately, thus avoiding redundant processing operations.

5.11 Survey 10: Route history list

We are going to describe and graphically display a simplified scenario that explains the functionality of the "Route History List", as this was described in Chapter 3. In this way, we will validate that the Route History List created and updated at the source side actually plays a significant role in the routing of packets. The description of this scenario together with the relevant screenshots is available in Appendix A.2.

Chapter 6

Conclusions and Future Work

THE simulation results of the previous section prove a rational behaviour of the modelled network, since this responds expectedly to variable input parameters. Furthermore, it is evident that this model may offer high quality routing in Opportunistic Networks, subject to the environment's parameters like area, population size and travelling speed. Performance metrics, such as delivery probability and end-to-end delay, show that this model could be indeed adopted in Delay Tolerant Networks. In fact, it may offer up to 100% delivery probability in the case that the destination is static and some lower values depending on the direction and speed in which this is moving.

So far, the area between the source and the destination has been considered empty-space and users travel on straight lines between their current and next positions. However, the same model could be used directly into a more realistic environment too, where e.g. nodes come across physical obstacles when moving around (so that they have to temporarily change their direction) or where they travel on well-defined map patterns. In spite of such added properties, the proposed routing algorithm should also work in such an environment, with slight or no adaptations at all. It should be able to discover routes to a destination and use these ones with a high probability of success, taking into account a) that the destination will be moving using a specific pattern and not randomly and b) that the destination cannot move very far from its current position into a certain amount of time.

Using this new routing protocol, the source nodes may be able to discover routes to send packets to their destinations, even if those routes are complicated and consist of numerous approximate intermediate locations. In this sense, one could think of the proposed routing algorithm as a *tracking* or *route discovery* algorithm, since the source node may be able to know/assume the approximate position of another node prior to routing any new packets to it. This information is based only on the previous experiences of successfully delivered and acknowledged packets.

The proposed routing model not only offers the capability to discover routes based on geographical criteria but also on population congestion criteria. As an example, if a specific area is very central during a certain time of day and many nodes are moving there towards various directions, this area will be an intermediate hop high likely to be recommended many times by packet generators. However, if this same location during a different part of the day is not central any more, it will gradually cease to be recommended. In other words, the algorithm is capable to identify where and when the traffic is concentrated and to adjust its routing decisions accordingly.

Furthermore, it would be interesting to point out that when sources recommend paths to be followed by the packets, they provide "GPS-like" information. This means that they let the carrier of the packet know how to "drive" the packet to its destination as well as how much the expected time to get there is (like a GPS device). Of course, there is no guarantee that the packet will be actually delivered to the destination, but this recommendation still significantly removes the uncertainty of such environments.

The routing protocol described may be considered as one paradigm of Self Aware Networks (SAN), and this is justified as follows. Routing information in the proposed network model becomes available *on demand*, that is only when required and only by those who will directly find it useful. Users may join or leave the network autonomously but they will discover paths only when the need to communicate arises. The challenges of SANs are present here as well. For instance, routing information may become obsolete even before this is propagated back to the source nodes that requested it. Additionally, the

phenomenon of *routing oscillations* similar to the one observed in SANs may occur, thus affecting the performance of the network. A central (popular) path may become seriously loaded and thus unable to carry new packets. This will mean that payload packets will be discarded, so ACKs will never go back. Consequently, the source nodes will need to discover new paths, subject to experience the same phenomenon [6].

Now, let's make an effort to interpret the delivery success or failure of a transmitted packet. So let's assume that according to the most recent routing history information available the source (S) assumes that the destination (D) is located at position (x_1, y_1) . Whether the packet will be delivered or not depends on how reliable or obsolete this information will be after time $t_1 + t_2 + t_3$, where:

- t_1 is the time it takes for the ACK to return to the source
- t_2 is the time interval between an arriving ACK and the generation of a new payload packet towards the same destination
- t_3 is the time it will take for the new payload packet to reach its destination.

Based on this description, the packet will not be delivered to its destination in the following two cases:

- During that time period, the destination has moved significantly from its previously recorded position. "Significantly" means that it has shifted from its initial position (x_1, y_1) to a position (x_2, y_2) , located at a distance larger than the communication range. So, even if a carrier node successfully reaches this originally recorded position (x_1, y_1) it will not be able to locate the destination and hand over the packet.
- Even if the packet does not find its destination at its final target position, it may have the chance to locate it once it continues moving and until the moment the packet's TTL expires. It should be clarified at this point that when packets have exhausted their recommended paths-to-follow without success, they are not "self-destroyed" but nodes keep carrying those packets hoping to meet their destination in the near future. This is actually quite likely to happen, since the destination may not have moved very

far from the previously recorded position, and if the carrier's and the destination's directions are not completely different, these may still have high chances to meet each other.

Based on these remarks, we conclude that an *ideal* (but not required) environment to use the proposed protocol would be one of the following:

1. A relatively small area, where source and destination are not located very far from each other. In this way, any previously acquired knowledge concerning routing will not immediately become obsolete.
2. An environment where source and destination nodes are not constantly moving, so that their behaviour is not totally unpredictable or unstable.
3. An area where parts of it are connected using very fast routes, so that the current knowledge of a source concerning a destination's location may lead to successive successful packet delivery, even if the destination is not highly static.

Below we present some proposals for future work, which however exceeded the current purposes of this specific study:

1. Related to the discussion preceded, it might be a good technique to perform some kind of clever multicast when the packet has exhausted the recommended path without finding its destination. In this way we would significantly reduce the possibilities that a packet will not be delivered to its destination after having already completed a correct, long route in order to get there and the destination is still in the surroundings of this area, i.e. not so far away (which is a very realistic case).

2. One more interesting idea for future work would be the prediction of the destination's future position by the source. For instance, if at time t_1 the source receives an ACK reporting that the destination is at position (x_1, y_1) and then at $t_1 + dt$ it receives a second ACK that the destination now is at $(x_1 + dx, y_1 + dy)$, then the source could predict where the destination might be at a future point in time. Such a prediction could

be possible based on an estimation of the destination's direction of movement and of its speed and assuming that these haven't changed yet. In addition to that, the source already has previous knowledge on how long it takes for a new packet to get to the destination (delay information), so that combining all this information (direction, speed, delay) it may create a recommended path that is not a "blind copy" of a previously reported route, but is based on an prediction of the destination's new position. The more the subsequent ACKs this source receives for the same destination, the more reliable this prediction will be.

3. Another idea for future work that may be worth simulating is supplying packets with more than one recommended routes so that if one fails, the packet could make an attempt to use an alternative one. Moreover, it would be challenging if the routes inside the packets were automatically adjusted based on the knowledge of a carrier's current position. At the moment, a packet carrier "blindly" tries to use the recommended path, but if for instance a packet somehow (even accidentally) found itself closer to one of the future recommended intermediate hops (but not yet there) it may be more efficient to self-adjust the recommended path. (Note: this adjustment happens now only if the packet has already *reached* this future recommended intermediate hop).

4. Similar to that idea would be for the source to combine more than one routes together so as to create a single "optimum" route. For instance, if route 1 reported how to get faster from point A to point B while route 2 reported how to get faster from point B up to the destination, these two routes could be combined in order to provide an enhanced, better copy (similar to the principles of "diversity theory"). This issue has been actually investigated in [26], where *Genetic Algorithms (GA)* are used in order to enhance the Route Discovery process. The GA algorithm may run in all source nodes (in background mode to reduce overload) in order to combine existing paths in the Route History List and create new ones using the *path crossover operation*. The combined paths will have the same Source-Destination and share at least one intermediate node. In this way, the GA will actually reveal new, valid paths, not yet discovered by the agents. These paths will be saved in the same Route History List and thus fairly compete with the rest of the paths

whenever a new packet is created and the path to recommend is searched. The end-to-end delay, which is the QoS metric used by source nodes at the moment, may be synthesized by adding the delays of each path component. This information is already available via ACKs, but is currently not exploited by source nodes in the model. Since smart decision making is taking place at source nodes and not in a distributed manner (as in pure CPN) we would expect that the GA will improve the network performance in terms of delay, provided that the path information mixed together will not usually involve obsolete path constituents.

5. There are some cases (as shown via the GUI), where lines connecting two successive intermediate hops of the same recommended path intersect each other. For instance, if a recommended path consists of the hops {A B C D E F G H}, it might sometimes happen that lines B-C and F-G intersect each other. In that case, the point where these lines meet could be used as a new intermediate hop, which would replace points C, D, E and F. This is not currently implemented to avoid complexity, but has been observed using the GUI. Such recommended routes are quickly replaced by better ones though, as also observed by the GUI.

6. At the moment, the ACK packets do not need to record their path while travelling towards the source, since this information is of no use. However, we may assume that after node A sends a message to node B, then it is high likely that node B will reply to node A and so on for some period of time. Consequently, it may be useful to enhance the model so that it offers two-way communication in a more clever way. For instance, an ACK message could record its path from node B to node A, and then A could report this information back to B through the new payload message. This would avoid unnecessary overhead in the network, since node B would otherwise have to use explorers in order to send packets to A. In the empty-space environment that has been implemented in this study, reversing the path used to get from A to B is sufficient to route a packet from B to A; yet, in real life this is not necessarily true.

7. The movement patterns implemented are realistic enough and making them

too sophisticated or complex exceeded the purposes of this study. Nevertheless, this is one more challenge for future work, since more realistic patterns would help evaluate the proposed routing model even better. Several ideas on how this could happen are analysed in [23] and the model described there has been already implemented and integrated with the ONE simulator. (The ONE simulator is also implemented in JAVA and the source code is available for downloading [27]).

8. A real map pattern may be used in order to evaluate the proposed protocol. Parts of the map could be considered smaller clusters where users are higher likely to move and clusters could be connected via central routes. By adjusting the size and number of these clusters, *locality* may increase or decrease, affecting in parallel the performance of the opportunistic network. However, such an approach also exceeded the objectives of the current study.

9. Although social relationships are present in this model (users gathering in the same house, office or evening activity may be family, colleagues and friends respectively), they are not taken advantage of when making routing decisions. It may prove efficient though, if users with such social relationships familiarised each other with their working-day patterns, so that a packet may be a priori relayed from one node to another based purely on this information.

10. The algorithm would be able to reveal the frequently-visited places of a destination node together with the corresponding times of day, meaning that the source may identify the working-day pattern of the destination. Consequently, if the packet generator used this information too, performance could increase even more.

11. At the moment, no retransmissions are initiated for lost packets. However, this model offers the knowledge of whether a packet has successfully reached its destination or not, through the existence or absence of ACK messages. Retransmission could automatically be initiated at the source's side if an ACK has not been received within a reasonable amount of time. Of course, the absence of an ACK does not necessarily imply that a packet has not reached its destination; it may have just happened that the

ACK failed to return back. Alternatively, carrier nodes could produce negative acknowledgements for payload packets that have just expired while attempting to reach their destination; however this approach would increase the network overhead and complexity, and their use may even prove to be redundant.

12. A packet holder is not modelled to combine the information of its carried agents, ACKs or payload packets. An idea for future study is to somehow combine this information in order to make better decisions. For instance, a carrier may decide to throw an agent packet that is still trying to discover the destination's location as long as it carries a recent ACK from this destination. Similarly, a carrier may update the recommended path of a payload packet if it carries a recent ACK that is going to report a better route back to the source.

13. Finally, one could easily model the switching on or off of devices in the network, as well as the arrival or departure of new nodes at/from the area under study.

The proposed model consists of three basic, stand-alone parts: a movement model, a routing model based on the notion of direction and a cognitive packet framework. The latter could even be adopted under totally different conditions, i.e. in a model with a different movement pattern or in a model where packets carry any other kind of routing information (not relevant to geographical data).

As a conclusion, the proposed CPN-type enhancement for Opportunistic Networks seems to make a genuine contribution to this challenging research field. This study aspires to lay the foundations for future work, which would potentially lead to the development of a generic, widespread routing algorithm for delay tolerant networks.

One may find the Java code written to implement this model in the accompanied CD. The code was written with the objective to be easily comprehended and expanded by anyone interested in this study. An electronic copy of this document is also included in the CD.

Bibliography

- [1] G. Ansa, H. Cruickshank, and Z. Sun, “An energy-efficient technique to combat dos attacks in delay tolerant networks,” *ICST Transactions on Ubiquitous Environments*, vol. 12, no. 1-3, 3 2012.
- [2] A. Lindgren, A. Doria, and O. Schelén, “Probabilistic routing in intermittently connected networks,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, Jul. 2003. [Online]. Available: <http://doi.acm.org/10.1145/961268.961272>
- [3] E. Gelenbe, Z. Xu, and E. Seref, “Cognitive packet networks,” in *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, ser. ICTAI '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 47–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=850950.853686>
- [4] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, “Towards networks with cognitive packets,” in *Proceedings of the International Conference on Performance and QoS of Next Generation Networking*, Nagoya, Japan, November 2002, pp. 3–17, opening Invited Paper.
- [5] E. Gelenbe, R. Lent, and Z. Xu, “Measurement and performance of a cognitive packet network,” *Comput. Netw.*, vol. 37, no. 6, pp. 691–701, Dec. 2001. [Online]. Available: [http://dx.doi.org/10.1016/S1389-1286\(01\)00253-5](http://dx.doi.org/10.1016/S1389-1286(01)00253-5)
- [6] E. Gelenbe, “Steps toward self-aware networks,” *Commun. ACM*, vol. 52, no. 7, pp. 66–75, Jul. 2009. [Online]. Available: <http://doi.acm.org/10.1145/1538788.1538809>
- [7] E. Gelenbe and R. Lent, “Mobile ad-hoc cognitive packet networks,” in *Proceedings of the IEEE ASWN Conference*, Paris, FR, July 2002.

- [8] R. Lent and F. Zonoozi, "Power control in adhoc cognitive packet networks," in *Proceedings of the 2005 Texas Wireless Symposium*, October 2005.
- [9] E. Gelenbe and R. Lent, "Power-aware ad hoc cognitive packet networks," *Ad Hoc Networks Journal*, vol. 2, no. 3, pp. 205–216, July 2004. [Online]. Available: http://www.sciencedirect.com/science?_ob=ArticleURL&.udi=B7576-4C8897Y-3&.coverDate=07%2F31%2F2004&.alid=499740262&.rdoc=1&.fmt=&.orig=search&.qd=1&.cdi=12890&.sort=d&.view=c&.acct=C000011279&.version=1&.urlVersion=0&.userid=217827&.md5=82edee8594df1809224c0ab1b1b7f27b
- [10] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "Cognitive packet networks: QoS and performance," in *Proceedings of the IEEE MASCOTS Conference*, Ft. Worth, TX, October 2002, pp. 3–12, opening Keynote Paper.
- [11] E. Gelenbe and P. Liu, "QoS and routing in the cognitive packet network," in *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2005, pp. 517–521.
- [12] G. Sakellari, "The Cognitive Packet Network: A Survey," *The Computer Journal: Special Issue on Random Neural Networks*, vol. 53, no. 3, pp. 268–279, June 2009, doi: 10.1093/comjnl/bxp053. [Online]. Available: <http://comjnl.oxfordjournals.org/cgi/content/abstract/bxp053>
- [13] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges," *Communications Surveys Tutorials, IEEE*, vol. 8, no. 1, pp. 24–37, quarter 2006.
- [14] L. Lilien, Z. H. Kamal, V. Bhuse, A. Gupta, and W. (wireless Sensornet Lab, "Opportunistic networks: The concept and research," in *Challenges in Privacy and Security, Proc. NSF Intl. Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN 2006)*, 2006.

- [15] Y. Li, Y. Jiang, D. Jin, L. Su, L. Zeng, and D. Wu, “Energy-efficient optimal opportunistic forwarding for delay-tolerant networks,” *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 9, pp. 4500–4512, nov. 2010.
- [16] Pelusi, “Opportunistic networking: data forwarding in disconnected mobile ad hoc networks,” *Communications Magazine, IEEE*, vol. 44, no. 11, pp. 134–141, november 2006.
- [17] P. Hui and J. Crowcroft, “Human mobility models and opportunistic communications system design,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1872, pp. 2005–2016, 2008. [Online]. Available: <http://rsta.royalsocietypublishing.org/content/366/1872/2005.abstract>
- [18] K. Xu, P. Hui, V. O. K.Li, J. Crowcroft, V. Latora, and P. Lio, “Impact of altruism on opportunistic communications,” in *Proceedings of the first international conference on Ubiquitous and future networks*, ser. ICUFN’09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 153–158. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1671729.1671759>
- [19] H. Cai and D. Y. Eun, “Crossing over the bounded domain: from exponential to power-law inter-meeting time in manet,” in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, ser. MobiCom ’07. New York, NY, USA: ACM, 2007, pp. 159–170. [Online]. Available: <http://doi.acm.org/10.1145/1287853.1287873>
- [20] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović, “Power law and exponential decay of inter contact times between mobile devices,” in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, ser. MobiCom ’07. New York, NY, USA: ACM, 2007, pp. 183–194. [Online]. Available: <http://doi.acm.org/10.1145/1287853.1287875>
- [21] A. Keränen, J. Ott, and T. Kärkkäinen, “The one simulator for dtn protocol evaluation,” in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, ser. Simutools ’09. ICST, Brussels,

- Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 55:1–55:10. [Online]. Available: <http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5674>
- [22] F. Bai and A. Helmy, “Chapter 1 a survey of mobility models in wireless adhoc networks.”
- [23] F. Ekman, A. Keränen, J. Karvo, and J. Ott, “Working day movement model,” in *Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models*, ser. MobilityModels '08. New York, NY, USA: ACM, 2008, pp. 33–40. [Online]. Available: <http://doi.acm.org/10.1145/1374688.1374695>
- [24] A. King, *Java game programming*, (accessed September, 2012), http://www.gamedev.net/page/resources/_/technical/game-programming/java-game-programming-part-i-the-basics-r1262.
- [25] J. Bodnar, *Java 2D games tutorial*, (accessed September, 2012), <http://zetcode.com/tutorials/javagamestutorial/movingsprites/>.
- [26] E. Gelenbe, P. Liu, and J. Laine, “Genetic algorithms for route discovery,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 6, pp. 1247–1254, dec. 2006.
- [27] J. O. e. a. Ari Kernen, *The Opportunistic Network Environment simulator.*, (accessed September, 2012), <http://www.netlab.tkk.fi/tutkimus/dtn/theone/>.

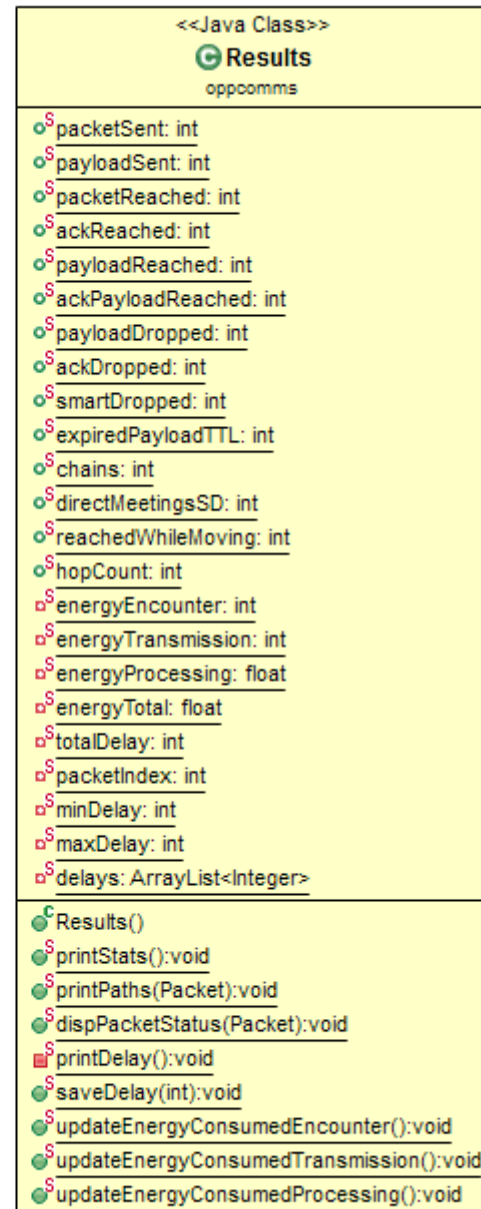
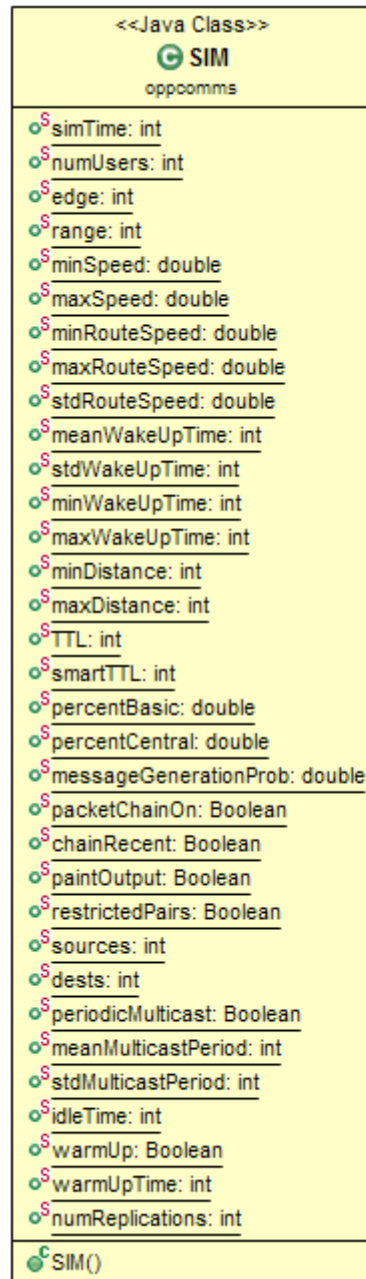
Appendix A

A.1 JAVA classes

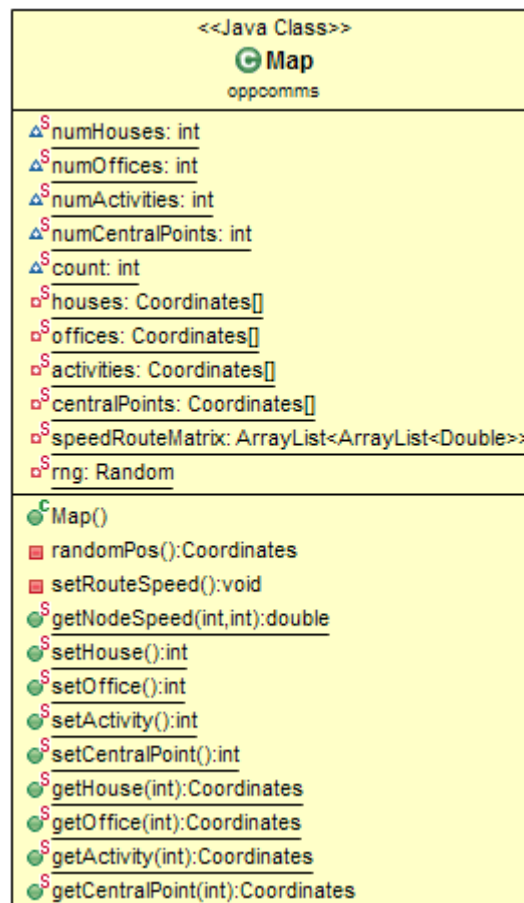
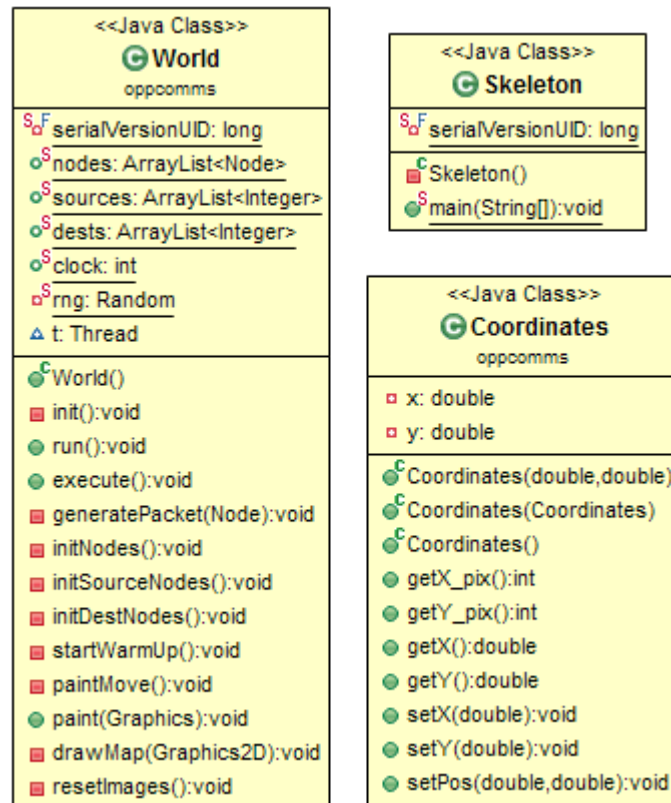
A.2 Scenario description of 5.11 (Route history list)

A.1: JAVA classes

I. SIM – Results

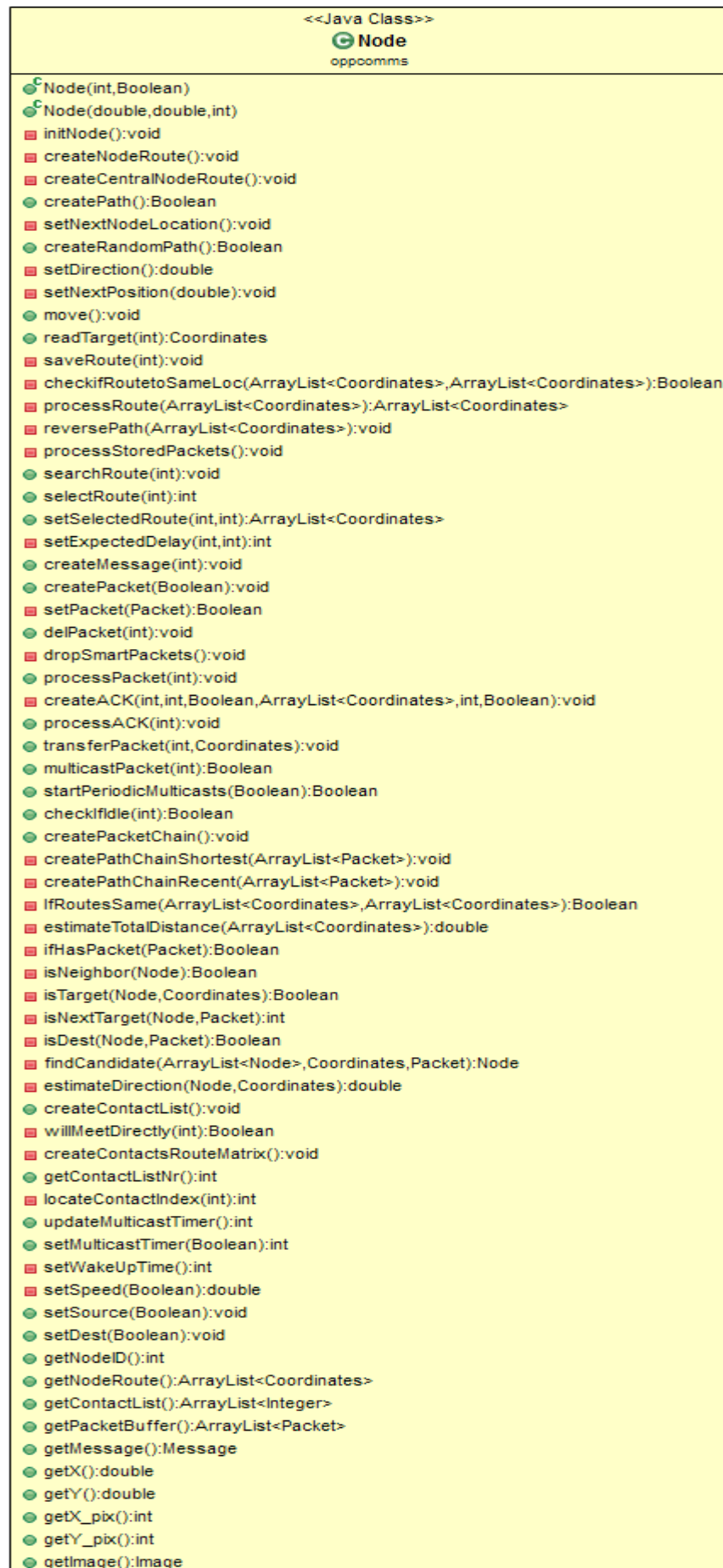


II. Skeleton – World – Coordinates – Map

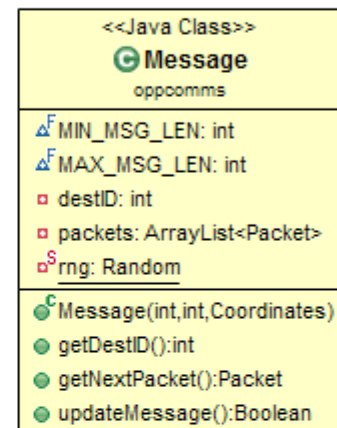
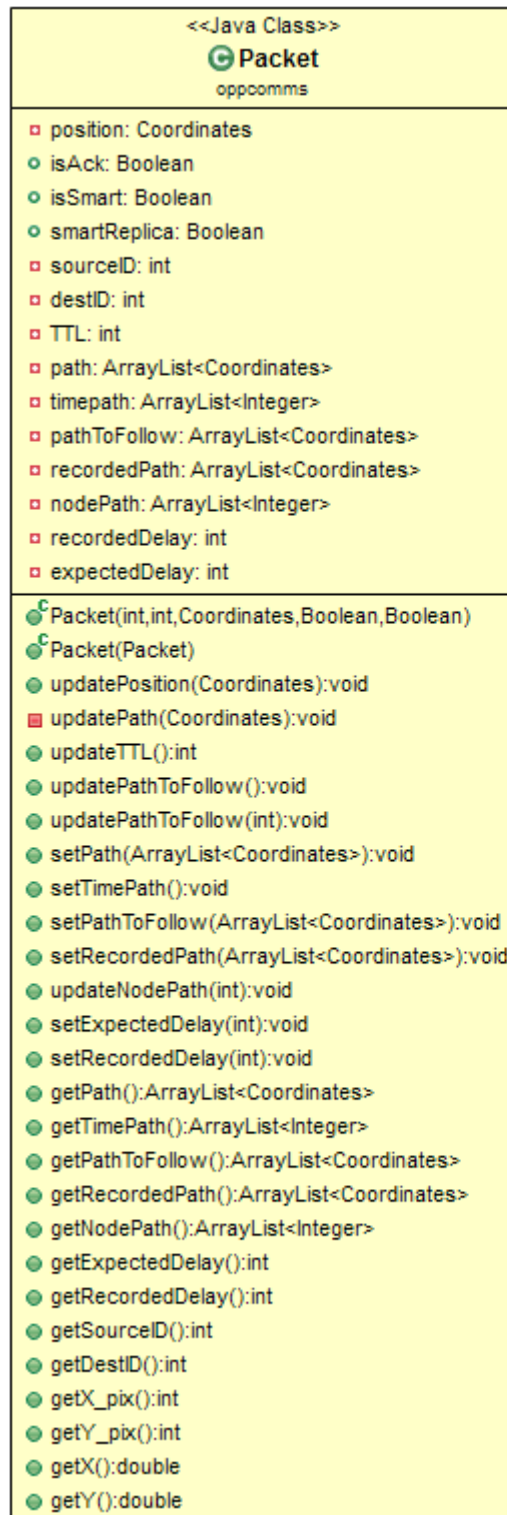


III. Node





IV. Packet – Message



A.2: Scenario description of 5.11 (Route history list)

We are going to describe a simplified scenario with one source-destination pair, which can be easily generalized for any number of source-destination pairs.

1. When the simulation begins, the source node creates a new payload packet. Since no recommended path can be found, agents will be sent, as shown below.

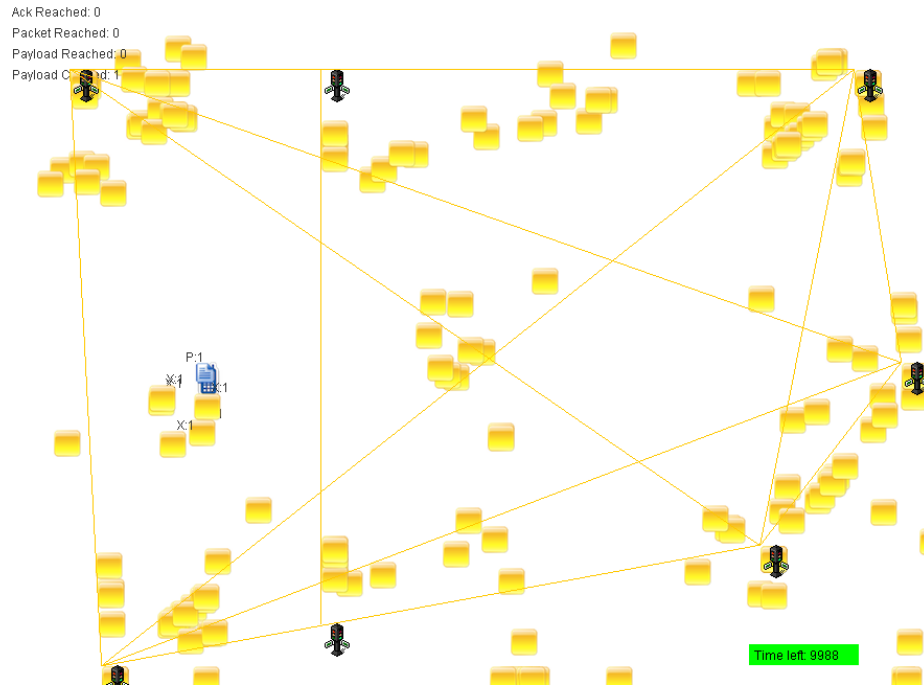


Figure A.1: Scenario description - Stage 1

2. When agents arrive at the destination, ACK messages are created which are going to return back to the source node following the reverse route. For simplicity, we do not display the agents from now on in this graphical output.

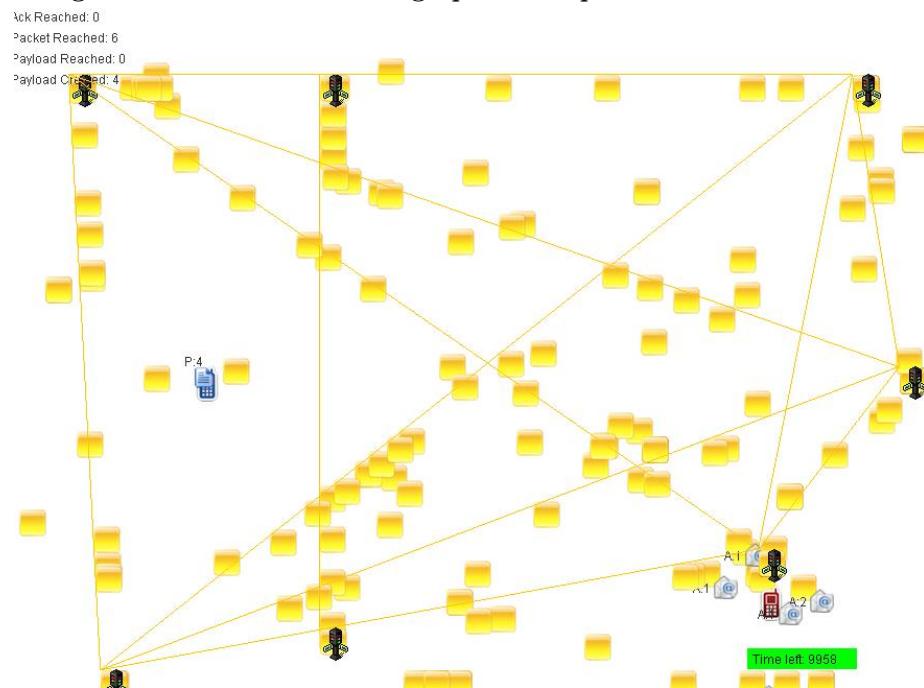


Figure A.2: Scenario description - Stage 2

3. When the first ACK reaches the source, the path that this ACK carries is going to be saved in the Route History List. This path contains all the positions where packet forwarding took place, i.e. where the packet changed direction. For as long as no new ACKs have arrived, this will be the only recommended path (usually not very good at first).

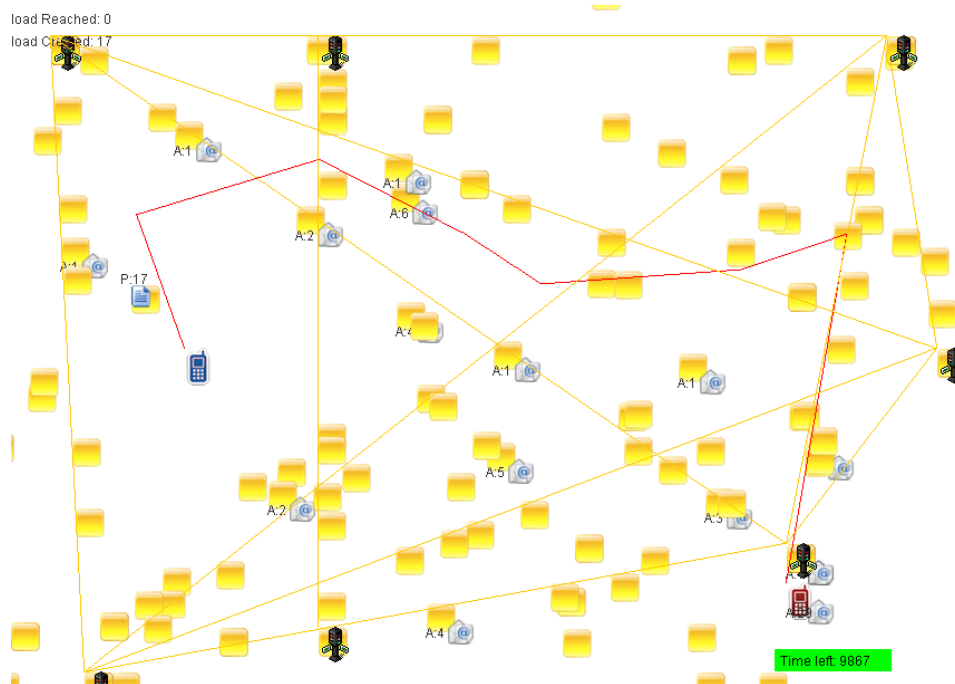


Figure A.3: Scenario description – Stage 3

4. The recommended path is constantly updated with better options throughout the simulation. As already explained, the recommended path towards a destination is based on the delay reported by ACKs. The delay is a function of the distance travelled and of the speed of nodes in the routes used. The optimal path is high likely to be like the following one, which is guiding packets through two intersections (triangles in Figure A.4) approximately, i.e. where there is higher traffic.

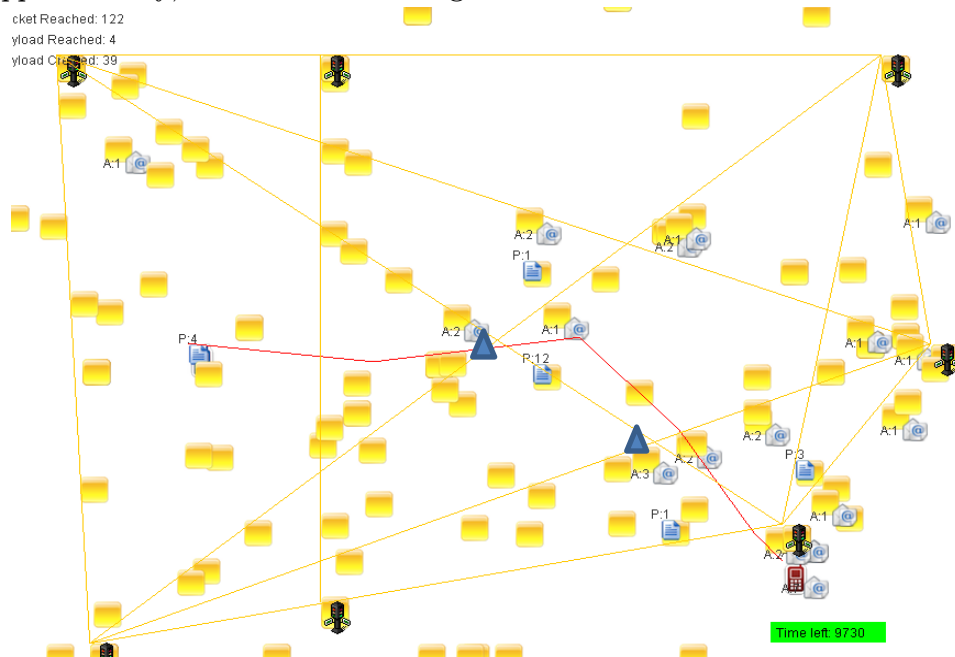


Figure A.4: Scenario description – Stage 4

5. When the source changes position, this recommended path is going to be successfully updated with a new one.

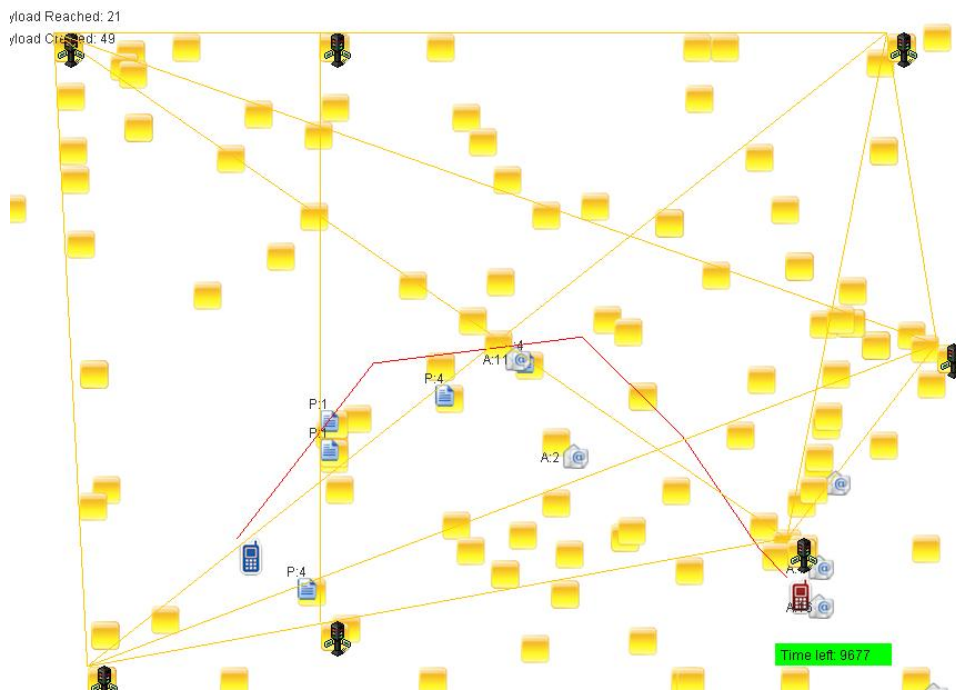


Figure A.5: Scenario description – Stage 5

As we can see, packets and ACKs are “surrounding” this recommended path, since they have the relative information stored inside them.

6. This path might be updated with an even faster one:

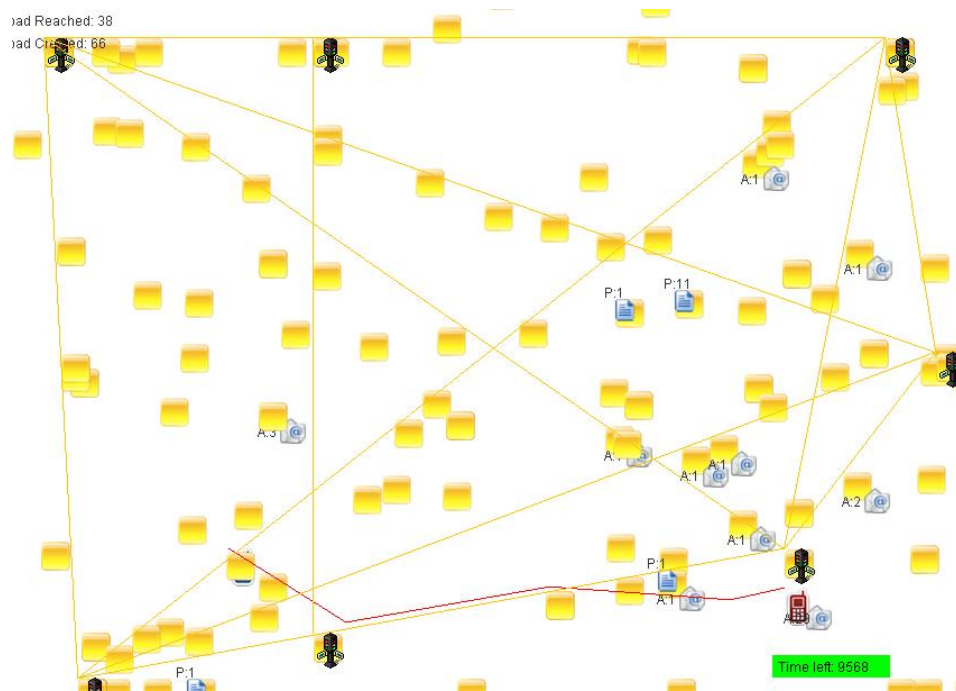


Figure A.6: Scenario description – Stage 6

7. Now, if the destination changes position, the source is still able to locate the route to this new location and find a good path.

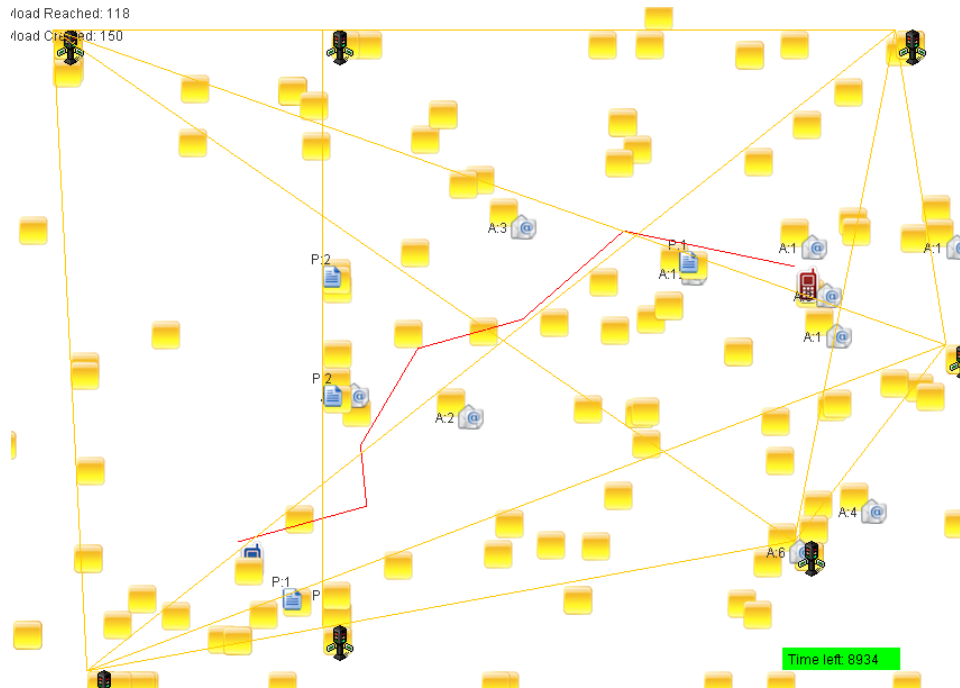


Figure A.7: Scenario description – Stage 7

When this screenshot was taken, packets and ACKs were already familiar with this path and thus they are concentrated around it in majority.

Finally, as already mentioned, this process is performed per source for all its contact list entries (potential packet destinations). (Below: two source nodes with the same destination node).

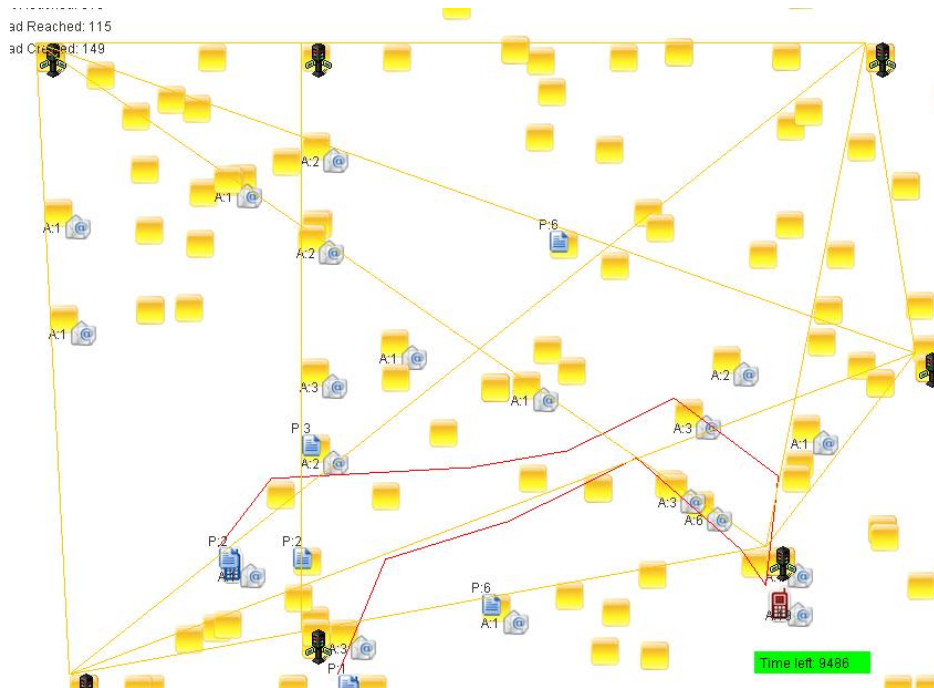


Figure A.8: Scenario description – Stage 8