Ατομική Διπλωματική Εργασία

IMPLEMENTATION AND EVALUATION OF THE BIOLOGICALLY – INSPIRED ANTHOCNET ROUTING PROTOCOL IN SENSOR NETWORK

ΣΑΝΤΥ ΜΙΧΑΗΛ ΑΓΙΑΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2011

Ατομική Διπλωματική Εργασία

IMPLEMENTATION AND EVALUATION OF THE BIOLOGICALLY – INSPIRED ANTHOCNET ROUTING PROTOCOL IN SENSOR NETWORK

ΣΑΝΤΥ ΜΙΧΑΗΛ ΑΓΙΑΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

IMPLEMENTATION AND EVALUATION OF THE BIOLOGICALLY – INSPIRED ANTHOCNET ROUTING PROTOCOL IN SENSOR NETWORK

ΣΑΝΤΥ ΜΙΧΑΗΛ ΑΓΙΑΣ

Επιβλέπων Καθηγητής
Δρ. Ανδρέας Πιτσιλλίδης

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2011

Acknowledgments

I would like to express my gratitude to my advisor Prof. Andreas Pitsillides for his continuous guidance, support, and patience during this research. I was really lucky to have him as my advisor. In addition, I would like to thank Dr. Pavlos Antoniou, for his valuable help and knowledge, insightful comments and being always present to answer all my questions.

My sincere thanks also goes to Mr. Samer Arandi for his continuous support and encouragement throughout this year.

Last but not least, I would like to thank my family: my parents Mohammad Ayas and Eleni Michael - Ayas, whom no matter what I do till the end of my life, I will never do them just. My dearest brothers: Malek and Hamdy and my grandparents: Vasiliki, Kiriakos, Khadija, Ahmad. Thank you all for the unconditional love and support.

Abstract

This work addresses with the problem of routing and congestion in wireless sensor networks (WSNs). In particular, it implements and evaluates a robust and self-adapting nature-inspired congestion control protocol for real-time event-based applications. We study the problem of congestion in WSNs from the perspective of nature-inspired design. The advantage of this approach is that it simplifies the implementation of every node, and requires minimal information exchange, while maximizing network lifetime and providing graceful performance degradation. Our main source of inspiration is the field of Ant Colony Optimization (ACO). The adaptivity and robustness of ACO algorithms allow handling the challenges presented in the environment. We present the AntHocNet routing algorithm, which combines ideas from ACO routing with techniques from dynamic programming and other mechanisms taken from more traditional routing algorithms. The main idea of the implementation of the AntHocNet is to ‘guide’ packets (ants) to flow towards the sink (global attractor), whilst trying to avoid congestion regions (obstacles). AntHocNet contains elements from both reactive and proactive routing. Specifically, it combines a reactive route setup process with a proactive route maintenance and improvement process.

We present a detailed description of our implementation of this algorithm within the NS-2 network simulator. We perform a thorough evaluation of our implementation using a number of scenarios under various network congestion conditions. We perform a guided exploration of the control parameter space to validate our implementation and reach an optimal combination for the tested scenarios. We also perform a detail examination of the different internal stages of the protocol for a better understanding of its inner working. And finally, we compare the performance of the protocol with the FlockCC protocol and show that for the scenarios and control parameters we used the AntHocNet achieves better performance.

Contents

Chapter 1

Introduction

1.1 General Introduction

In this diploma project, we investigate the development of adaptive routing algorithms for ad hoc wireless multi-hop networks using techniques from artificial intelligence, and in particular from swarm intelligence. The thesis has the following targets:

a) The implementation of an efficient algorithm that addresses the challenging task of routing in ad hoc wireless networks by taking advantage of artificial intelligence techniques.
b) Evaluation of the AntHocNet in the context of WSN.
c) Evaluation of the implemented algorithm and comparison with other protocols/algorithms.

The main challenge facing protocol designers is to create a high reliability and low energy tax, transportation protocol with the following properties: (a) low number of collisions and retransmissions, (b) low packet loss resulting in high packet delivery ratio, (c) low latency, and (d) high fault tolerance. In sensor networks, packet bursts can be dynamically and randomly initiated at any sensor node within the network which is expected to arrive to the sink once the occurrence of a given event has been detected.[4]

Recently, the increased use of wireless communication, is one of the most important developments in this area. Consequently, many different wireless technologies have grown explosively and made wireless communication networks one of the most important areas of research in computer science. In this thesis, we will focus on one important type of wireless networks: ad hoc wireless multi-hop networks (AHWMNs)[4]. AHWMNs are communication networks that consist of wireless nodes with an ad hoc displacement. The Nodes satisfy the following assumptions: they have routing capabilities, they can forward data packets towards other nodes and they can join and leave the network at any time[8].

WSNs, in many ways, can be likened to social groups (with nodes being constituents of these social groups) attempting to accomplish their tasks collectively (by simple neighbor-to neighbor interactions), in a decentralized manner, and in the absence of (external) central supervision. Bio-systems usually exhibit remarkable survivability and robustness to external stimuli and internal perturbations or loss of units, as well as excellent scaling properties. Adaptation is one of the major strengths bio-systems as they must respond to addition or removal of members, as well as to sudden changes in the environment. In this work we utilize ideas adopted from the behavioral tendencies of natural systems for designing robust network control techniques. Drawing inspiration from the collective behavior of social groups, local behavior can be dictated easier and an emergent global behavior of minimum congestion and direction of information flow to the sink can be determined. In this way, self-properties, e.g. self-organization and self-adaptation, are not implemented explicitly into individual devices or nodes, but emerge as a result of the design of the nature-inspired model[4].

In this thesis, we focus on the problem of routing in AHWMNs. Routing is the task of selecting and maintaining the paths in a network, and is responsible for connecting the remote source to the destination nodes. Routing is complicated in AHWMNs because they are:

a) Dynamic networks: The AHWMNs are dynamic because the connections between nodes in the network are set up in an unplanned manner, and are often changed while the network is in use so the routing algorithm should be adaptive.

b) The wireless communication is unreliable: Data and control packets can easily get lost during transmission, especially when multiple transmissions take place simultaneously and interfere with each other. A routing algorithm should be robust.

c) The network bandwidth, node processing power, memory, battery power are limited: A routing algorithm should be efficient.

d) Usually the network size is huge: With the ever growing numbers of portable wireless devices, many AHWMNs are expected to grow to very large sizes. Routing algorithms should be scalable.

To solve the challenging problem of routing in AHWMNs, we use techniques from swarm intelligence (SI)[8] and ant colony optimization (ACO)[8]. SI is the collective behavior of decentralized, self-organized systems, whose concept is from artificial intelligence that is focused on the design of algorithms inspired by the collective behavior of social insects and other animal societies. A subcategory of SI is ACO that takes its inspiration from the foraging behavior of ants living in colonies.

In particular, in this work we implement a protocol (AntHocNet[8]) that provides congestion avoidance by mimicking ants behavior, where packets are modeled as ants moving over a topological space, e.g. a sensor network. The main idea of the implementation of the AntHocNet is to 'guide' packets (ants) to flow towards the sink (global attractor), whilst trying to avoid congestion regions (obstacles). AntHocNet contains elements from both reactive and proactive routing. Specifically, it combines a reactive route setup process with a proactive route maintenance and improvement process. AntHocNet was in the first place inspired by the ACO approach to routing. This is evident in the way that it gathers, stores and uses routing information. Nevertheless, AntHocNet contains elements from distance vector routing. In particular, the information gathering process used in its proactive route maintenance and improvement process combines the route sampling strategy from ACO routing with an information bootstrapping process that is similar to the one used in distance vector routing algorithms. The way both approaches are combined is novel and allows the algorithm to get the best of both worlds[8].

In this thesis, we investigate how ideas from ACO routing can be used efficiently to build an adaptive routing algorithm for AHWMNs. Our aim is to develop a routing algorithm for AHWMNs that contains the advantages of adaptivity and robustness of ACO routing. Our algorithm is based on the one presented in[8], however we apply it in the field of WSNs. We provide an extensive investigation of the internal working of the algorithm, and we also carry out a detailed simulation study in a realistic environment. Finally, we compare the implementation of AntHocNet routing algorithms with the Flock-CC algorithm[3,4].

1.2 Contribution of this diploma project

This thesis has the following contributions:

- We have implemented the AntHocNet algorithm and adapted it for Wireless Sensor Networks. We have slightly modified the original algorithms to improve its performance.
- We present a thoroughly validation and evaluation of the implemented protocol and its properties when utilized in the context of WSN under various configuration and conditions.
- We present a quantitative comparison with another biologically inspired routing protocol.

1.3 Outline of the diploma project

The rest of this thesis is organized as follows: In Chapter 2 we provide background information and previous work related to the topics discussed in this thesis. In Chapter 3 we present a detailed description of the AntHocNet algorithm and our implementation. A detailed evaluation of the implemented algorithm is presented in Chapter 4 and Chapter 5 concludes this thesis and discusses future work.

Chapter 2

Literature Review

2.1 Ad Hoc Wireless Multi-hop Networks

Wireless communication technologies is ever increasingly utilized by many of the devices in our current times. More and more of the electronic devices is requiring to “communicate” with each other using technologies like Bluetooth, WiFi, etc. Wireless networks exist in many types and configurations. The emphasis of this thesis is on a specific type called Ad Hoc Wireless Multi-hop Networks (AHWMN)[8].

AHWMN is a network consisting of nodes that communicate exclusively through wireless connections, in which data can be forwarded over multiple hops. Such nodes are at least partly deployed in an *ad hoc* manner. Ad hoc deployment entails that little or no planning is needed, and so changes in the network (such as adding, moving or removing nodes) can be done with minimal extra work.[8]

Differences between AHWMNs and traditional telecommunication networks[8]:

a) The most important difference between AHWMNs and traditional telecommunication networks is that the topology of an AHWMN is dynamic. This means that it can be extended by adding new wireless nodes, reduced by removing nodes, or changed in a continuous way if some of the nodes are mobile. As such, no element of planning is involved, but rather the topology emerges from the dynamic placement of the nodes which are connected with each others over the wireless radio signal. Note that at any time, only a sub-part of the total nodes is “visible” by each node.

b) Another difference is that AHWMNs data’s transport is less reliable. Moreover there exist less available bandwidth because they rely on wireless links. And the nodes typically have limited resources (memory, power, etc..).

c) Furthermore, it is difficult to optimize the use of network resources because AHWMNs are decentralized.

d) AHWMNs are expected to grow to very large sizes, and so scalability is important.

AHWMNs are classified into:

a) Wireless mesh networks (WMNs) [13] are more heterogeneous than other AHWMN. They consist of mesh client nodes and mesh router nodes, that are less mobile but have more resources. Mesh clients are mobile, and usually communicate through one wireless interface (an antenna). They appear as end points of data traffic and as routers. Mesh routers are more static, and they are more powerful devices than the mesh clients. Moreover they support many different wireless technologies. Mesh routers can create a structured organization and can improve the applicability and the capacities of the network.

e) Mobile ad hoc Networks (MANETs) [11] are networks that are made up of a set of homogeneous mobile devices. They use an antenna to communicate through wireless connections. MANETs are dynamic, flat, and fully decentralized networks without central control or overview. MANET algorithms are typically highly adaptive to the ever changing environment, they are bound to be robust in order to deal with unreliable wireless transmissions, they should work in a fully distributed way and be efficient in their use of the limited network resources, such as bandwidth and power. All nodes have the capability to send/receive data and relay/route traffic because they are all homogeneous without a specialization.

f) Sensor networks [1] are AHWMNs that consist of wireless sensor nodes (WSNs). WSNs have been deployed for several mission-critical tasks (e.g. as platforms for health monitoring, process control, environmental observation, battlefield surveillance), and are expected to operate unattended for extended periods of time. Typically, WSNs comprises a small (and often cheap) cooperative devices (nodes), which may be (severely) constrained in terms of computation capability, memory space, communication bandwidth and energy.

   In the context of WSNs, autonomous nodes may interact (a) with the environment so as to sense or control physical parameters, and (b) with each other in order to exchange information or forward data towards one or more sink nodes. This mass of interactions, in conjunction with variable wireless network conditions, may result in unpredictable behavior in terms of traffic load variations and link capacity fluctuations. The network condition is worsened due to topology changes driven by node failures, mobility, or intentional misbehavior. These stressful situations are likely to occur in WSN environments, thus increasing their susceptibility to congestion. Problems specific to sensor networks stem from the fact that sensor nodes are small and have very limited resources for storage, processing and transmission, so that highly efficient algorithms are needed. This problem is acerbated by the fact that sensor batteries can often not be replaced. Moreover, the use of cheap, low power radio technology also means that communication is highly unreliable and irregular so algorithms have to be robust and should be able to deal with unidirectional links. Another issue in sensor networks is that their topology is usually very dynamic and so it is easy for failures to occur as the nodes are typically lightweight devices with limited power. What exacerbate the situation is the fact that often new sensor devices are added since in such networks vast numbers of nodes are typically deployed. Finally, data traffic patterns show certain characteristic regularities[3].

2.2 Wireless Sensor Networks

The unpredictable nature of WSNs necessitates robust, self-adaptive, and scalable mechanisms which are very basic to the mission of dependable WSNs. The focal point of this study is to design a robust and self-adaptive congestion control (CC) mechanism for delivering enhanced application fidelity at the sink (in terms of packet delivery ratio and delay) under varying network conditions, inspired by nature. This implementation should be simple at individual node level with minimal exchange of information. [4]

WSNs, in many ways, can be likened to social groups (with nodes being constituents of individuals within such groups) attempting to accomplish their tasks collectively (by simple neighbor-to neighbor interactions), in a decentralized manner, and in the absence of (external) central supervision. Bio-systems usually exhibit remarkable survivability and robustness to external stimuli and internal perturbations or loss of units, as well as excellent scaling properties. Adaptation is one of the major strengths bio-systems as they must respond to addition or removal of members, as well as to sudden changes in the environment. This study intends to explore what can be learned from the behavioral tendencies of natural systems for designing robust network control techniques. Drawing inspiration from the collective behavior of social groups, local behavior can be dictated easier and an emergent global behavior of minimum congestion and direction of information flow to the sink can be determined. In this way, self-properties, e.g. self-organization and self-adaptation, are not implemented explicitly into individual devices or nodes, but emerge as a result of the design of the nature-inspired CC model.

The problem of congestion in WSNs, refers to the symptoms and the consequences of congestion, and presents a number of recent CC approaches. Congestion occurs when a link or node is carrying so much data that its quality of service deteriorates. Typical effects include queuing delay, packet loss and blocking of new connections. A consequence of the latter two is that incremental increases in offer load lead either only to small increases in network throughput, or to an actual reduction in network throughput[3]. Network protocols which use aggressive retransmissions to compensate for packet loss tend to keep systems in a state of network congestion even after the initial load has been reduced to a level which would not normally have induced network congestion. Thus, networks using these protocols can exhibit two stable states under the same level of load. The stable state with low throughput is known as Congestive Collapse[23]. WSNs operate under large, sudden, and correlated-synchronized impulses of packets, that may suddenly arise in response to a detected or monitored event. All packets must be directed towards one or more sink nodes.

Types of congestion phenomena in WSNs:

1. Node-based: A node is accumulating packets in its buffer when its outgoing channel capacity is exceeded by a huge income of traffic. If the traffic is persistent then the buffer capacity of the node is not able to handle the long queues and as result it overflows occurring long delays.

2. Link-based: Link- based congestion can be caused from the multi-hop nature of WSNs, the shared communication medium and the limited bandwidth.

   In wireless networks, local channel contention arises in the vicinity of a sensor node due to the limited bandwidth and interference among multiple neighboring nodes that try to access the wireless medium simultaneously. As a result, the time variant nature of the outgoing channel capacity makes the congestion level fluctuating and unpredictable.[3]

Congestion Consequences are energy waste, throughput reduction, increase in collisions and retransmissions at the MAC layer, increase of queuing delays and even information loss, leading to the deterioration of the offered quality of service (QoS), decrease of network lifetime and even the decomposition of network topology in multiple components[3].

2.3 Swarm Intelligence

Real network topologies are typically very large with thousands of nodes, so algorithms handling such sizes need to scale well. Swarm intelligence (SI)[24] is inspired from nature that deals and solve successfully and effectively with similar types of complex problems. Swarm intelligence is the collective behavior of decentralized, self organized natural or artificial systems. SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such agents lead to the emergence of "intelligent" global behavior, unknown to the individual agents. Natural examples of SI include ant colonies, bird flocking, animal herding, bacterial growth etc. The application of swarm principles to robots is called swarm robotics, while 'swarm intelligence' refers to the more general set of algorithms.

SI techniques, motivated by the collective behavior of social insect societies living in decentralized, self-organizing, and adapting environments, reportedly provide a promising basis for computing environments that need to exhibit these characteristics [17]. Research in SI has provided computer scientists with powerful methods for designing distributed control and optimization algorithms. These methods are applied
successfully to a variety of scientific and engineering problems. In addition to achieving good performance on a wide spectrum of ‘static’ problems, swarm-based algorithms tend to exhibit a high degree of flexibility and robustness in dynamic environments [10]. Social groups found in nature (e.g. ant colonies, bird flocks, etc.) carry out their tasks collectively in order to contribute to a common goal. Even though

individuals coordinate to accomplish a given global mission in a complex world (e.g. foraging, migration, nest building, defend against predators, etc.), an individual has only local perception of the surrounding environment and exhibits specific behavioral tendencies which are governed by a few simple rules.

2.4 Ant Colony Optimizations

Routing is the task of selecting and maintaining the paths in a network, and is responsible to connect the remote source to the destination nodes. As mentioned before, routing is complicated in AHWMNs because they are:

a) Dynamic networks: The AHWMNs are dynamic because the connections between nodes in the network are set up in an unplanned manner, and are often changed while the network is in use so the routing algorithm should be adaptive.

b) The wireless communication is unreliable: Data and control packets can easily get lost during transmission, especially when multiple transmissions take place simultaneously and interfere with each other. A routing algorithm should be robust.

c) The network bandwidth, node processing power, memory, battery power are limited: A routing algorithm should be efficient.

d) Usually the network size is huge: With the ever growing numbers of portable wireless devices, many AHWMNs are expected to grow to very large sizes. Routing algorithms should be scalable.

To solve the challenging problem of routing in AHWMNs, we use techniques from swarm intelligence (SI) and ant colony optimization (ACO)[8]. A subcategory of SI is ACO that takes its inspiration from the foraging behavior of ants living in colonies[8].

The ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. In the natural world, when ants find food, they return to their colony while laying down pheromone trails so if other ants find such a path, they follow the trail, returning and reinforcing it if they eventually find food. The pheromone trail starts to evaporate over time, so its attractive strength is reduced. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate.

For example a short path, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. When one ant finds a good path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants following a single path. These paths then attract more ants, which in turn increases their pheromone level, until there is a convergence of the majority of the ants onto the shortest paths. The ants completing paths can be seen as repetitive samples of possible paths, while the laying and following of pheromone results in a collective learning process guided by implicit reinforcement of good solutions. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around our network topology representing control packets.[22].

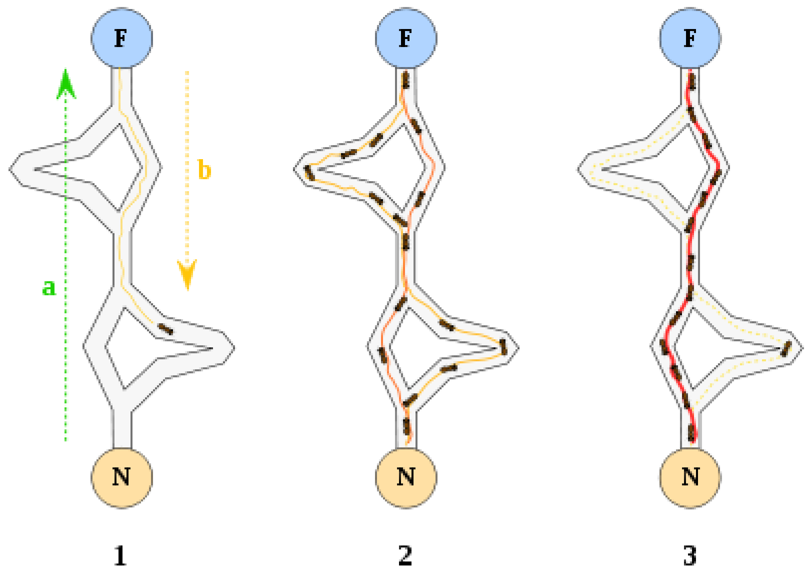

Figure1 shows an example of how ants find the shortest path between food source and the next[22].

1. The first ant finds the food source (F), via any way (a), then returns to the nest (N), leaving behind a trail pheromone (b)
2. Ants indiscriminately follow four possible ways, but the strengthening of the runway makes it more attractive as the shortest route.
3. Ants take the shortest route, long portions of other ways lose their trail pheromones.

In a series of experiments[22] on a colony of ants with a choice between two unequal length paths leading to a source of food, biologists have observed that ants tended to use the shortest route. A model explaining this behaviour is as follows:

1. An ant (called “blitz”) runs more or less at random around the colony;
2. If it discovers a food source, it returns more or less directly to the nest, leaving in its path a trail of pheromone;
3. These pheromones are attractive, nearby ants will be inclined to follow, more or less directly, the track;
4. Returning to the colony, these ants will strengthen the route;
5. If there are two routes to reach the same food source then, in a given amount of time, the shorter one will be traveled by more ants than the long route;
6. The short route will be increasingly enhanced, and therefore become more attractive;
7. The long route will eventually disappear because pheromones are volatile;
8. Eventually, all the ants have determined and therefore “chosen” the shortest route.

ACO algorithms for routing in networks differ from traditional algorithms. They gather routing information through the repetitive sampling of possible paths between source and destination nodes using artificial ant packets. Probabilistic distance vector tables, called pheromone tables, fulfill the role of pheromone in nature, with artificial ants being forwarded along them in a hop-by-hop way using stochastic forwarding decisions. Also data packets are forwarded stochastically using similar tables, resulting in automatic data load balancing[8].

ACO routing algorithms work in a highly distributed way, and have properties such as adaptivity, robustness and scalability. This makes them particularly interesting to deal with the complications of AHWMN routing we mention above[8].

As mentioned before the main source of inspiration behind ACO is a behavior that is displayed by ants finding the shortest path during foraging[17]. Finding the shortest path among several available paths is interesting because each individual ant is a simple creature, with very limited vision and computing power. The only way that this task can be accomplish is through the cooperation between the individuals in the colony. Pheromone is the key behind the colony level shortest path behavior and is a volatile chemical substance that is secreted by the ants in order to observe the behavior of other ants and of itself. Pheromones are used by many different species of social animals for a wide variety of tasks that involve coordinated behavior.

The shortest path finding process of the ants is highly distributed and self-organized. There is no central control mechanism; instead, the organization of the behavior emerges from the simple rules communications via pheromones. Second, it is highly robust. This is related to the property of self-organization: the system has no single point of failure, but instead consists of a high number of individually unimportant agents, so that even significant agent losses do not have a large impact on the performance. Third, the process is adaptive. Since none of the ant behavior is deterministic, and some individuals keep exploring also longer paths, the system can adapt to changes in the environment. Finally, the process is scalable: the process can be scaled to arbitrarily large colonies[8]

It is worthy to note that, the distributed shortest path finding process of foraging ants described above has been an important source of inspiration for artificial intelligence (AI) researchers. In particular, it was the basis for the development of the ACO meta-heuristic [16]. This is a general framework for the development of algorithms to solve optimization problems.

2.5 FlockCC Protocol

Flock Congestion Control protocol[3,4] is another routing protocol that draws inspiration from biological mechanisms. In particular, inspiration is drawn from the flocking and obstacle avoidance behavior of birds to ‘guide’ packets bypass obstacles like congested regions and dead node zones. Recent studies showed that the flock-based congestion control (Flock-CC) approach is robust, self-adaptable and energy-efficient, involving minimal information exchange and computational burden when used in uniform grid topologies. Flock-CC demonstrated robustness against failing nodes, and outperformed other congestion-aware routing approaches in terms of packet delivery ratio, end-to-end delay and energy tax.

The proposed approach mimics the flock flocking behavior of birds, where packets are modeled as birds flying over a topological space (sensor network). The packets are generated by sensor nodes and are ‘guided’ to form flocks and ‘fly’ towards a global attractor (sink), whilst trying to avoid obstacles (congested regions). The direction of motion is influenced by (a) repulsion and attraction forces exercised by neighboring packets, as well as (b) the gravitational force in the direction of the sink. The flock-based congestion control (Flock-CC) approach provides congestion detection on the basis of node and channel loading and traffic redirection over multiple paths[3].

The proposed approach involves reference to artificial bird flocks consisting of individuals with finite range of view (perception) which interact with each other as well as with the environment. The behavior of each individual is influenced by other individuals within its neighborhood. Typically, four simple behavioral rules govern the individual behavior:

1) repel from neighbors (if too close) to avoid collisions,
2) attract to neighbors (if apart) to maintain coherence among the members of a flock,
3) match velocity (speed and direction) with neighbors, and
4) introduce a random element that allows for exploration.

These behavioral rules govern individual-level interactions which collectively result in the emergence of group-level transitions[4].

2.6 Basic Definitions from Networking

In this section we provide the unfamiliar reader with a brief definition of the technical terms we use often in this thesis. Part of the definitions has been adapted from[8].

The physical layer

The physical layer is concerned with issues regarding the physical transmission of data between two nodes. The Physical Layer is the first and lowest layer of computer networking. And it consists of the basic hardware transmission technologies of a network. It is a fundamental layer underlying the logical data structures of the higher level functions in a network. The Physical Layer defines the means of transmitting raw bits rather than logical data packets over a physical link connecting network nodes.

The data link layer

The data link layer is concerned with the organization of transmission and retransmission of data between two nodes. The Data Link Layer corresponds to, or is part of the link layer of the TCP/IP reference model. The Data Link Layer is the protocol layer which transfers data between adjacent network nodes in a wide area network or between nodes on the same local area network segment. The Data Link Layer provides data transfer across the physical link. That transfer can be reliable or unreliable. Often, the data link layer is identified with its most important component, the medium access control (MAC) sub-layer. This component deals with the coordination of the access of different nodes to a shared communication medium. In AHWMNs, this comes down to avoiding interference while maximizing spatial reuse. A different goal, which is mainly important in sensor networks, is power saving.

The transport layer

The transport layer is situated above the routing layer. It is concerned with the organization of the transport of data between the source and destination nodes of a communication session over the path provided by the routing algorithm. In computer networking, the Transport Layer provides end-to-end communication services for applications within a layered architecture of network components and protocols. The transport layer provides convenient services such as connection-oriented data stream support, realiability, flow control and multiplexing.

The two main transport protocols in the internet are the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). TCP offers a secure transport service: a triple handshake mechanism is used to establish a connection at the start of the transport session, the arrival of each data packet is acknowledged by the receiver, warnings are sent from the destination to the source if packets arrive out of order, lost packets are retransmitted, and a congestion control mechanism is used to adapt the speed of the data flow in case there is packet loss. UDP, on the other hand, offers a service without any guarantees: it just sends packets on their way from source to destination, without using any of the mentioned mechanisms. UDP can be preferred over TCP for short data transports when the underlying network is very reliable. In what follows, we comment on the use of TCP and UDP in AHWMNs.

The Network Layer

The Network Layer provides the functional and procedural means of transferring variable length data sequences from a source host on one network to a destination host on a different network. The Network Layer performs fragmentation and reassembly, and report delivery errors, also it performs routing functions as a result it is responsible for routing packets delivery including routing through intermediate routers. Routers operate at this layer—sending data throughout the extended network and making the Internet possible. The Network Layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination host via one or more networks while maintaining the quality of service functions.

Network Routing

Routing is the task of directing data flows from source nodes to destination nodes while maximizing network performance. This is particularly complicated in the network has dynamic nature, the topology can change constantly, and paths between sources and destinations that were initially efficient can quickly become inefficient or even infeasible. This means that routing information should be updated more regularly than in traditional wired telecommunication networks, and is a problem because the network has limited bandwidth and node resources, and their possibly unreliable communication channels.

Routing metrics

Routing algorithms need a metric to evaluate different routes and choose one (or several) among them. The most straightforward choices are: (a) the end-to-end delay and (b) the hop count.

The end-to-end delay is usually unstable because the traffic for one neighbor suffers from high delays for the traffic for other neighbors. This happens because there is often only one wireless device, with one queue that is shared for all wireless links. Furthermore the differences in wireless connection quality can lead to high delay variations.

Compared to delay, the hop count is a very simple and stable metric. Hop count is not necessarily a good metric [9]. This is because paths with a low number of hops usually consist of long hops, which can be of low quality and break easily as a consequence of node movement, and because short paths tend to go through the center of the AHWMN area, where congestion and wireless channel contention is higher.

A number of other metrics have been proposed to evaluate paths.

a) In Associativity-Based Routing (ABR), nodes periodically broadcast beacon messages, and they count how many of these messages they have received from each of their neighbors.

b) Links with a high number of associativity ticks are considered more stable, and are therefore preferred.

c) Power aware routing algorithms evaluate paths based on their power usage. Different power based metrics are possible. One can e.g. choose the path which uses minimal power, or the path going over nodes with most power left.

d) An interesting newly proposed metric is the Expected Transmission Count (ETX). This metric estimates the expected number of transmissions that will be needed to successfully conclude the transfer of a data packet over a link and it includes possible retransmissions of the data packet due to failures, and the transmission of the acknowledgment packet in backward direction.

Address-centric versus data-centric routing

Address-centric routing is the normal way of operation in computer networks: data are routed through the network based on the address of their destination node.

Data-centric routing provides a radically different approach: destination addresses are not used, and data are instead forwarded based on their contents. Under data-centric routing, routing information does not indicate the next hop corresponding to a specific destination address, but corresponding to certain data properties.

Data-centric routing was in the first place developed for sensor networks, to support the typical communication patterns of such networks, where data measured by sensors spread over a wide area need to be drawn to one or more sink nodes. In this context, data-centric routing has some important advantages.

a) Data forwarding is made more efficient: sensor nodes only send the data that are requested by the sink node, and similar data coming from different sensors can be aggregated at intermediate nodes to travel more efficiently till the sink. The efficiency advantages due to aggregation have been confirmed via analytical modeling in.
b) No destination node address is used in the routing. This can help reduce the need for network wide topology information, as in directed diffusion
c) Eliminates the need for a global space of unique addresses. These are important elements to improve scalability and efficiency in very large sensor networks.[8]

Unicasting versus multicasting and broadcasting

In unicast routing protocols, a sender sends to one particular receiver. In multicast protocols, the group of receivers is a subset of the nodes of the network, where each member needs to be explicitly identified. Multicasting could be provided via the use of parallel unicast sessions between the source and each of the destinations. Such an approach would however be quite inefficient. A commonly used alternative is to built a routing structure between the members of the multicast receiver group, so that messages from the source can efficiently be forwarded between them. Finally, Broadcasting, has a group of receivers contains all the nodes in the network.

Proactive versus reactive routing algorithms

Traditionally, AHWMN routing protocols are classified as proactive or reactive protocols. In reactive routing protocols, nodes only gather the routing information that is strictly needed. This is the case when a new data session is started, or when a currently used path fails. Gathering routing information usually involves a route discovery or a route repair phase. Reactive algorithm can greatly reduce the overhead they create, so that they are in general more efficient. The disadvantage is that they are never prepared for disruptive events, and that some data packets can therefore incur large delays, e.g. during the route setup phase at the start of a communication session.

In proactive routing protocols, all nodes of the network try to maintain consistent routing information about all other nodes at all times. This means that routing information needs to be updated after each change in the network. Such algorithms have the advantage that routing information is always readily available when data needs to be sent. Also, all changes in the network are taken into account, so that new routing opportunities can be exploited, and backup paths can be provided when primary paths fail. On the downside, these algorithms can become quite inefficient or even break down completely when a lot of changes need to be tracked. This is the case when the topology is highly dynamic, or when the network is large[8].

Chapter 3

AntHocNet Algorithm

3.1 AntHocNet

AntHocNet is a hybrid, adaptive routing algorithm that utilizes both reactive and proactive routing. Specifically, it combines a reactive route setup process with a proactive route maintenance and improvement process. The way AntHocNet gathers, stores and uses routing information is inspired by the ACO approach to routing and from distance vector routing.

Next, we present an overview of the AntHocNet algorithm followed by a detailed examination of each of its components.

3.2 General description of the algorithm and terminologies

AntHocNet contains both reactive and proactive elements. The algorithm is reactive in the sense that it only gathers routing information about destinations that are involved in communication sessions. It is proactive in the sense that it tries to maintain and improve information about existing paths while the communication session is going on. Routing information is stored in pheromone tables. Forwarding of control and data packets is done in a stochastic way, using the information stored in the tables. Two tables are used.

Pheromone tables:

Each node i maintains one pheromone table Ti, which is a two-dimensional matrix. An entry T_{ij}^{d} of this pheromone table contains information about the route from node i to destination d over neighbor j:

a) This information includes the pheromone value τ_{ij}^{d}, which is a value indicating the relative goodness of going over node j when traveling from node i to destination d.
b) Virtual Pheromone value

Neighbor table:

This table keeps track of the wireless nodes to which it has a wireless link.

The algorithm is composed of two main parts the Reactive part and the Proactive part. Next is a description of the two parts of the algorithm. The algorithm is also depicted at all parts of Figure 3.

A) Reactive Route Setup

This part represents the reactive component of the algorithm. It starts at the beginning of a communication session.

1) The source node of the session controls its pheromone table, to see whether it has any routing information available for the requested destination.
2) If it does not, it starts a reactive route setup process, in which it sends an ant packet out over the network to find a route to the destination. Such an ant packet is called a reactive forward ant.
3) Each intermediate node that receives a copy of the reactive forward ant, forwards it. This is done via broadcasting in case the node does not have routing information about the ant's destination in its pheromone table. If routing information is available the packet is unicast to its neighbor.

Reactive forward ants:

1) As these ants travel the network towards the sink (via broadcasting or unicasting) they store the nodes that they have visited on their way in a list inside the packet.
2) The first copy of the reactive forward ant to reach the destination is converted into a reactive backward ant, while subsequent copies are destroyed.

Reactive backward ant:

1) Retraces the exact path that was followed by the forward ant back to the source.
2) On its way, it collects quality information about each of the links of the path.
3) At each intermediate node and at the source, it updates the routing tables based on this quality information.

B) Proactive route maintenance process:

1) This represents the proactive component of the algorithm. It tries to update, extend and improve the available routing information. This process runs for as long as the communication session is going on.
2) It consists of two different sub-processes: pheromone diffusion and proactive ant sampling.

Pheromone diffusion:

1) It is the first sub-process of proactive route maintenance. It can be considered a cheap but unreliable way of spreading pheromone information.
2) Spreads out pheromone information that was placed by the ants.
3) Nodes periodically broadcast messages containing the best pheromone information they have available.
4) Using information bootstrapping, neighboring nodes can then derive new pheromone for themselves and further forward it in their own periodic broadcasts.

Before continuing we define Virtual and Regular pheromones:

Virtual pheromone: is the pheromone that is obtained via pheromone diffusion and is kept separate from the normal pheromone placed by the ants because of its potential unreliability. It is obtained via diffusion and it is spread via proactive and sampling.

Regular pheromone: is the reliable pheromone placed by the ants.

<u>Proactive ant Sampling :</u>

1) This sub-process turns the virtual pheromone into reliable regular pheromone.
2) All nodes that are the source of a communication session periodically send out proactive forward ants towards the destination of the session.
3) These ants construct a path in a stochastic way, choosing a new next hop probabilistically at each intermediate node.
4) Different from reactive forward ants, they are never broadcast.
5) When calculating the probability of taking a next hop, proactive forward ants consider both regular and virtual pheromone.
6) This way, they can leave the routes that were followed by previous ants, and follow the (potentially unreliable) routes that have emerged from pheromone diffusion.
7) Once a proactive forward ant reaches the destination, it is converted into a proactive backward ant that travels back to the source and leaves pheromone along the way (regular, not virtual pheromone), just like reactive backward ants.
8) Proactive ants can follow virtual pheromone and then, once they have experienced that it leads to the destination, convert it into regular pheromone.

<u>Data packet forwarding:</u> The gathered pheromone values are used for data packet routing.

1) Routing decisions are taken hop-by-hop, based on the locally available pheromone.
2) Only regular pheromone is considered, as virtual pheromone is not considered reliable enough.

3) Each forwarding decision is taken using a stochastic formula that gives preference to next hops that are associated with higher pheromone values.

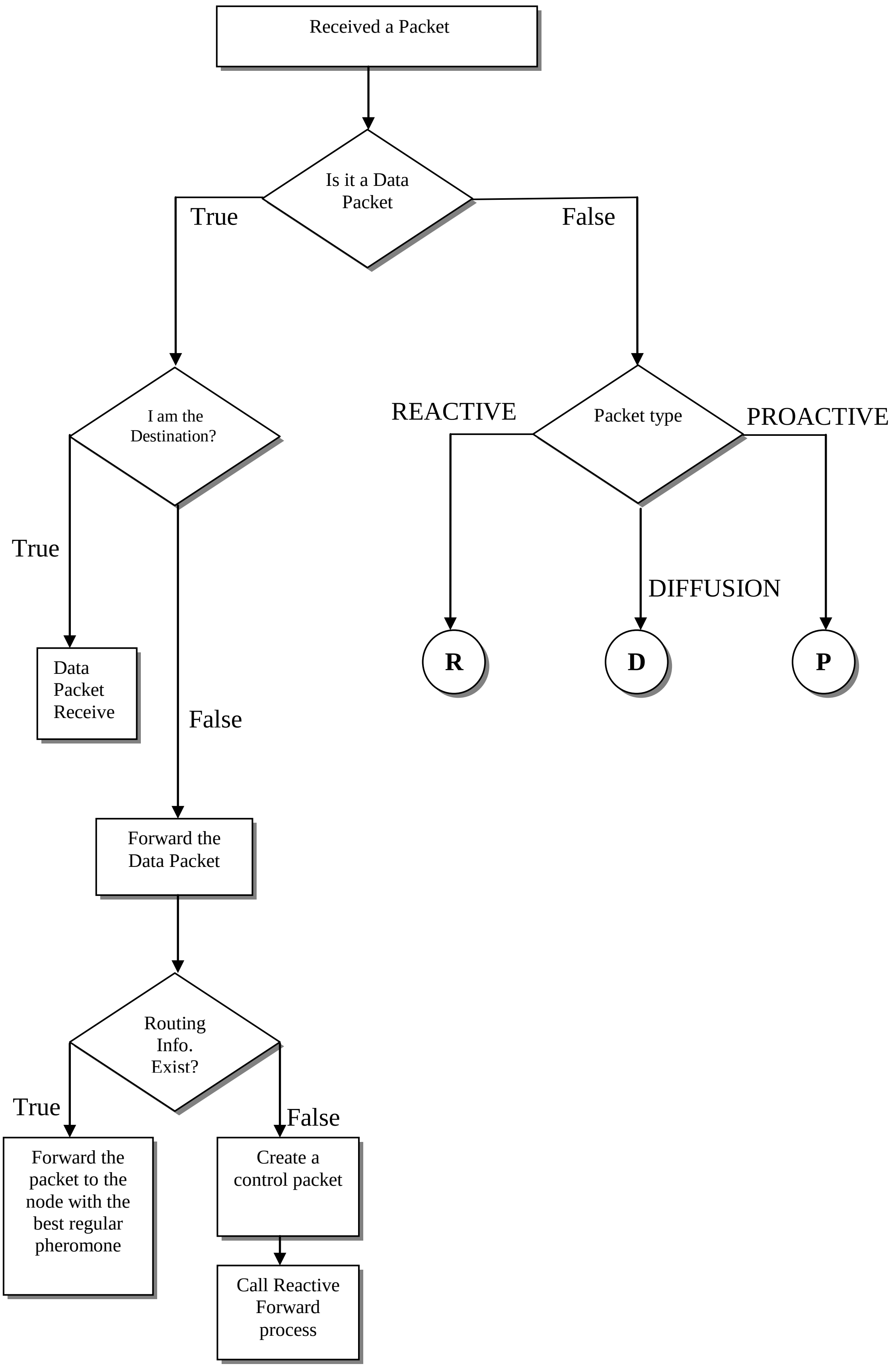

Figure 3.1: The algorithm of the process when a packet is received

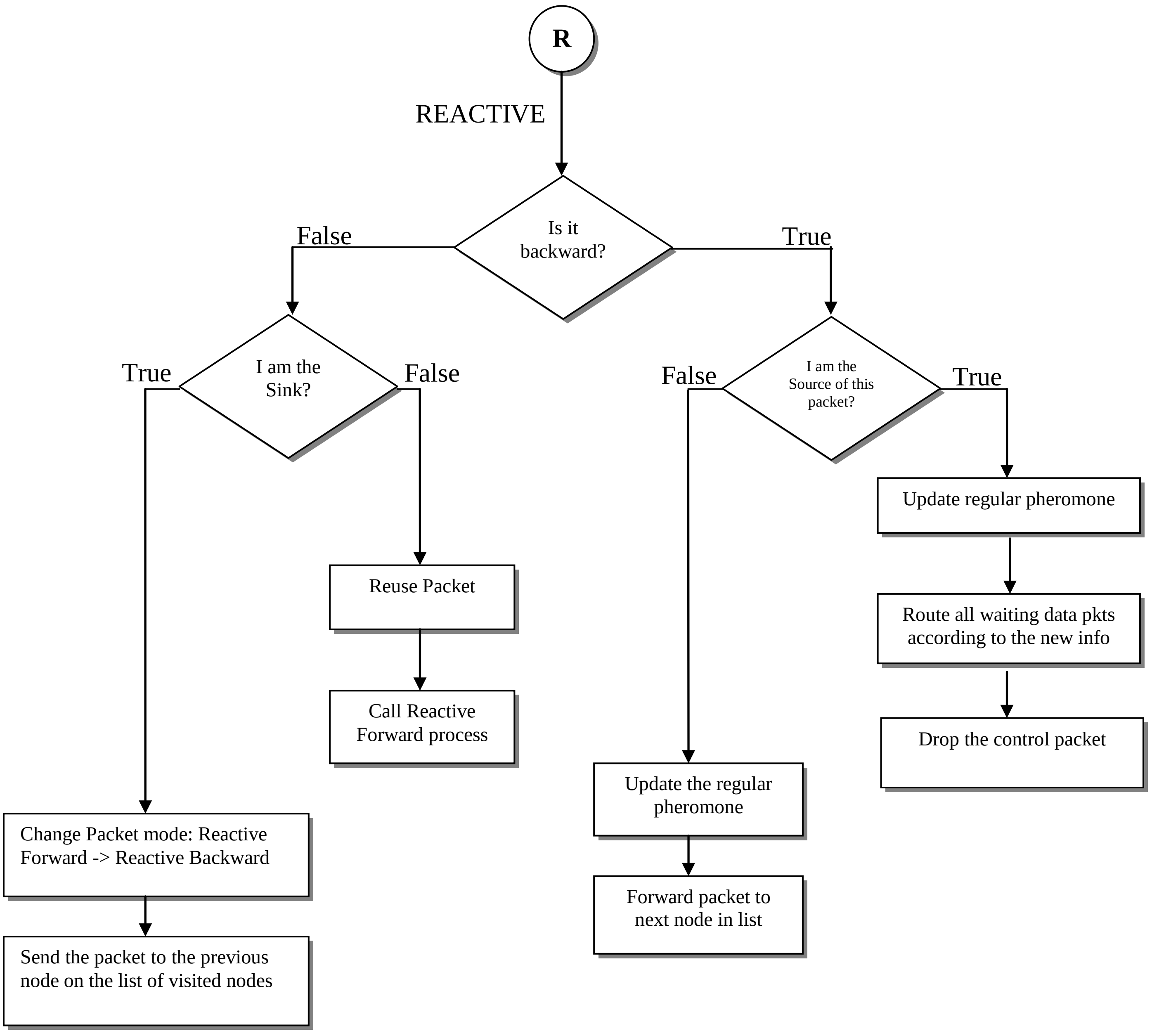
R
REACTIVE
Is it backward?
False
True
I am the Sink?
True
False
I am the Source of this packet?
False
True
Update regular pheromone
Reuse Packet
Route all waiting data pkts according to the new info
Call Reactive Forward process
Drop the control packet
Update the regular pheromone
Change Packet mode: Reactive Forward -> Reactive Backward
Forward packet to next node in list
Send the packet to the previous node on the list of visited nodes

Figure 3.2: The algorithm of the process when the packet received is reactive ant packet.

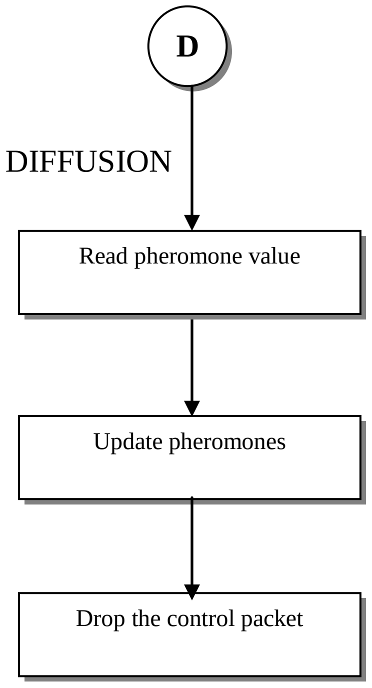

Figure 3.3: The algorithm of the process when the packet received is diffusion ant packet.

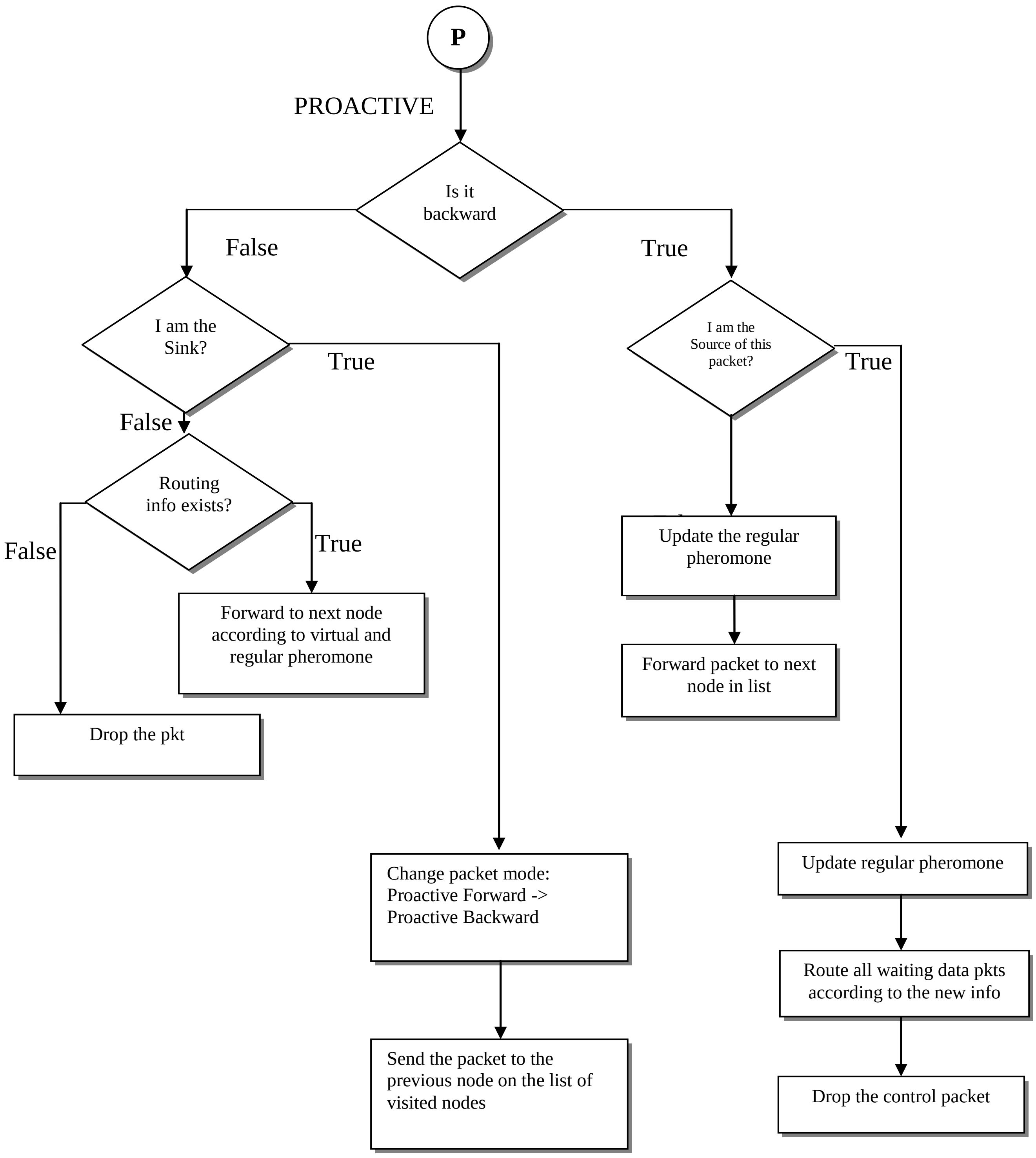
P
PROACTIVE
Is it backward
False
True
I am the Sink?
True
False
Routing info exists?
False
True
Forward to next node according to virtual and regular pheromone
Drop the pkt
Change packet mode: Proactive Forward -> Proactive Backward
Send the packet to the previous node on the list of visited nodes
I am the Source of this packet?
True
Update the regular pheromone
Forward packet to next node in list
Update regular pheromone
Route all waiting data pkts according to the new info
Drop the control packet

Figure 3.4: The algorithm of the process when the packet received is proactive ant packet.

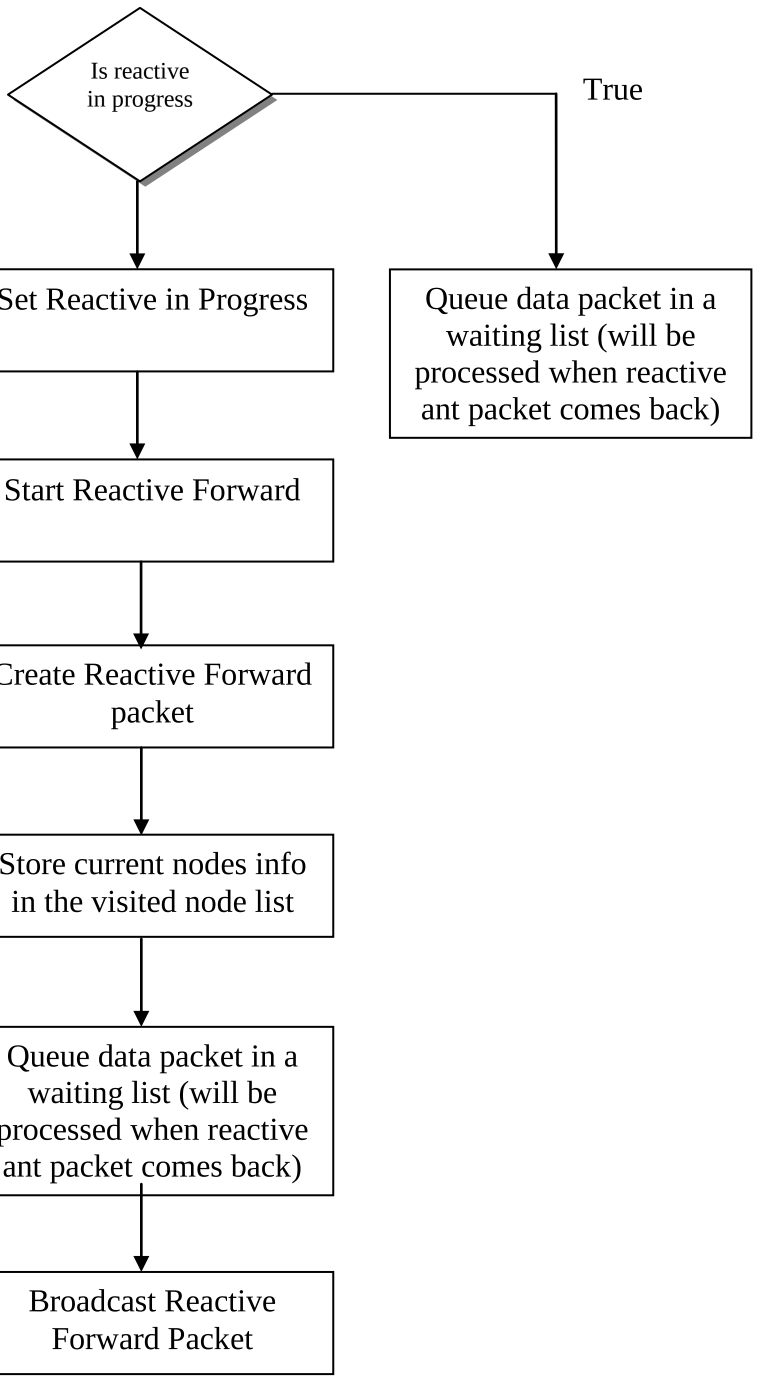
Is reactive in progress
True
Set Reactive in Progress
Queue data packet in a waiting list (will be processed when reactive ant packet comes back)
Start Reactive Forward
Create Reactive Forward packet
Store current nodes info in the visited node list
Queue data packet in a waiting list (will be processed when reactive ant packet comes back)
Broadcast Reactive Forward Packet

Figure 3.5: The algorithm of the Reactive process

3.3 Detailed description of the algorithm

In this section we provide in depth details on the data structures and components of the AntHocNet algorithm. The description here is based on the one presented in [8]. It is important to note that in [8] the algorithm assumes multiple destination (sink) nodes, however in the case of WSN we assume one destination (sink).

Data structures in AntHocNet

Pheromone table:

Each node i maintains a pheromone table Ti, which is a two dimensional matrix. An entry T_{ij}^{d} of this matrix contains information about the route from node i to destination d over neighbor j.

This includes: a) a regular pheromone value τ_{ij}^{d},

b)virtual pheromone value ω_{ij}^{d},

c) an average number of hops h_{ij}^{d}.

Regular pheromone value τ_{ij}^{d} is an estimate of the goodness of the route from i to d over j. Goodness is expressed as the inverse of a cost. Regular pheromone is updated by backward ants. These can be reactive, proactive or repair backward ants.

Virtual pheromone value ω_{ij}^{d} forms an alternative estimate of the goodness of the route from i to d over j. Virtual pheromone is obtained through information bootstrapping using goodness values reported by neighbor nodes during the proactive route maintenance process.

Average number of hops h_{ij}^{d} is like the regular pheromone, updated by backward ants. h_{ij}^{d} is used when deciding how long to wait for repair backward ants.

Neighbor table:

Each node i maintains a neighbor table N_i kept by node i as a one-dimensional vector with one entry for each of i' s neighbors. The entry N_{ij} corresponding to i' s neighbor j contains a time value th_{ij} that indicates when i last heard from j. Node i uses this time value to derive whether there is a wireless link with node j, and to detect link failures. Please note that the task of implementing the link failure is left as a future work.

Important Notes:

Nodes do not necessarily always have values available for τ_{ij}^d, ω_{ij}^d, h_{ij}^d because nodes do not maintain routing information about all possible destinations in the network and because for a specific destination, nodes do not necessarily have a route available over each of their possible neighbors. Since τ_{ij}^d and h_{ij}^d are both obtained from backward ants, nodes that have a value for one of the two will also have a value for the other. Since regular and virtual pheromones are obtained through different processes, it is possible that a node has a value for τ_{ij}^d but not for ω_{ij}^d or vice versa.

The approach of keeping virtual and regular pheromone separate means that bootstrapped pheromone is not used directly for the forwarding of data packets, since data packets only consider regular pheromone when choosing a next hop. Virtual pheromone is used when forwarding proactive forward ants towards their destination.

Reactive and proactive ants follow similar procedures with small differences. Unlike reactive forward ants, proactive forward ants rely both on regular and virtual pheromone for their routing decisions: they use the maximum between regular and virtual pheromone to calculate the probability of each next hop. Also different from reactive forward ants is that proactive forward ants are never broadcast: when they arrive at a node that does not have any routing information for their destination, they are discarded.

Reactive route setup:

The reactive route setup process is triggered whenever a node receives a data packet that was locally generated for a destination for which no routing information is available. We have no information when the data packet that is sent is the first of a new communication session and no routing information for its destination is available or because the data packet belongs to an ongoing session for which all routes have become invalid. The reactive route setup process involves the sending of a reactive forward ant from source to destination, and a reactive backward ant from destination to source. The reactive route setup consists of reactive forward and reactive backward ant.

The reactive route setup process described leads to the availability of a single route from the source of a communication session to its destination, indicated by regular pheromone values in the pheromone tables of the nodes. Moreover, each neighbor node of the destination also has a one-hop route to the destination.

<u>Reactive forward ants:</u>

1) At the start of the reactive route setup process, the source node s creates a reactive forward ant. This is a control packet that has as a goal to find a path from s to an assigned destination d.

2) At the start, the ant contains just the addresses of s and d. Later, as it proceeds through the network, it collects a list P = [1, 2, …, d - 1] of all the nodes that it has visited on its way from s to d.

3) After its creation at s, the reactive forward ant is broadcast, so that all of s's neighbors receive a copy of it. At each subsequent node, the ant is either unicast or broadcast, depending on whether the current node has routing information for d.

4) If routing information is available, the node chooses a next hop for the ant probabilistically, based on the different pheromone values associated with next hops for d.

$$P_{in}^{d} = \frac{(\tau_{in}^{d})^{\beta 1}}{\sum j \in N_i^d \, (\tau_{ij}^{d})^{\beta 1}}$$

$\beta 1 >= 1$

- node i
- chooses node n as next hop
- probability a node chose the next hop for the ant P_{in}^{d}
- N_i^d set of neighbors of i over which a path to d is known
- $\beta 1$ is a parameter value which can control the exploratory behavior of the ants (on 20, relatively high because we want to obtain the initial route as fast as possible, and limit the time we spend on exploration at this stage)

5) In case the intermediate node i does not have routing information for d, it broadcasts the reactive forward ant. Due to this broadcasting, a reactive forward ant can build up quickly over the network, with different copies of the ant following different paths to the destination.

6) In order to limit this overhead, nodes only forward the first copy of the ant that they receive. Subsequent copies are simply discarded.

7) At the destination, the reactive forward ant is converted into a reactive backward ant, which follows the list of nodes P visited by the forward ant back to s.

8) If more than one copy of the forward ant is received, only the first is accepted and converted into a backward ant, while subsequent copies are discarded. This way, only one route is set up during the reactive route setup process.

<u>Reactive backward ants</u>

1) The reactive backward ant created by the destination node in response to a reactive forward ant contains the addresses of the forward ant's source node s and destination node d, as well as the full list of nodes P that the forward ant has visited.

2) The reactive backward ant is unicast from d and among the nodes of P back to s.

3) The aim of the reactive backward ant is to update routing information in each of the nodes of P and in s. At each node i that it visits, it updates the number of hops and the regular pheromone value in the pheromone table entry $\mathbf{T}_{\mathbf{ij}}^{\mathbf{d}}$, where n is the node that it visited before i on its way from d.

4) The updating of number of hops is done using a moving average.

$$\mathrm{h}_{\mathrm{in}}^{\mathrm{d}} \leftarrow \mathrm{ah}_{\mathrm{in}}^{\mathrm{d}} + \ (1-\mathrm{a})\mathrm{h}$$

$$a \in [0,1]$$

- h is the number of hops that the backward ant has traveled between d and i
- α is a parameter regulating how quickly the formula adapts to new information (it is kept on 0.7)

5) Updating of the regular pheromone is done based on the cost of the route from i to d. This cost can be calculated using different metrics, such as the number of hops, the end-to-end delay, etc.. Here, we talk in terms of a generic cost $\boldsymbol{c}_{\boldsymbol{i}}^{\boldsymbol{j}}$,

where is the cost of the link from i to its neighbor j (it is the cost count from the number of hop). Under AntHocNet, each node maintains a local estimate of the cost c_i^j to go to each of its neighbors j.

6) The reactive backward ant reads at each node i the local estimate c_i^n of the cost to go from i to the next hop n that the ant is coming from.

7) The reactive backward ant adds this cost to the total cost c of the route from n to d (which it has been calculating on its way back from d), which is stored inside the ant.

8) The new cost c_i^d is used to update the pheromone value in node i, using the moving average formula of equation

$$\tau_{ij}^d \leftarrow \gamma\tau_{ij}^d + (1-\gamma)(c_i^d)^{-1}$$

$$\gamma \in [0,1]$$

- γ is a parameter regulating the speed of adaptation of the pheromone to new cost values. (γ was kept on 0.7)
- The cost value c_i^d is inverted to calculate the pheromone value τ_{ij}^d, as pheromone indicates the goodness of a route, rather than its cost.

Proactive route maintenance

The proactive route maintenance process serves to update and extend available routing information. In particular, it builds a mesh of multiple routes around the initial route. The proactive route maintenance process consists of: pheromone diffusion and proactive ant sampling.

Pheromone diffusion is aimed to spread all this pheromone information over the network through the use of periodic update messages and information bootstrapping, so that a field of pheromone pointing towards the destination becomes available in the network. This field of pheromone is indicated in the virtual pheromone values in the pheromone tables of the nodes. The fact that pheromone is spread out is similar to the normal diffusion of real pheromone in nature, which allows ants further away to sense it.

The pheromone diffusion is executed by all nodes throughout their whole lifetime, and is not particularly bound to the occurrence of a single session.

Proactive ant sampling is aimed at controlling and updating pheromone information through path sampling using proactive forward ants.

The proactive ant sampling is started by the source node of a communication session at the start of the session and continues for as long as the session is going on.

Pheromone diffusion

1) A crucial role in the pheromone diffusion process is played by hello messages. These are short messages broadcast every t_{hello} seconds asynchronously by all the nodes of the network throughout their whole lifetime. In AntHocNet, $\boldsymbol{t_{hello}}$ is set to 1 second. t_{hello} messages allow nodes to find which are their immediate neighbors so they can detect link failures and convey routing information inside these hello ant messages. While also in AntHocNet hello messages serve as the periodic update messages that are needed in the information bootstrapping process of pheromone diffusion. (The idea is to convey extra routing information inside hello messages.)

2) Nodes include in each hello message that they send out routing information they have available. In particular, a node i constructing a hello message consults its pheromone

table, and picks a maximum number k of destinations it has routing information for.

3) For each one of these destinations d, the hello message contains the address of d, the best pheromone value that i has available for d, v_i^d and a bit flag.

4) This best pheromone value v_i^d is taken over all possible values for regular pheromone and virtual pheromone associated with d in i's pheromone table Ti.

5) The bit flag is used to indicate whether the reported value was originally regular or virtual pheromone.

6) A neighboring node j receiving the hello message from i goes through the list of reported destinations. For each listed destination d, it derives from the hello message an estimate of the goodness of going from j to d over i, by applying information bootstrapping: it combines the reported pheromone value , which indicates the goodness of the best route from i to d, with the locally maintained estimate of the cost c of hopping from j to i.

$$\kappa_{ji}^d = ((v_i^d) + c_j^i)^{-1}$$

- v_i^d goodness value
- c_j^i cost value
- κ_{ji}^d the result of the calculation is what we call the bootstrapped pheromone value

7) With κ_{ji}^d, node j has obtained a new estimate for the goodness of the path to d over i in a relatively cheap way. Thanks to the use of information bootstrapping, all that was needed in terms of communication overhead was the sending of the value v_i^d from i to j.

8) Node j maintains in its pheromone table entry two distinct pheromone values for the route over its neighbor i to destination d: the regular pheromone value and the virtual pheromone. Of these, only the virtual pheromone value is normally updated with the new bootstrapped pheromone value κ. This way, the pheromone obtained via the pheromone diffusion process is kept separate from the regular pheromone, which is the product of ant based route sampling and is therefore considered more reliable. The updating is done by replacing ω by κ.

9) If the regular pheromone of a node is empty and it has no information, the regular pheromone is obtained via the pheromone diffusion process. This updating replacement improves our algorithm because instead of no information it has a less reliable information stored. Furthermore we keep a flag so with the first chance to update the regular pheromone with reliable information.

10) When reaching the destination, proactive forward ants are converted into proactive backward ants, which do deposit regular pheromone, which in turn is used for routing data packets. So, in this way, bootstrapped pheromone influences data forwarding indirectly. One could say that the potentially unreliable bootstrapped pheromone provides hints about possible routes, which are then explored and verified by the proactive forward ants.

11) There is one situation that forms an exception to this normal mode of operation, in which we do allow the bootstrapped pheromone value κ to be used for updating the regular pheromone value τ and influencing data forwarding directly. This is the case when the following two conditions are fulfilled:

a) j already has a non-zero value for the regular pheromone τ

b) the bootstrapped pheromone κ was derived from a reported pheromone value v that was based on regular pheromone in i, rather than virtual pheromone .

12) When a node receives a diffusion, hello message packet it update its tables with the new information and it deletes the ant packet.

Proactive ant sampling

1) The proactive ant sampling process is started by the source node of a session at the moment the first data packet of a new session is received, and continues for as long as the session is going on. The aim of the process is to use ant based sampling to gather routing information for ongoing sessions.

2) To this end, proactive forward ants are generated. These ants can follow regular pheromone, which is routing information placed by previous ants, or virtual pheromone, which is the result of the pheromone diffusion process described above.

3) While the former leads the ants to update goodness estimates of existing routes, the latter allows them to find new routes based on the hints provided by the pheromone diffusion process. This way, the single route that was initially constructed in the reactive route setup process is extended to a full mesh of multiple paths.

4) Each node which is the source of a communication session periodically (normally, we use a period of t_{hello} seconds) schedules the transmission of a proactive forward ant towards the session's destination.

5) In order to improve efficiency, the actual sending of a proactive forward ant is conditional to the availability of good new virtual pheromone: only if the best virtual pheromone is significantly better than the best regular pheromone, a proactive forward ant is sent out. In our experiments: we have found that the best threshold to use was: at least 20% better. This avoids congestion due to overloading the network with proactive forward packets, which occurred at smaller threshold values like 5% or 10%.

6) The aim of the proactive forward ant is to find a route towards the destination, and to store the list of nodes P that it visits on the way. The proactive forward ant takes a new routing decision at each intermediate node i, using the formula

of equation to calculate the probability of choosing each possible next hop n.

$$P_{in}^{d} = \frac{[\max(\tau_{in}^{d}, \omega_{in}^{d})]^{\beta 2}}{\sum_{j \in N_i^d} [\max(\tau_{ij}^{d}, \omega_{ij}^{d})]^{\beta 2}}$$

- β2>=1
- the function max(a,b) takes the maximum of the two values a and b
- β2 is a parameter that deffnes the exploratory character of the ants (is normally kept 20)
- list of nodes P that it visits on the way
- node i
- next hop n

7) When a proactive forward ant arrives at its destination, it is converted into a proactive backward ant, which is sent back to the source. Proactive backward ants have the same behavior as reactive backward ants:
    - follow the exact path P recorded by their corresponding forward ant back to the source
    - update regular pheromone entries at intermediate nodes and at the source

8) An important aspect to note here is that while the proactive forward ants can follow both regular and virtual pheromone, proactive backward ants always deposit regular pheromone. This way, the proactive ant sampling process can investigate promising virtual pheromone, and if the investigation is successful turn it into a regular route that can be used for data.

Data packet forwarding

1) Data packets are forwarded from their source to their destination in hop-by-hop fashion, taking a new routing decision at each intermediate node.

2) Routing decisions for data packets are based only on regular pheromone. This means that they only follow the reliable routes that are the result of ant based sampling, and leave the virtual pheromone information that is the result of information bootstrapping out of consideration.

3) The combination of the reactive route setup and the proactive route maintenance processes leads to the availability of a full mesh of such reliable routes between the source and destination of each session. Nodes in AntHocNet forward data packets stochastically, based on the relative values of the different regular pheromone entries they have available for the packet's destination.

$$P_{dn} = \frac{(\tau_{in}^{d})}{\sum_{j \in N_i^d} (\tau_{ij}^{d})^{\beta 3}}^{\beta 3}$$

- β3>=1
- β3 for the power function of the pheromone values (is normally set 20)
- probability P for a node i to pick next hop n when forwarding a packet with destination d
- node i
- next hop n
- destination d

4) By adapting the β3 parameter, one can make data forwarding less or more greedy with respect to the best available routes.

By setting β3 low, data is spread over multiple routes, considering also low quality ones. Using multiple routes for data forwarding can improve throughput, as the data load is spread more evenly over the available network resources.

By setting β3 high, on the other hand, data is concentrated on the best routes. This can also be a good choice, since the routes that according to the ant sampling give the best performance, are exploited as much as possible.

3.4 AntHocNet Parameters

In this section we present a brief recap of the parameters controlling the inner work of the AntHocNet algorithm.

β1 is a parameter value which can control the exploratory behavior of the ants (on 20, relatively high because we want to obtain the initial route as fast as possible, and limit the time we spend on exploration at this stage)

$$P^{d}_{in} = \frac{(\tau^{d}_{in})^{\beta 1}}{\sum j \in N^{d}_{i} (\tau^{d}_{ij})^{\beta 1}} \qquad \beta 1 >= 1$$

β2 is a parameter that defines the exploratory character of the ants (is normally kept 20)

$$P^{d}_{in} = \frac{[\max(\tau^{d}_{in}, \omega^{d}_{in})]}{\sum_{j \in N^{d}_{i}} [\max(\tau^{d}_{ij}, \omega^{d}_{ij})]^{\beta 2}}^{\beta 2} \qquad \beta 2 >= 1$$

β3 for the power function of the pheromone values (is normally set 20)

$$P_{dn} = \frac{(\tau^{d}_{in})}{\sum_{j \in N^{d}_{i}} (\tau^{d}_{ij})^{\beta 3}}^{\beta 3} \qquad \beta 3 >= 1$$

α is a parameter regulating how quickly the formula adapts to new information (it is kept on 0.7)

$$h_{in}^{d} \leftarrow ah_{in}^{d} + (1 - a)h \qquad a \in [0,1]$$

γ is a parameter regulating the speed of adaptation of the pheromone to new cost values. (γ was kept on 0.7)

$$\tau_{ij}^{d} \leftarrow \gamma\tau_{ij}^{d} + (1 - \gamma)(c_{i}^{d})^{-1} \qquad \gamma \in [0,1]$$

Chapter 4

Evaluation

4.1 Evaluation

In this chapter we present the evaluation of the implemented AntHocNet protocol. We show the results of a large set of simulations, in which we introduce variations to the parameters controlling the inner behavior of the protocol under a number of different scenarios. Moreover, we analyze the effect of the different components of the AntHocNet and finally we compare the performance of the AntHocNet protocol with the FlockCC protocol under three different scenarios.

We have used the ns-2 network simulator which facilitated a fair comparison with existing algorithms (in particular the FlockCC).

The rest of this chapter is organized as follows. First, in section 4.2, we describe the setup of the evaluation study. Next, in section 4.3, we present results of the experimental investigation of the internal working of AntHocNet. Finally, in section 4.4 we present the results of the comparison with the FlockCC routing algorithm.

4.2 Setup

As it is both expensive and complex to build a real AHWMN testbed for research, it is more convenient to use a simulator for the evaluation of research in this area. Moreover, a simulation makes it much easier to control all of the parameters involved (which might not be always possible in a real environment) and perform a large number of repeatable tests.

4.3 The NS-2 Simulator

A number of different network simulator software packages are available to the research community. These include ns-2 [19], OPNET [20], SWAN [15] and QualNet [21]. For the work presented here, we have used the ns-2 simulator. This is an open source discrete event network simulator developed by Scalable. Ns-2 is typically used for simulating routing protocols, especially in ad-hoc networking research (including wired and wireless networks alike). This encouraged our use of the simulator in addition to the fact that it is free and open source. Moreover, the implementation and evaluation of the protocol we are comparing against (FlockCC) have been performed in this simulator, which makes the comparison between the two protocols more fair. In Appendix A, we list the files we added to the simulator to implement the AntHocNet protocol. In addition, we list the existing simulator files we have changed and the addition/modification information. The actual files (both new added files and modified existing simulator files) will be provided on an accompanying electronic storage medium.

4.4 The Simulation Scenarios

NS-2 utilizes a special TCL script is used to instantiate the main simulator object and create the elements of the simulation scenario. The term scenario refers to a specific definition of the set of the variables affecting all aspects of the simulation. The main elements of the scenario include: the properties of the nodes and the events that will take part during the simulation.

The main properties of the nodes include: the number of the nodes (the size of the network), the displacement and mobility of the nodes (stationary or on the move), wireless range, energy model, queue type, the rate of data traffic, the routing protocol supported by the nodes and the internal parameters of the selected routing protocol.

The events of the simulation scenario are actions that occur at specific moments in during the simulation time and mainly include: the registration/connection of the node in the network (starting of the node), the sending of data, failing of the nodes (injected failure of the nodes), disconnection of the nodes (stopping of the node).

In all the scenarios we ran, we set the size of the network to 300 nodes displaced in a lattice of 20 rows and 15 columns. The distance between the node was set to 50 meters. For all the scenarios one node is designated as the destination of all the data packets (we refer to this node as the sink) and a number of nodes are chosen to be the data sending nodes (each call a source). Each such node starts a data session and sends a number of packets every second. The rate the data packets are sent is called the Constant Bit Rate (CBR). Each of the nodes implements the AntCC routing protocol and is capable of point-to-point or broadcasting of data or control packets. For a specific scenario all the nodes have the same properties. For the simulation of radio propagation, we used the two-ray signal propagation model (commonly used for modeling the propagation of wireless signals in open space [18]). The two-ray model assumes receiver gets the signal over two paths: a direct one and one reflected over the ground. At the physical layer, we use the IEEE 802.11 protocol. At the MAC layer, we use the CSMA IEEE 802.11 protocol with 2 Mbps/s transmission rate. We assume a radio range of 50m. For the transport layer, we use the UDP protocol.

We have used three different scenarios for our evaluations. The first scenario (scenario 1) has 10 source nodes at the back of the network sending packet for the sink. All the source nodes start sending the data around the 10th second of the simulation. This resembles a quite common case where an external stimuli causes the triggering of nearby nodes at the same time. The second scenario (scenario 2) has 10 source nodes, with 5 of them placed at the back of the network and 5 at the front. The first 5 nodes start sending at the 10th second and the second 5 nodes start sending at the 50th second. At the 70th second the first 5 nodes stop sending data packets. The third scenario (scenario 3) is similar to scenario 1, however at the 40th second of the simulation a large number of nodes (between the source nodes and the sink) fail. All the scenarios ran for 100 seconds. For each of the scenarios we use thee different (rather high) CBR values in our simulations: 25 packet/second, 35 packet/second and 45 packet/second. These CBR rates represent slightly congested, congested and heavily congested cases of network traffic. The size of the network, duration of simulation, and the CBR value that we select for the simulation represent highly demanding scenarios of usage for the network and thus can be considered a real indicator of the performance of the protocols simulated at hand. Figure 7 illustrates the 2D spatial displacement of the nodes in each scenario.

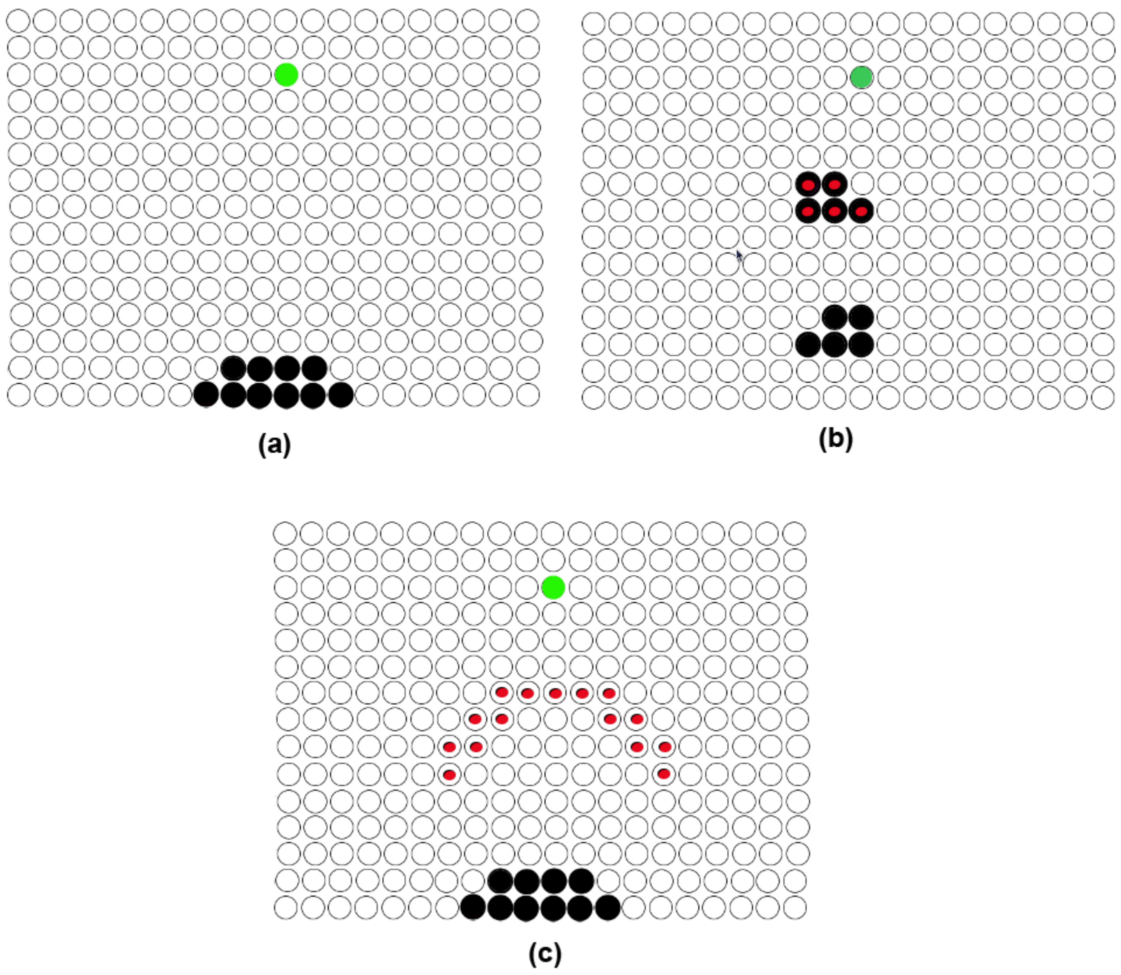

Figure 7: 2D spatial placement of the nodes in (a) scenario 1 (b) scenario 2 (c) scenario 3. Sender nodes are shown in black. Sink nodes are shown in green. Failing or deactivated nodes are shown with a red center.

4.5 Evaluation metrics

In this section, we describe the metrics we use to evaluate the performance of the routing algorithms. The first metric is the data delivery ratio. This is the ratio of correctly delivered data packets out of the total sent packets. Remember that in AHWMNs, due to the dropping of part of the packets (due to congestion for example) it might not be possible to deliver all the data packets. The second metric is the end-to-end packet delay. This is the accumulation of the delays experienced by packets traveling between their source and destination. The third metric is the Mean Energy which measures the amount of energy (in Joul) dissipated by all the nodes during the simulation.

4.6 Internal Analysis of the protocol

In this section we evaluate the internal properties of the implemented protocol. First, we perform an exploration of the parameter space of the variables controlling the work of the algorithm. Second, we evaluate the benefits of the different stages of the protocol.

4.7 Parameterization

A number of parameters control the working of the AntCC protocol (see section 3.4). We perform an exploration of the parameter space to find the best combination that yields the optimal results in terms the metric we have chosen. While one could focus on one or more metrics, we chose to focus on finding the optimal combination that maximizes the delivery ratio. In addition to finding the optimal combination, we try to validate the correctness of our implementation by comparing the best combination we find with the ones reported in[8].

To limit the size of the search space (as we have 5 parameters, 3 CBR rates and 3 different scenarios), instead of performing an exhaustive search, we optimize our search by starting from the parameter values reported in [8] and every time we vary one of the parameters while leaving the rest at the reported values. This is scheme is not without shortcomings, however, it leverages the results reported at [8] to perform a reasonable exploration of the parameter space within the limited time and resources we have.

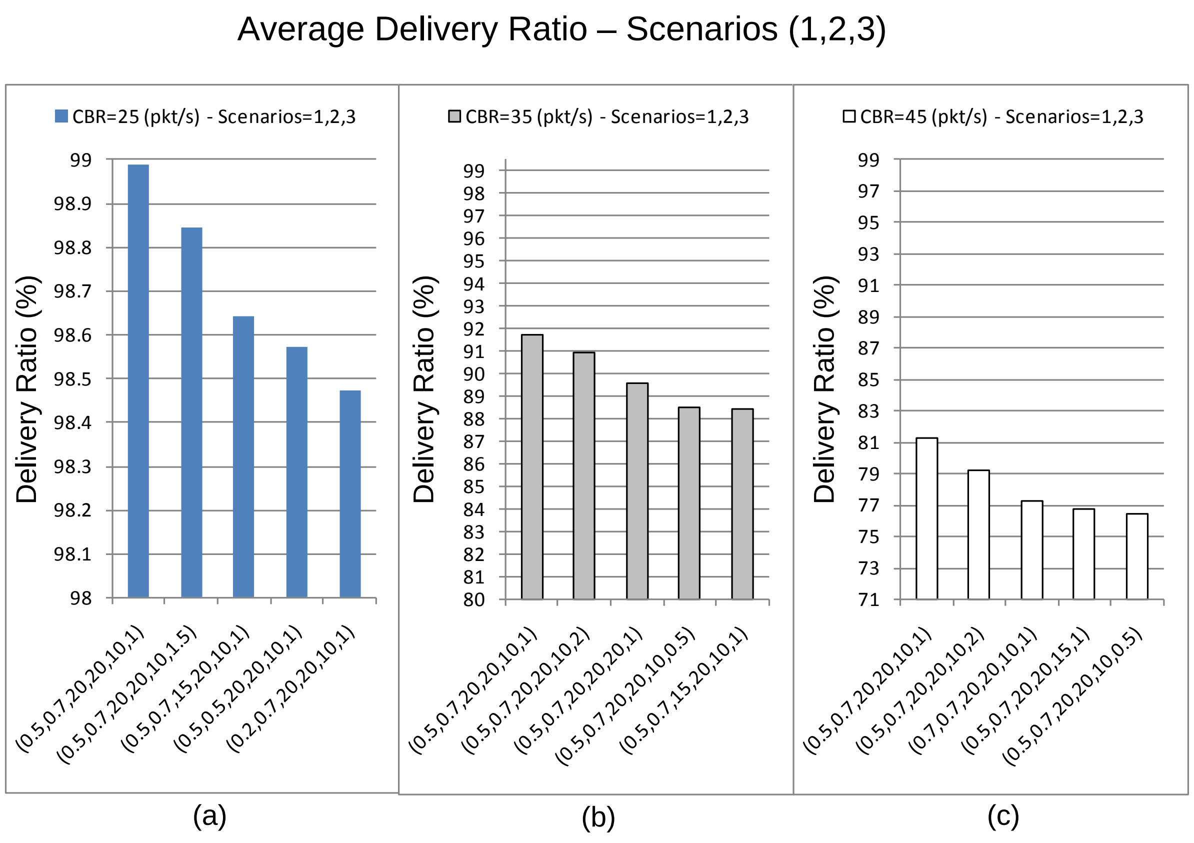

Figure 8: the delivery ratios for the best five parameter combinations for CBR rates (25, 35 and 45 pkt/sec). The parameters combination format is (α,γ,β1,β2,β3,t).

We have performed simulations for all the resulting combinations of the five parameters for scenarios 1, 2 and 3 and the CBR rates of 25, 35, and 45. For each combination we calculated the average delivery ratio for the three scenarios per a specific CBR rate. We used that average to rank the combinations. Figure 8 reports our results. In the figure, we show the best five parameter combinations for each CBR rate. The results show that the best combination in each of the three CBR rates is (α=0.5, γ =0.7, β1=20, β2=20, β3=10 and t=1) which is the same combination reported in [8], which validates the correctness of our implementation. Moreover, notice also that the 2nd best combination has the first four parameters set to the same values as the ones in the best combination, with the fifth parameter set to a higher value for parameter t. In addition to confirming the optimality of the combination we found, this also supports our earlier observation (see 3.3) that less frequent activation of proactive (induced by a higher value of t) leads to better performance in general as it causes less traffic in the network.

We have found that the results for the average end-to-end delay metric corresponded generally to the results of the average delivery ratio, i.e. the parameters leading to better delivery ratio also yields better (smaller) end-to-end delay with minor exceptions. We did not find a similar correspondence or the lack-of in the results of the mean energy metric.

As this combination of parameters yielded the best performance in terms of delivery ratio, we use them in all the following evaluation experiments.

To further validate our implementation and understand its properties we show in Figures 9, 10 and 11 a visual representation of the number of packets arriving at each node at a certain instant of time (snapshot) during the simulation. The snapshot was taken at the 30th second of the simulation (around the end of the first third of the simulation time) for all the scenarios and CBR values. One can notice immediately the similarity of the packet routes in the figures to the tracks of ants in nature. More importantly, it appears that the routes are relatively long and sparse which indicates that the algorithm does not necessarily favor short routes. Moreover, as the congestion rate increases the distribution of the packet to the routes becomes more uniform. This can be seen clearly in scenarios 1 and 3, as for both scenarios when the CBR rate increases the color distribution that corresponds to the amount of visited nodes (the lighter the color of the node the more packets it has received and vice versa) becomes more uniform.

In Figure 10, it can be seen that only the five top sender nodes are active at T=30s since the other five sender nodes are not activated until T=50s. Figure 12 shows what happens for this scenario at T=90s. It can be seen that the algorithm manages to respond to the activation of the five sender nodes at the bottom (and the deactivation of the upper five at T=70s) and build new routes to deliver the packets to the sink. The same can be inferred from Figures 11 and 13; in Figure 11 at T=30s all sender nodes are active and the rest of the nodes function normally and so we can see that the algorithm is successfully routing the packet to the sink. However, at T=40s a large number of nodes in the middle fail, thus cutting some of the already established routes.

Figure 13 shows what happens at T=90s (60 seconds after the failure). The gap in the figure appears in the place of the failing nodes. It can be seen from the figure that despite the disruption of some of the routes, the algorithm manages to successfully route the packet along the other ones circumventing the gap. It is interesting to note that the algorithm routed the packet via the routes that survived the disconnection and did not build new auxiliary routes.

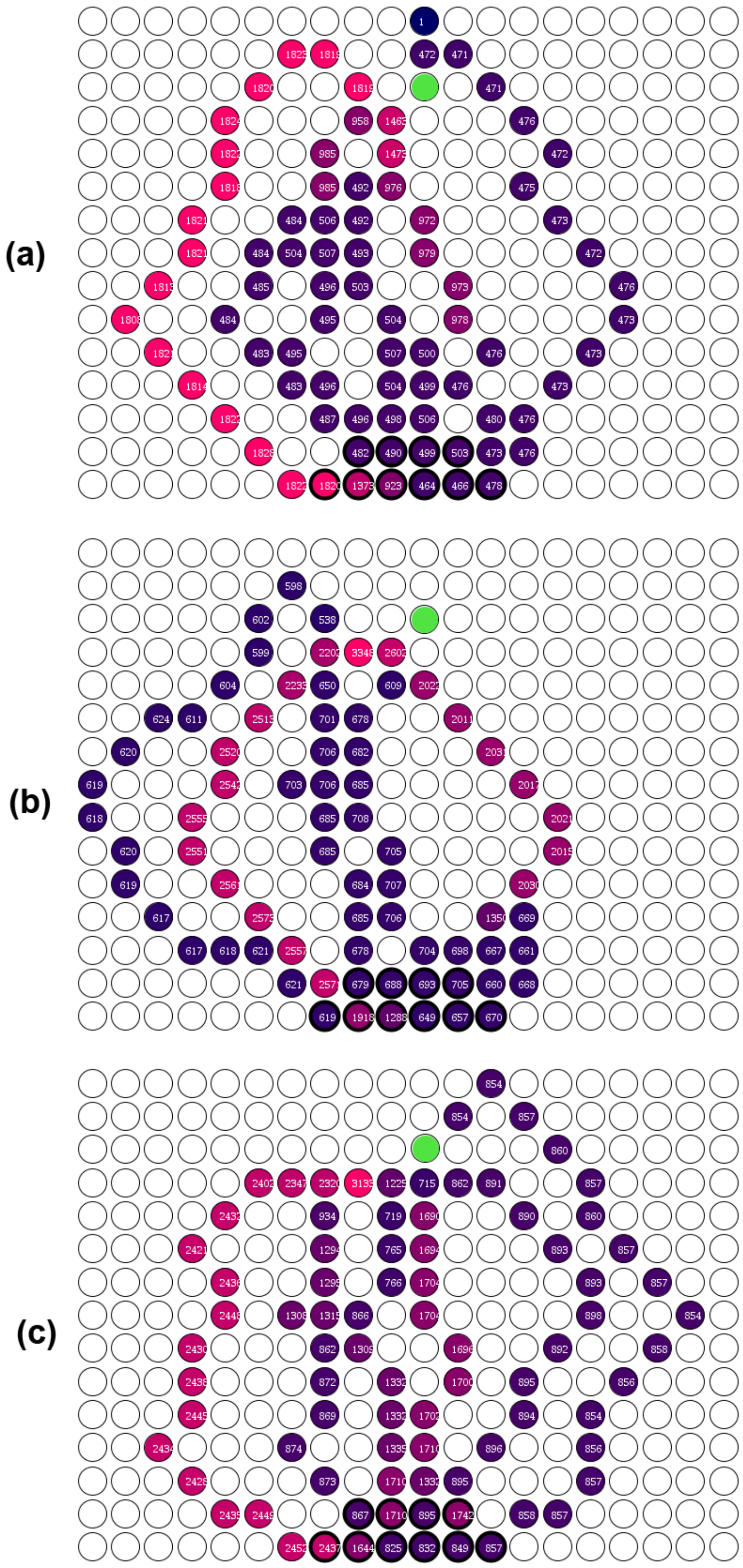

Figure 9: Visual representation of the number of packets arrived to each nodes for scenario 1 at T=30, for (a) CBR=25 pkt/sec (b) CBR=35 pkt/sec (c) CBR=45 pkt/sec

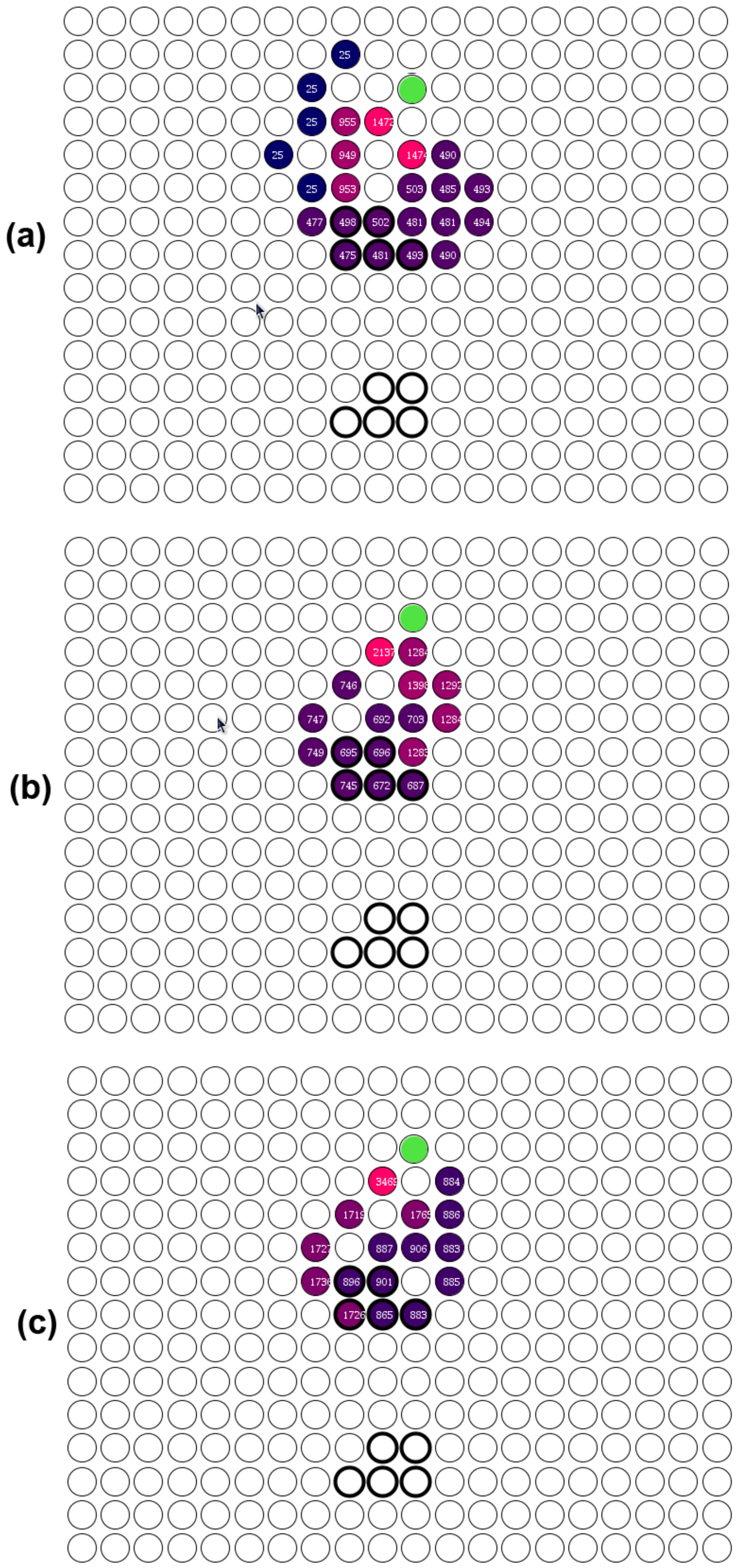

Figure 10: Visual representation of the number of packets arrived to each nodes for scenario 2 at T=30, for (a) CBR=25 pkt/sec (b) CBR=35 pkt/sec (c) CBR=45 pkt/sec

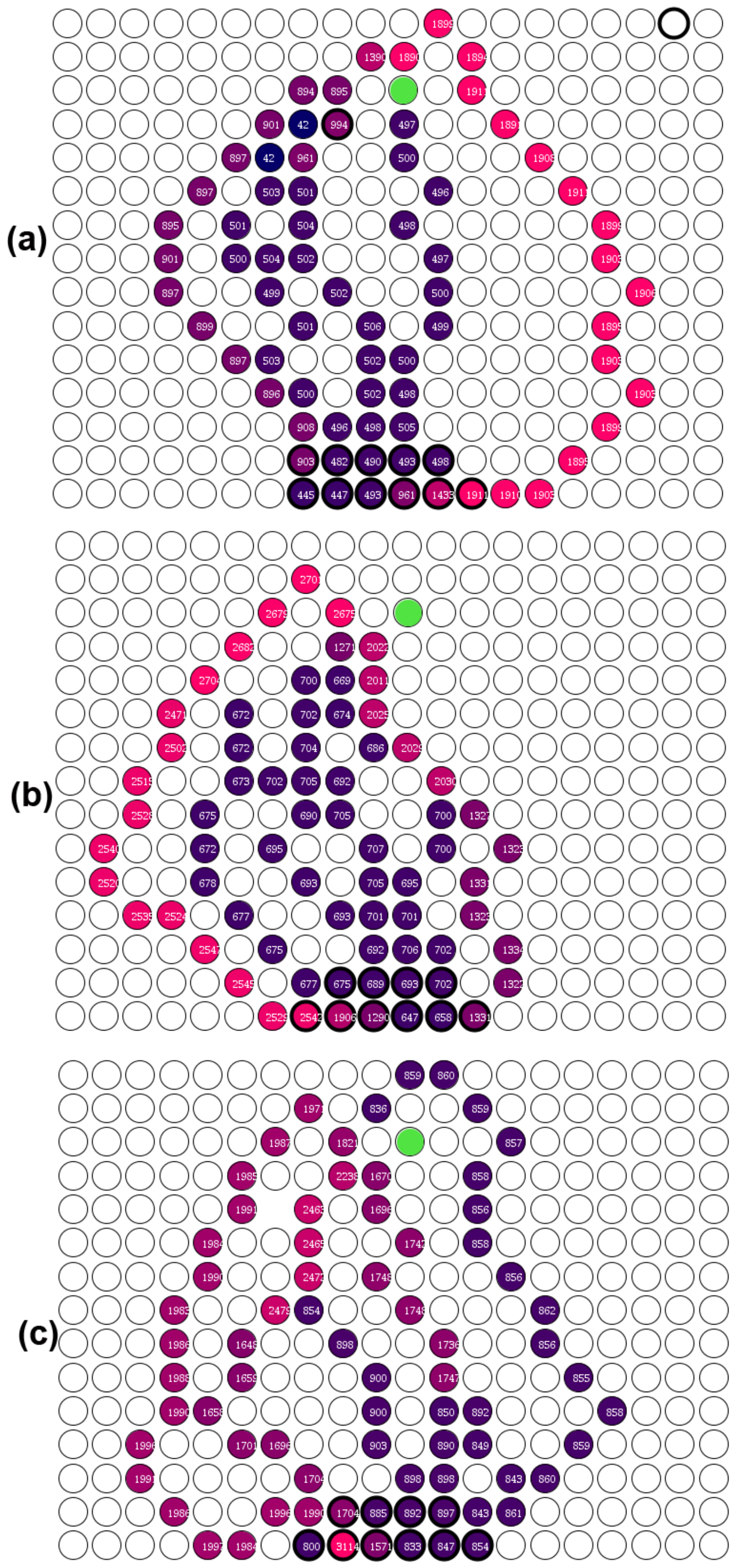

Figure 11: Visual representation of the number of packets arrived to each nodes for scenario 3 at T=30, for (a) CBR=25 pkt/sec (b) CBR=35 pkt/sec (c) CBR=45 pkt/sec

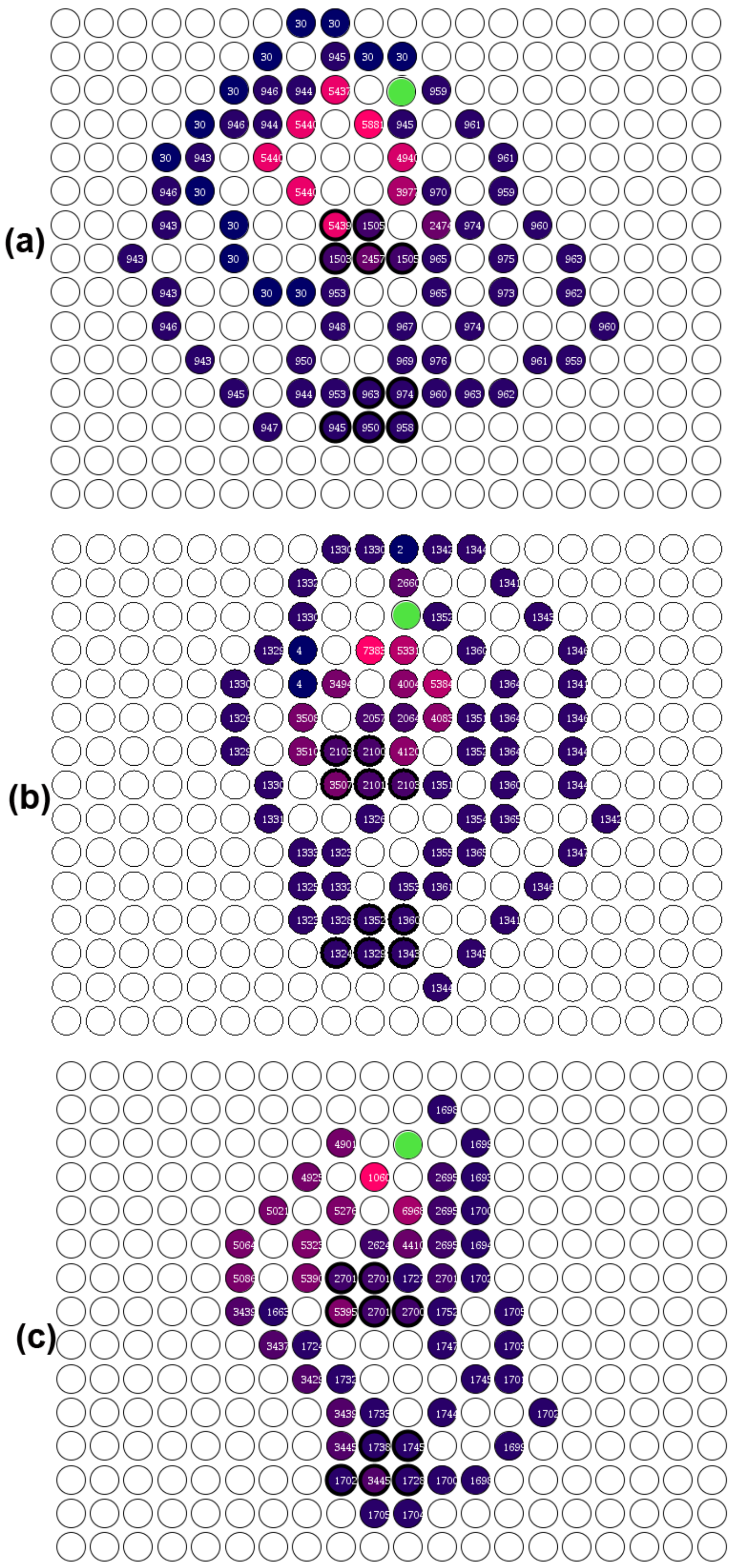

Figure 12: Visual representation of the number of packets arrived to each nodes for scenario 2 at T=90, for (a) CBR=25 pkt/sec (b) CBR=35 pkt/sec (c) CBR=45 pkt/sec

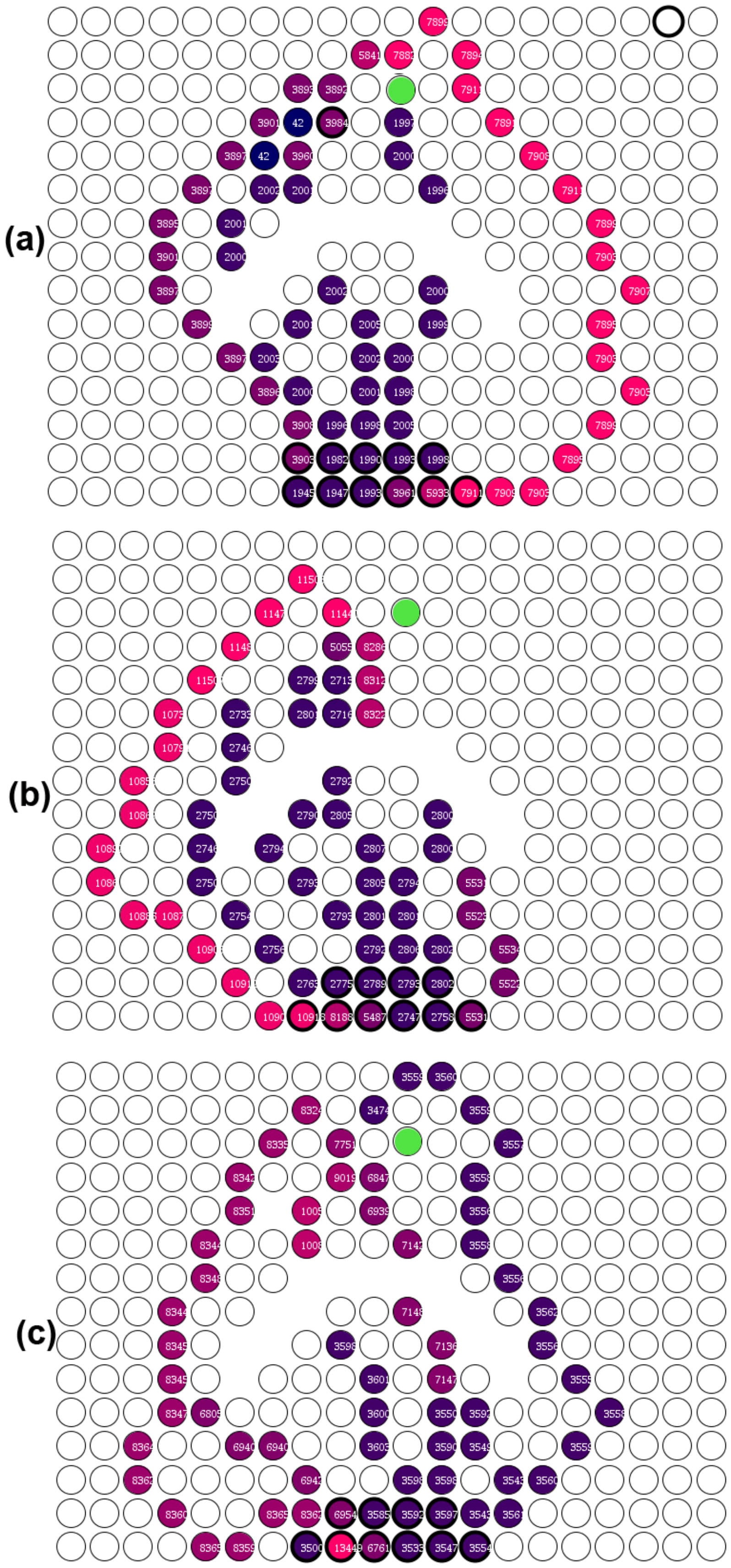

Figure 13: Visual representation of the number of packets arrived to each nodes for scenario 3 at T=90, for (a) CBR=25 pkt/sec (b) CBR=35 pkt/sec (c) CBR=45 pkt/sec

4.8 Reactive, Diffusion and Proactive Evaluation

In this section we evaluate the different stages of the AntHocNet algorithm. We executed all the scenarios with the CBR rates of 25 pkt/sec, 35 pkt/sec and 45 pkt/sec under three modes: the first only enabling the Reactive part of the algorithm, the second with both Reactive and Diffusion on and the third one with all stages of the algorithm on, i.e. with Reactive, Diffusion and Proactive. We refer to the three modes as R, RD and RDP respectively. The results are illustrated at Figures 14.

In Figure 14 we show the results of the delivery ratio and the End-to-End Delay. In Figure 14-a the delivery ratio results show that as the CBR rate increases the delivery ratio decreases which is expected, as due to the congestion less packet arrives to the sink. More importantly, the results show that on general the full mode of the AntHocNet protocol (RDP) performs the best. The advantage of the RDP mode compared to the R mode is less clear when the network is slightly congested (CBR=25 pkt/s) and heavily congested (CBR=45 pkt/s). We attribute that to the fact that in the first case the paths constructed by the Reactive stage maintain their good quality and any news paths discovered by the Proactive stage doesn't add much value. In the second case, although the Proactive stage might discover better (less congested paths) the extra traffic generated by the diffusion/proactive stages degrades the performance.

Figure 14-b End-End delay (%) – Scenarios 1, 2, 3 – CBR (25, 35, 45) shows the End-to-End delay results. It is clear from the figure that the End-to-End delay results correspond to the Delivery Ratio results. In particular, low delivery rates are generally associated with higher end-to-end delay values. Note that this need not be the case always as a good quality route (and hence one that achieves better delivery ratio) can be a long one which leads to higher end-to-end delay.

It is important to note in the original description of the algorithm in[8] the diffusion doesn't yield better results on its own compared to the reactive, as it doesn't not contribute to the real pheromone value. However, as we slightly modified the algorithm to change the real pheromone value in certain cases (see 3.3), the diffusion mode (mode RD) achieves an improved result relative to the reactive in one case (scenario 1, CBR=35 pkt/s). In the rest of the cases mode RD only adds the extra overhead of the diffusion packets without adding a real advantage and thus it achieves less performance than mode R.

Figure 15 shows the corresponding results for the Mean Energy. It is clear from the figure that for the same scenario, as the CBR rate increases, the mean energy increases, since when the CBR rate is higher more packets are sent, consequently, the congestion is increased and the buffer-overflows and retransmissions and so more energy is consumed.

Another thing to note is that the second scenario consumes far more energy than the first and the third, since less source nodes are sending packet across time for that scenario (remember that half of the source nodes start only in the 2nd half of the simulation and the other half stops sending at the start of the 3rd quarter of the simulation time). It is also worthy to note the similarity in the energy signature for the scenario 1 and scenario 3, which is expected as both scenarios are similar except for the nodes that fail before the end of the 2nd half of the simulation (this similarity actually applies to the other metrics -shown in Figure 15 as well).

Third, in the congested case where the benefits of the proactive is clear the proactive consumes less energy than the reactive although it involves more work as it manages to discover less congested paths while the reactive suffers from more buffer-overflows and re-transmission and so consumes more energy. In the cases where the benefits of the proactive is less obvious (as explained before in the previous section) the energy consumption in the Reactive mode is less than that of the RD or RDP in general.

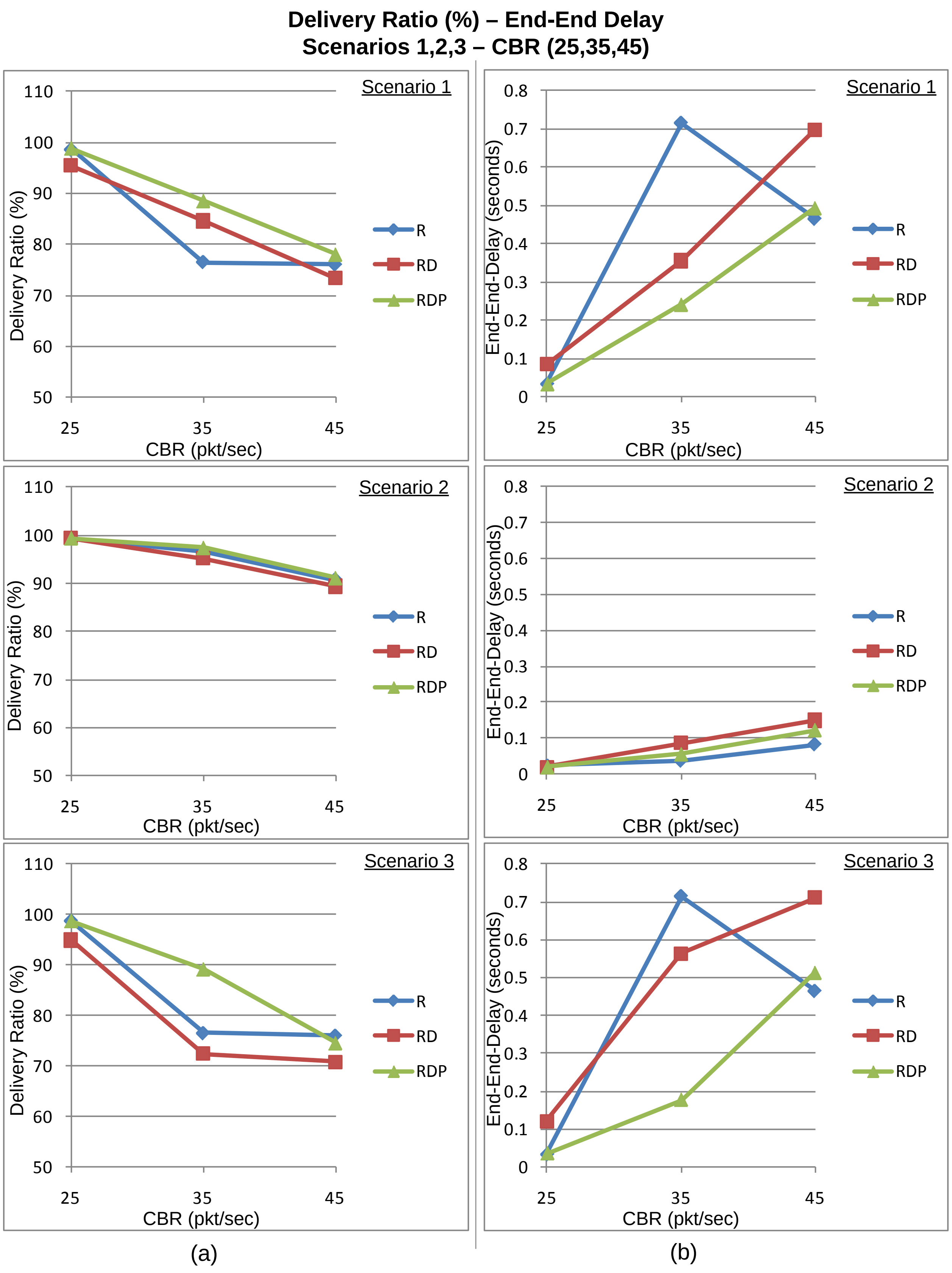

Figure 14: (a) Delivery Ratio (b) End-to-End delay. For scenarios 1,2,3 and CBR=25,35,45 pkt/sec

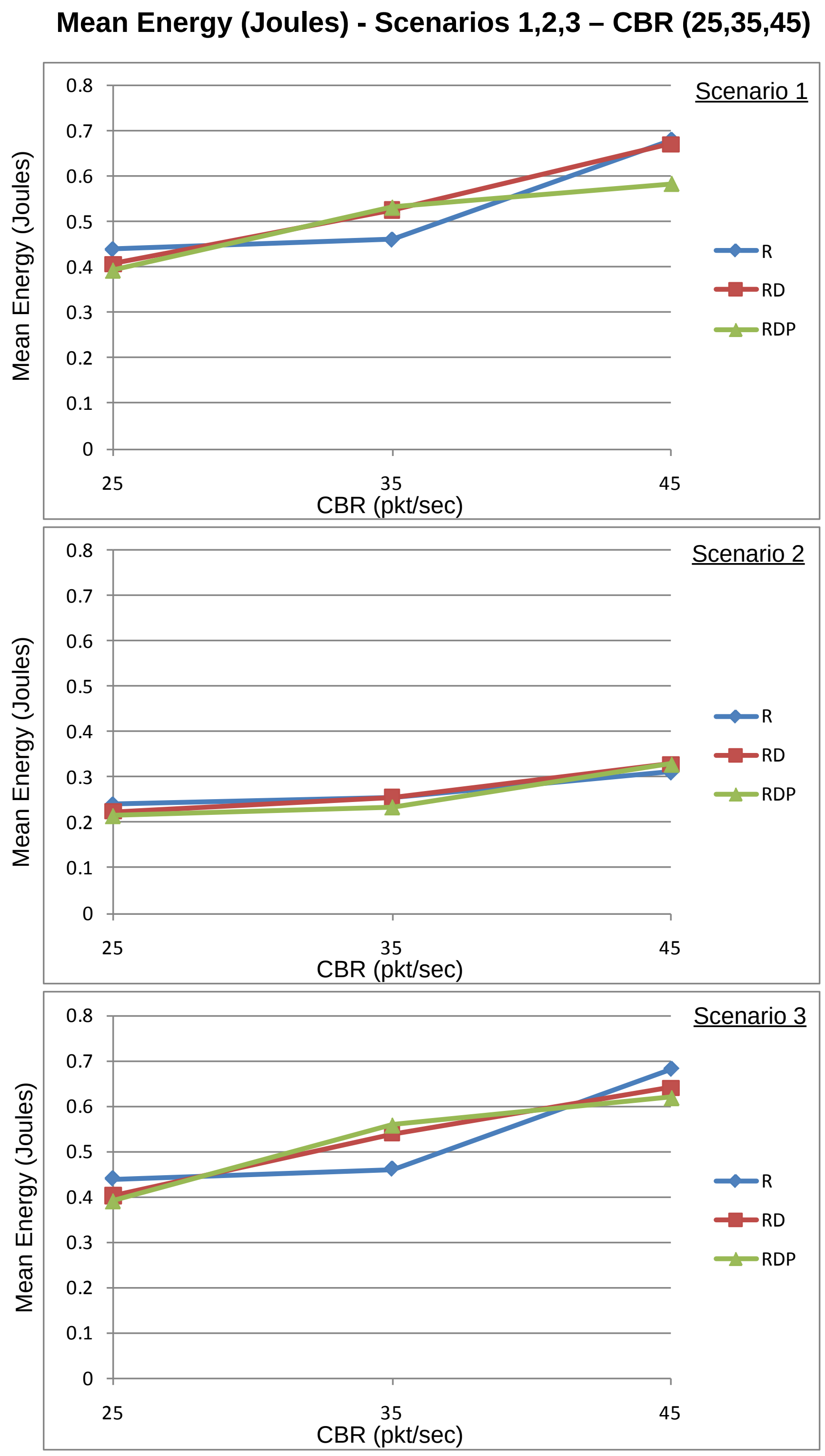

Figure 15: Mean Energy for scenarios 1,2,3 and CBR = 25,35,45 pkt/sec

4.9 Comparison with the FlockCC protocol

In this section we present the results of comparison between the AntHocNet and the FlockCC [3] protocol. We test both protocols for all the scenarios and the different CBR rates. For the AntHocNet protocol we use the protocol parameters combination found in section 4.7 that yielded the best performance (in term of delivery ratio). Similarly, for the FlockCC, which is controlled via two parameters (T and τ), we used the combination yielding best performance (in terms of delivery ratio) for each CBR value. For a CBR of 25 pkt/sec we have used the combination (T=0.5 and τ=1). For CBR = 35 and 45 pkt/sec, we have used the combination (T=0.5 and τ=0.75).

Figures Delivery Ratio (%) Scenarios 1, 2, 3 – CBR (25, 35, 45), End-to-End Delay (sec) Scenarios 1, 2, 3 – CBR (25, 35, 45) and Mean Energy (Joule) Scenarios 1, 2, 3 – CBR (25, 35, 45) depicts the results of the comparison.

Figure 16 illustrates a comparison of the delivery ratio for both protocols for the range of scenarios and CBR values. The results show that for both protocols when the CBR increases and thus the congestion in the network increases the delivery ratio decreases which is expected. Most importantly, the results show that for the tested scenarios and network configuration, the AntHocNet achieves better performance than the FlockCC especially when the amount of congestion in the network is high. This is clear when the CBR value is high (CBR=35 and 45 pkt/sec). Another indication of this is that for scenario 2, which has fewer source nodes sending packets over time (leading to a generally less congested network), the advantage of the AntHocNet over FlockCC is smaller (on all CBR values) compared to the relatively more congested 1 & 3scenarios.

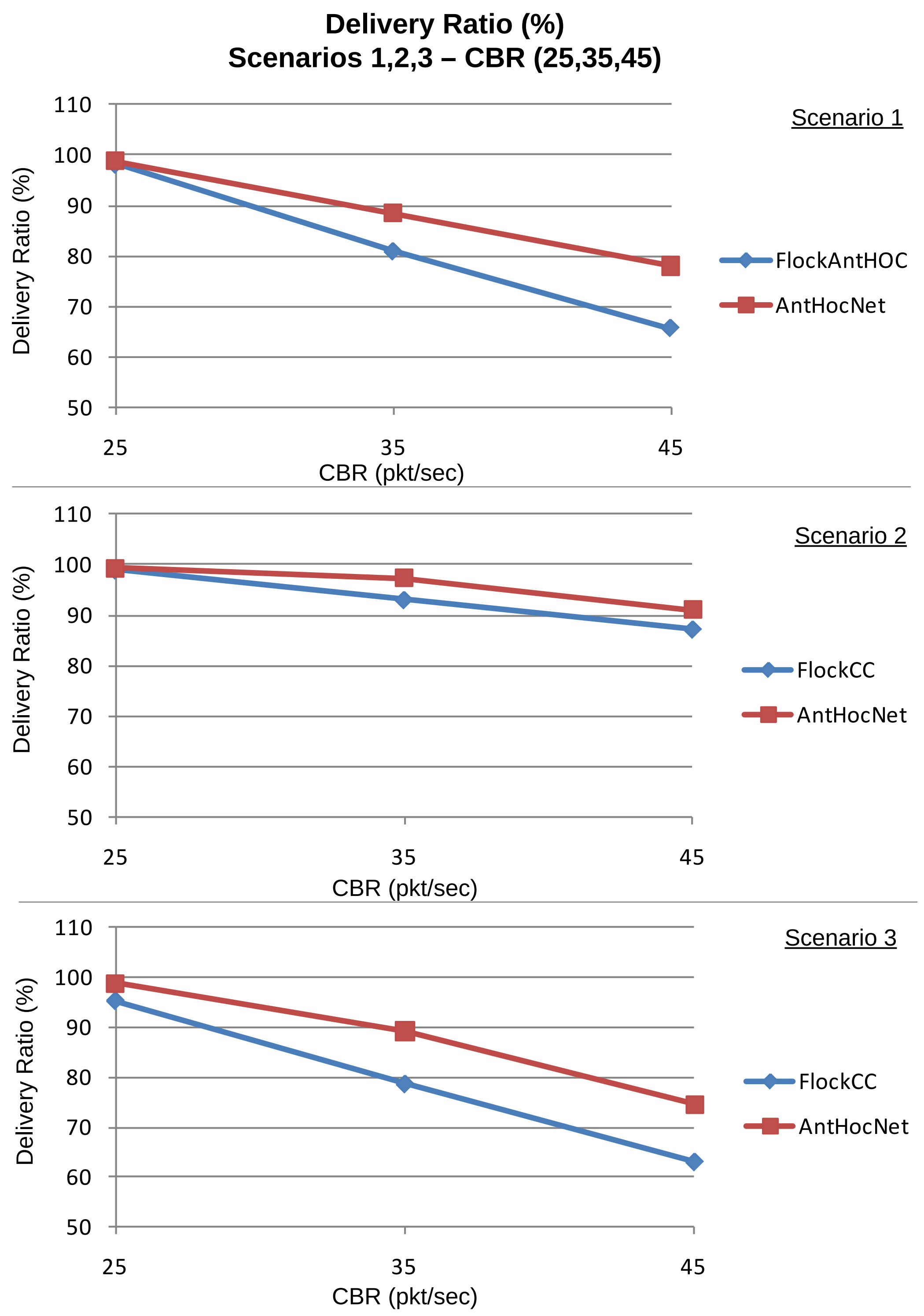

Figure 16: Delivery ratio comparison, FlockCC v.s.AntHocNet

The results of the End-to-End delay depicted at Figure 17 shows a similar trend as the delay in the case of the AntHocNet is less than in the FlockCC for all the scenarios especially when the amount of congestion is high in the network.

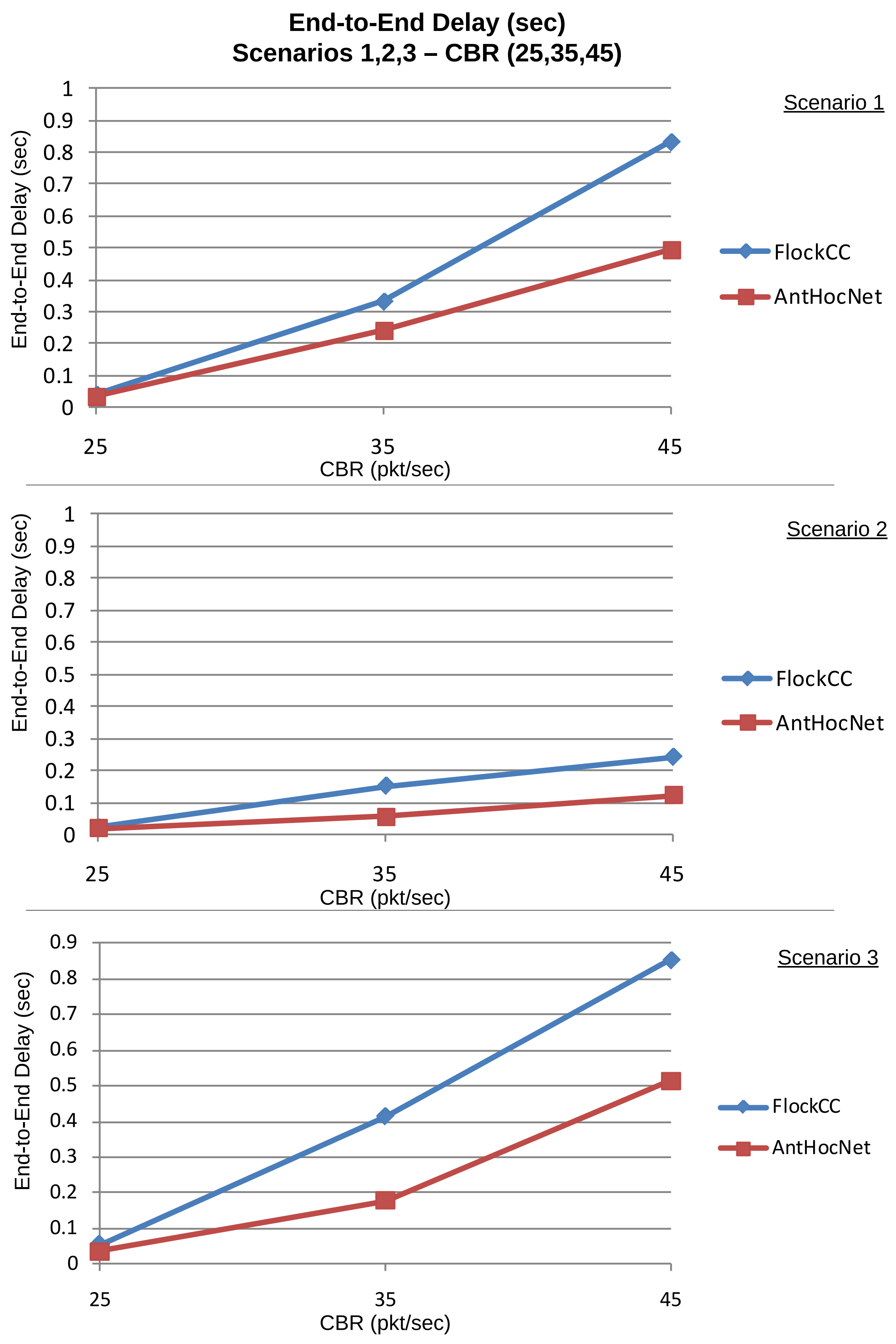

Figure17: End-to-End delay comparison FlockCC v.s. AntHocNet

The mean energy results depicted at Figure 18 show that in general the amount of energy consumed by the AntHocNet protocol is less than the one consumed by the FlockCC. The significance of this is that AntHocNet consumes less energy while at the same time achieves better performance (in terms of delivery ratio and end-to-end delay) for the tested scenarios.

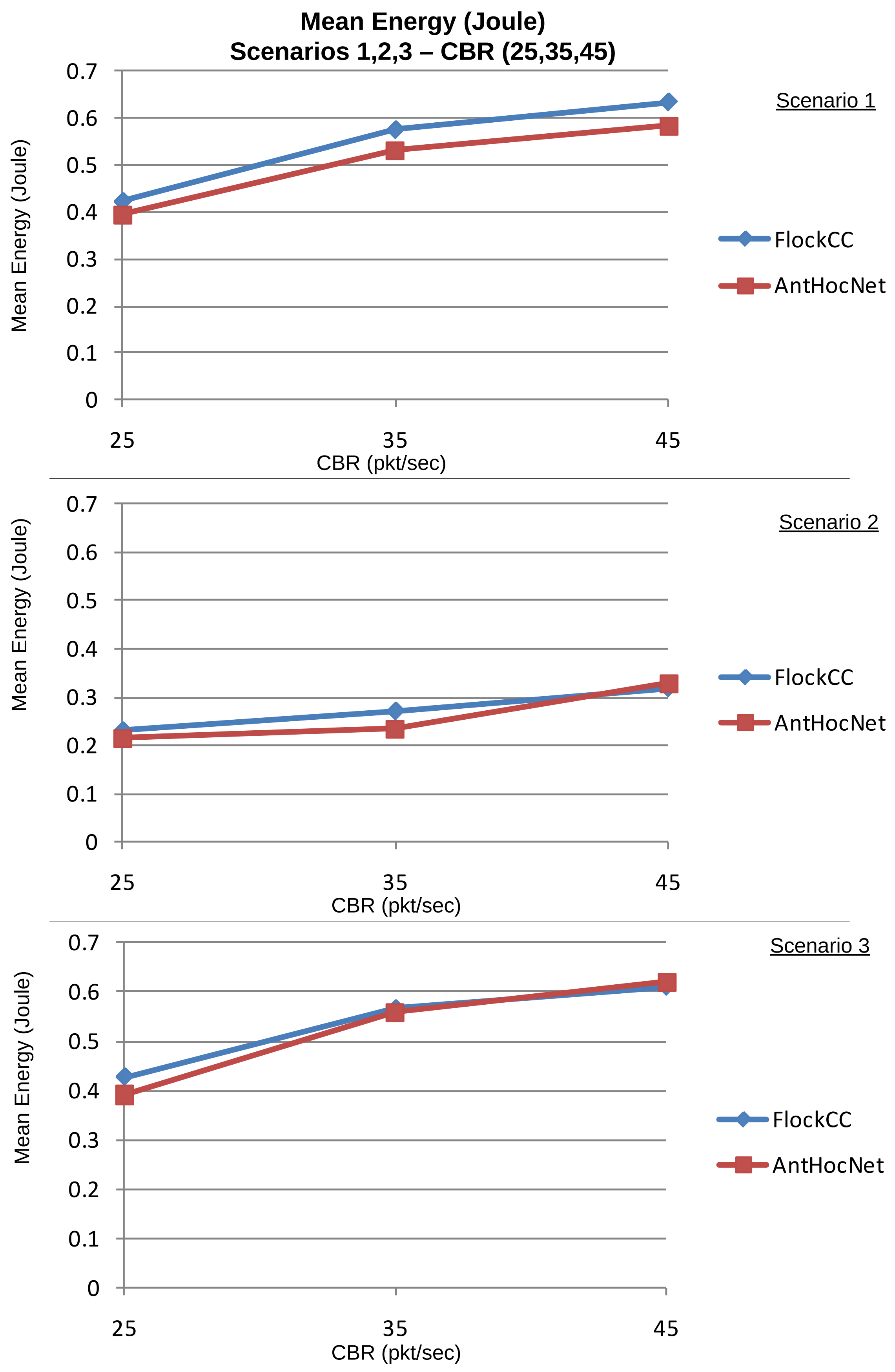

Figure 18: Mean energy comparison, FlockCC v.s. AntHocNet

4.10 Complexity and Storage requirement comparison

Comparing the ANTHOCNET and FlockCC protocols from a computational aspect, we find that the ANTHOCNET is more complex in terms of the structure and number of main operations performed by the protocol and the complexity of each operation (number of calculations involved). Next we provide a quantitative approximation of the computational aspects for both protocols.

The ANTHOCNET performs the following main operations:

- For every reception of a control packet:
    - It stores the received packet and checks if it was received so as to ignore it
    - If the packet pertains to a reactive forward or proactive forward stage, then the routing agent performs operations of complexity in the order of multiples of M (M is the number of neighbors for that node). The operations include basic mathematical operations in addition to power/exponentiation operations.
    - If the packet pertains to a reactive backward or proactive backward stage, the routing agent performs calculation in order of a dozen basic mathematical operations.
    - If the packet pertains to the diffusion stage, the routing agent performs calculations in the order of M mostly basic mathematical operations.
- For every forward of a data packet the routing agent performs calculations of complexity in the order of multiples of M operations. The operations include basic mathematical operations in addition to exponentiation operations. In the case no information is found the data packet is stored in the node and a broadcast is initiated (the first step of the reactive stage).
- For every timer expiration event (governed by the sampling rate parameter) the routing agent performs calculations of complexity in the order of multiples of M operations. The operations include basic mathematical operations in addition to exponentiation operations. In addition broadcasting of diffusion packets are

performed.

On the other hand the FlockCC protocol performs the following main operations:

- For every reception of a control packet the routing agent performs M simple operations to update its neighbor information.
- For every forward of a data packet the routing agent performs calculations of complexity in the order of a dozen M operations. Part of these operations includes square root and log operations in addition to random number generation, all used to calculate the Gaussian mean.
- For every timer expiration event (governed by the sampling rate parameter) the routing agent performs M simple operations to access the neighbor information.

Note that the ANTHOCNET also involves extra storage overheads as every agent on a node stores the identification (mainly using the sequence number of the packet) of received control packets so as not to receive multiple packet originating from the same source for the reactive forward ant stage. Moreover, when receiving data packet and no routing information is available the data packet has to be stored as well. Another factor to consider is that the size of the control packet in the case of the ANTHOCNET is larger than that in the FlockCC as the packet stores its route as it flows from the source to the sink and keeps this information on the way back.

4.11 Discussion and Limitations

The overall results show the validity of our implementation of the AntHocNet protocol and its efficiency in the context of WSNs. It is important to note, however, that although the AntHocNet protocol achieved better performance compared to the FlockCC (a well established and robust routing protocol), this is only true for the given scenarios and network configuration. More investigation is needed to establish the merit of the AntHocNet protocol over the FlockCC or vice versa.

Furthermore, when collecting the results during our simulations, we have noticed that the outcome of the AntHocNet simulation runs in terms of delivery ratio was not always stable. As there existed a difference between the results of consecutive runs of the same scenario/configuration. Although getting small differences in the results of such experiments is normal, we noticed that the difference was in many cases larger than that in the case of the FlockCC. For example, some of the consecutive runs had differences of approx. 20%. To mitigate this issue, we have run every case at least 3 to 5 times and reported the average. And in the cases where we noticed the difference between the runs to be large we ran that case more times to have a more representative average. On the other hands the results of the FlockCC were much more stable and robust with minor differences. Although we can partly attribute this fluctuation in the performance of the AntHocNet to the element of randomness existing in the protocol, more investigation is needed to study this issue satisfactorily. We leave that for future work.

Chapter 5

Conclusion

5.1 Conclusion

In this thesis we have presented a detailed description of our implementation of the biologically inspired AntHocNet routing protocol in the context of WSNs. The protocol was implemented within the NS-2 network simulator and thoroughly validated and evaluated using a number of scenarios under a variety of congestion conditions and combination of control parameters. A detailed investigation of the inner workings of the protocol was also performed. Finally, we compare the performance of the protocol with the FlockCC protocol (a robust, biologically inspired routing protocol) and showed that for the tested scenarios/configurations the AntHocNet achieves better performance in terms of delivery ratio, end-to-end delay and mean energy. The results show the validity of our implementation and indicate the potential of this protocol in the context of WSNs.

5.2 Future Work

More work is to be done before the full merits on the AntHocNet protocol in the context of WSNs is to be established.

Foremost is the evaluation of the protocol under more scenarios that represent a variety of diverse situations. A more ambitious (maybe even exhaustive) exploration of the parameter space in a variety of network configurations (much higher or much lower number of nodes and more extreme values for CBR), higher number of sending nodes and for larger or shorter session times, in addition to the use of other metric for evaluating the performance of the protocol.

Understanding and solving the issue of the performance fluctuation is another important task to be accomplished.

The implementation and evaluation of the link-failure detection and repair components of the AntHocNet which were not done in the current implementation, would be a valuable addition to the robustness of the protocol to handle cases with extreme failure in the network.

Finally, comparison with other routing protocols both traditional and novel would add to the understanding of the strengths and shortcomings of this protocol relative to the already existing ones.

Bibliography

[1] Akyildiz Ian, Su Weilian, Sankarasubramaniam Yogesh, and Cayirci Erdal Georgia Institute of Technology, “A survey on Sensor Networks”, IEEE Communications magazine, Vol. 40(8), pp 102-114, August 2002.

[2] Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: A survey”, Computer Networks Journal, Elsevier Science, Vol. 38(4), pp 393–422, March 2002.

[3] Antoniou Pavlos, Pitsillides Andreas, Andries Engelbrecht, Blackwell Tim, “Mimicking the Bird Flocking Behavior for Controlling Congestion in Sensor Networks”. 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010), Invited Paper, Rome, Italy, November 7-10, 2010.

[4] Antoniou Pavlos, Pitsillides Andreas, Blackwell Tim, Engelbrecht Andries, Michael Loizos, “Congestion Control in Wireless Sensor Networks based on the Bird Flocking Behavior”, IFIP 4th International Workshop on Self-Organizing Systems(IWSOS 2009), Zyrich, Switzerland, December 9-11, 2009, pp. 200-205.

[5] Chonggang Wang and Kazem Sohraby, University of Arkansas Bo Li, The Hong Kong University of Science and Technology, Mahmoud Daneshmand, AT&T Labs Research Yueming Hu, South China Agricultural University, “A Survey of Transport Protocols for Wireless Sensor Networks”, IEEE Network, Vol. 20, May/June 2006, pp. 34-40.

[6] Chonggang Wang and Kazem Sohraby, University of Arkansas Bo Li, The Hong Kong University of Science and Technology,Weiwen Tang, Sichuan Communication Research Planning & Designing Co., Ltd., Chengdu, China, “Issues of Transport Control Protocols for Wireless Sensor Networks”. In

Communications, Circuits and Systems, 2005. Proceedings. 2005 International Conference on, volume 1, pages 422–426, Arkansas Univ., Fayetteville, AR, USA, May 2005.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: Ascalable and robust communication paradigm for sensor networks. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networks (Mobicom), pages 56-67, Boston, MA, USA, 2000.

[8] Ducatelle, F., Adaptive Routing in Ad Hoc Wireless Multi-hop Networks, PhD thesis, Universita della Svizzera Italiana, Istituto Dalle Molle di Studi sull/Intelligenza Artificiale, 2007.

[9] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris. Performance of multihop wireless networks: Shortest path is not enough. In Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I). Princeton, New Jersey: ACM SIGCOMM, October 2002.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz, Inspiration for optimization from social insect behavior, Nature, Vol. 406, 2000, pp. 39-42.

[11] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. IEEE Personal Communications.6(2):46–55, April 1999. Proceedings of IEEE ICC'98, pages 156–160, August 1998. [7] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[12] G. Di Caro, F. Ducatelle and L. M. Gambardella, AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks, European Trans. On Telecommunications, Self-organization in Mobile Networking, 16, 2005, pp. 443-455.

[13] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. Computer Networks Journal, Vol. 47:4, pp 445-487, March 2005.

[14] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparisonof Multi-Hop Wireless Ad Hoc Network Routing Protocols. In Proceedings of the Fourth Annual ACM/IEEEInternational Conference on Mobile Computing and Networking, pages 85-97. ACM/IEEE, October 1998.

[15] J. Liu, L. F. Perrone, Y. Yuan, and D. Nicol. The simulator for wireless ad hoc networks (SWAN). Bucknell University, 2005. Available from: http://www.eg.bucknell.edu/swan.

[16] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. Artificial Life, 5(2), pp. 137-172, 1999.

[17] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. Naturwissenschaften, 76(12), pp. 579-581, 1989.

[18] W. C. Y. Lee. Mobile Communications Engineering: Theory and Applications. McGraw Hill Professional, 1997.

[19] UC Berkeley, LBL, USC/ISI, and Xerox PARC. The ns Manual. Available from: http://www.isi.edu/nsnam/ns/ns-documentation.html.

[20] OPNET Technologies, Inc. OPNET Users' Manual. Avilable from: http://www.opnet.com.

[21] Scalable Network Technologies, Inc. QualNet Simulator, Version 3.8, 2005. Available from: http://www.scalable-networks.com.

[22] http://www.stafflogic.eu/faq/ant-colony-optimization/

[23] http://en.wikipedia.org/wiki/Network_congestion

Appendix A

In this appendix we list a summary of the changes we made for the ns-2 simulator code to implement the AntHocNet protocol. We have used the ns-2 version 2.31 for our implementation.

1-We have added the following files in a new directory in the simulator:

> A – antcc.cc, antcc.h and antcc_pkt.h. Contain the code that implements the AntHocNet protocol encapsulated as an Agent class.
>
> B- mytraffic2.cc mytraffic2.cc.h. Contains the code that implements the traffic agent which generates the data packet based on the chosen CBR values
>
> C- antcctest.tcl. This tcl script has the code that initializes the simulator object and runs the different scenarios of the simulation.

2-We have added a number of constants/functions/macros in the ns simulator files:

> A- added a number of defintions for the antcc protocol (PT_ANTCC identifier, name, HDR_ANTCC_PKT(p) macro) in common/packet.h
>
> B - in cmu-trace.cc & cmu-trace.cc.h:
>
> - we added a function to print the information of the antcc packet that will be used for the tracing
> - added a function format_antcc(Packet *p, int offset) in the CMUTrace class
> - added a call to this function in the main format function in the CMUTrace class
>
> C - in the Makfile, added one line for the mytraffic2 and antcc agents:
>
> ants/antcc.o ants/mytraffic2.o \

D- in ns-default.tcl, we added a number of definitions for the mytraffic2 and antcc agents

```
Agent/Antcc set packetSize_ 10           ;# control packet size // sandy
Agent/MyTraffic2 set CBR_packetSize_ 50   ;# data packet size    // sandy
Agent/MyTraffic2 set thisID_ 0                    ;# node ID               // sandy
Agent/MyTraffic2 set sinkID_ 0          ;# sink ID for active nodes // sandy
```

E- in ns-packet.tcl, added a defintion to the protocol list

```
...
foreach prot {
Flockcc
Antcc
AODV
ARP
...
```

F- in ns-lib.tcl, added the tcl procedure "create-antcc-agent", and added code for calling this procedure in "create-wireless-node" procedure. Also added code for attaching the mac for the antcc protocol. (for all the stuff we added we actually copied what was done for the flocc).

G- in lattice.tcl, added support for the antcc agent for the node creation code